UNIVERSIDAD DE MÁLAGA

DEPTO. DE LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

TESIS DOCTORAL

PROSPECCIÓN DE DATOS, APRENDIZAJE COMPUTACIONAL Y TÉCNICAS ESTADÍSTICAS PARA LA OBTENCIÓN DE REGLAS

INMACULADA FORTES RUIZ

MÁLAGA, 2001

Escuela Técnica Superior de Ingeniería Informática
Departamento de Lenguajes y Ciencias de la Computación

Prospección de datos, aprendizaje computacional y técnicas estadísticas para la obtención de reglas

Inmaculada Fortes Ruiz

Tesis Doctoral

Universidad de Málaga

Noviembre 2001

D. Rafael Morales Bueno

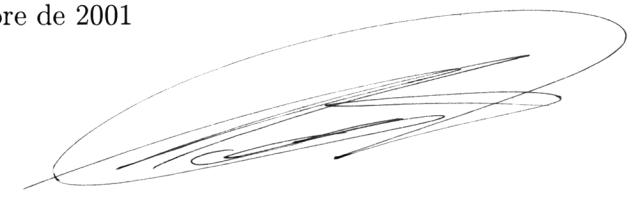
CERTIFICA:

Que Dña. Inmaculada Fortes Ruiz, Licenciada en Ciencias (Sec. Matemáticas), ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la E.T.S. de Ingeniería Informática de la Universidad de Málaga, bajo mi dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulado:

Prospección de datos, aprendizaje computacional y técnicas estadísticas para la obtención de reglas

Revisado el presente trabajo, estimo que puede ser presentado al Tribunal que ha de juzgarlo. Y para que conste a efectos de lo establecido en el artículo octavo del Real Decreto 778/1998, autorizo la presentación de este trabajo en la Universidad de Málaga.

Málaga, a 23 de Noviembre de 2001



Dr. Rafael Morales Bueno

Quiero expresar mi agradecimiento a todas las personas que han hecho posible la realización de este trabajo y sobre todo a Rafael Morales por su confianza desde el primer momento y por poder trabajar con él. A José Luis Balcázar por las conversaciones tan enriquecedoras que han hecho posible parte de esta memoria. También quiero agradecer a los miembros del grupo de investigación $(\mathrm{I\,A})^2$ su apoyo y orientación.

A mis compañeros del Departamento de Matemática Aplicada por los ánimos que me han transmitido y en especial a Inma P. de Guzmán por su preocupación para que vaya siempre hacia adelante.

No puedo olvidarme de dar las gracias a mis padres, Águeda y Manuel, por su cariño y dedicación para que tuviera una formación y una educación.

Quiero agradecer de forma especial a Sixto su paciencia, apoyo, constancia y su "estar ahí" en todos los momentos.

Índice General

Introducción

Este trabajo se enmarca dentro del área de descubrimiento de conocimiento, concretamente en el área de la prospección de datos. En los últimos años, la capacidad de generar y almacenar datos ha crecido de una forma considerable llegando a cantidades enormes. Se hace muy difícil interpretar la información almacenada en estos grandes volúmenes de datos. Los métodos tradicionales de consulta en el área de bases de datos como SQL (Structured Query Language), sólo dan respuestas informativas que el usuario tiene que interpretar. El análisis multivariante estadístico tiene problemas de intratabilidad y las técnicas usadas hasta ahora en Aprendizaje Automático (*Machine Learning*) resultan insuficientes.

Por lo tanto, surge la necesidad de desarrollar nuevas técnicas con la capacidad para analizar de manera automática grandes volúmenes de datos y cuyo fin es obtener información útil y valiosa. Todas estas técnicas son el objeto de estudio del área de descubrimiento de conocimiento (*Knowledge Discovery in Databases*) [Fayyad et al., 1996]. Una definición clásica dada por Fayyad, Piatetsky-Schapiro y Smith en 1996 es la siguiente:

> *"El Descubrimiento de Conocimiento en Bases de Datos es el proceso no trivial de identificar patrones en los datos que sean válidos, nuevos y potencialmente útiles, además de que sean comprensibles."*

Dentro del proceso del descubrimiento de conocimiento existen varias

fases:

1. Selección de los datos.
2. Preprocesamiento de los datos.
3. Transformación de los datos.
4. Prospección de datos (*Data Mining*).
5. Interpretación y evaluación de los resultados.

La fase de prospección de datos determina los métodos, algoritmos y técnicas para obtener cierto conocimiento sobre un problema, a partir de un conjunto de experiencias estructuradas del mismo. Los algoritmos utilizados para encontrar patrones significativos son limitados por cuestiones de eficiencia computacional. Esta eficiencia está ligada al número de veces que hay que recorrer la base de datos y a los cálculos que hay que realizar en cada pasada. Las tareas y métodos que existen en la prospección de datos son: asociación, clasificación, regresión, agrupación (*clustering*), secuenciación, prospección en texto y en la red, etc.

La prospección de datos la podemos dividir en dos grandes áreas: Aprendizaje Automático y Técnicas Estadísticas. Dentro del Aprendizaje Automático se encuadra el Aprendizaje Inductivo. Una posible clasificación de los distintos tipos de aprendizaje que engloba es:

Aprendizaje Inductivo:

- No Supervisado
- Supervisado:
  - Con Conocimiento Adicional
  - Sin Conocimiento Adicional:
    - Con Modelo:

- Exacto
- Aproximado

* Sin Modelo

Tenemos aprendizaje no supervisado cuando las experiencias de nuestro problema no están marcadas. Si las experiencias están marcadas, tenemos aprendizaje supervisado que puede ser con conocimiento adicional (cuando hay aspectos de la estructura del problema que nos son conocidos) y sin conocimiento adicional dentro del cual hay dos tipos, el de con modelo y el de sin modelo. Es aprendizaje con modelo cuando suponemos algún modelo teórico sobre el concepto que vamos a aprender. El aprendizaje con modelo puede ser a la vez exacto o aproximado. Será exacto si exigimos que el concepto aprendido sea idéntico al concepto objetivo. Por último tenemos el aprendizaje sin modelo, es decir, no disponemos de ningún modelo teórico sobre el concepto que vamos a aprender.

En este trabajo ponemos nuestro interés en las tareas de asociación, clasificación y prospección en la red. Dentro de la tarea de asociación nos centramos en una de las subrutinas con más aplicación: la de los algoritmos de búsqueda de reglas de asociación. Este apartado se encuadra dentro del aprendizaje no supervisado. Dentro de la tarea de clasificación nos centramos en los algoritmos de aprendizaje inductivo de árboles de decisión que se encuadrarían dentro del aprendizaje supervisado, sin conocimiento adicional y sin modelo. Por último, el apartado de prospección en la red lo tratamos en este trabajo dentro del aprendizaje supervisado, sin conocimiento adicional y con modelo.

Pasamos ahora a comentar brevemente cada uno de estos apartados haciendo un resumen de sus antecedentes. Acabaremos con una sección donde se exponen las aportaciones hechas en cada apartado.

Asociación: Conjuntos frecuentes

Históricamente la búsqueda de reglas de asociación vino motivada por el análisis de la gran cantidad de datos que se generan sobre las compras que se efectúan en un supermercado. Si las transacciones de la base de datos son las compras de los clientes y los registros corresponden a los productos del supermercado, entonces se intentan buscar reglas de asociación del tipo "aperitivos $\Rightarrow$ refrescos(87%)". Esta regla se interpreta como que el 87% de los clientes que compraron aperitivos también compraron refrescos. En este caso, los conjuntos frecuentes que se buscan son los conjuntos de productos que se compran frecuentemente en el supermercado.

Desde que Agrawal et al. [Agrawal et al., 1993] introdujeron el problema de encontrar reglas de asociación, se han desarrollado diversos algoritmos. Se pueden buscar muchos tipos de reglas de asociación, como por ejemplo las cuantitativas [Srikant y Agrawal, 1996], las generalizadas [Hipp et al., 1998], [Srikant y Agrawal, 1995]. También se ha estudiado la búsqueda de patrones en secuencias [Agrawal y Srikant, 1995], [Mannila et al., 1997]. Además, se han desarrollado algoritmos para buscar conjuntos frecuentes a partir de atributos específicos o mediante medidas de calidad; por ejemplo, con restricciones [Ng et al., 1998], [Srikant et al., 1997], con optimizaciones [Fukuda et al., 1996], [Rastogi y Shim, 1999], etc.

El problema clásico consiste en encontrar reglas de asociación en una base de datos booleana a modo de tabla. Cada fila de la tabla corresponde con una transacción y cada columna con un atributo. La i-ésima entrada de una fila puede ser un 1 o un 0, dependiendo si el atributo está o no presente en la transacción. Esto se formaliza en términos de funciones parciales, las cuales en cada atributo, pueden incluir el atributo (valor 1), excluirlo (valor 0), o no considerarlo (indefinido). Una regla de asociación es una expresión de la forma $X \Rightarrow Y$, donde X e Y son conjuntos de atributos. El significado intuitivo de esta regla de asociación es que los datos de la base de datos que contienen a X tienden a contener a Y. Una regla es evaluada por un

soporte y una confianza. El soporte es el porcentaje de datos que contienen a X y a Y. La confianza es el porcentaje de los datos que contienen a Y entre los que contienen a X.

Otras medidas para evaluar una regla de asociación pueden ser: convicción [Brin et al., 1997], interés [Brin et al., 1997], [Agrawal y Yu, 1998], *lift* [Bayardo et al., 1999] y factor de certeza [Berzal et al. 2001]. Por último, en [Gras, 1996] se introduce la "implicación estadística" para hacer un refinamiento de la medida convicción. Recientemente, se han propuesto dos modelos como alternativa al modelo soporte–confianza. Uno se basa en la correlación entre atributos [Brin et al., 1997a] y el otro propone la medida *collective strength* [Agrawal y Yu, 1998].

El problema de encontrar las reglas de asociación en el modelo de soporte–confianza es un problema de enumeración y tiene dos pasos fundamentalmente que son: encontrar todos los conjuntos de atributos frecuentes con un soporte mínimo y generar las reglas que tienen confianza alta.

Algoritmos para la búsqueda de conjuntos frecuentes

La mayoría de los algoritmos existentes para la búsqueda de conjuntos frecuentes se pueden clasificar por su estrategia para organizar la búsqueda de conjuntos frecuentes y por su estrategia para determinar las frecuencias de los conjuntos de atributos. Por su estrategia para organizar la búsqueda de los conjuntos frecuentes, se dividen en algoritmos de búsqueda del primero en profundidad (*depth-first search* DFS) o búsqueda del primero en amplitud (*breadth-first search* BFS). Por su estrategia para determinar la frecuencia de los conjuntos de atributos, se dividen entre los que buscan la frecuencia contando directamente sus ocurrencias en la base de datos o los que lo hacen mediante intersecciones de listas de transacciones de conjuntos.

En la figura 1 se muestra la clasificación de los algoritmos más habituales [Hipp et al., 2000]. Para un mayor detalle ver [Agrawal y Srikant, 1994]

Clasificación			
BFS		DFS	
Recuento	Intersección	Recuento	Intersección
Apriori AprioriTID DIC Ready and Go	Partition	FP-growth	Eclat

Figura 1: Clasificación de los algoritmos

(Apriori), [Agrawal y Srikant, 1994] (AprioriTID), [Brin et al., 1997] (DIC), [Baixeries et al., 2000] (Ready and Go), [Savsere et al., 1995] (Partition), [Han et al., 2000], (FP-growth) y [Zaki et al., 1997] (Eclat). Los algoritmos propuestos en [Domingo et al., 1999], [Toivonen, 1996] realizan un muestreo por la base de datos.

En [Gunopulos et al., 1997] y [Mannila y Toivonen, 1997] se establece esencialmente que los algoritmos con la misma estrategia de poda que Apriori se pueden usar en otros entornos en los que se busca una teoría o lenguaje formal de acuerdo con una relación monótona de generalización/especialización y un cierto predicado. Problemas de cómo convertir una fórmula booleana monótona en formal normal conjuntiva, a forma normal disyuntiva y cómo encontrar el transversal de un hipergrafo, están estrechamente relacionados con la búsqueda de conjuntos frecuentes.

Clasificación: Árboles de decisión

La idea original de generación de un árbol de decisión a partir de un conjunto de experiencias se debe a [Hunt et al., 1966]. En ese artículo se describen experimentos con varias implementaciones de *concept learning systems* (CLS). Otros algoritmos de generación de árboles de decisión son los siguientes: ID3 [Quinlan, 1979], [Quinlan, 1983], [Quinlan, 1986],

[Quinlan, 1986a], PLS1 y ASSISTANT86 [Cestnik y Kononenko, 1987] y el desarrollado en [Breiman et al., 1984]. Los algoritmos básicos TDIDT (*Top Down Induction of Decision Tree*) [Quinlan, 1986], [Quinlan, 1986a] suponen que todos los valores de los atributos en las experiencias son conocidos.

Si las experiencias contienen valores desconocidos, existen dos puntos de vista para tratar los valores desconocidos: aprendizaje inductivo y análisis estadístico de datos. Desde el aprendizaje inductivo se han hecho varias aproximaciones y se han comparado empíricamente en [Quinlan, 1989]. La conclusión es que no se puede considerar que haya un enfoque superior a otro. Existen dos enfoques comunes: rellenar los valores desconocidos con un valor fijo o rellenar los valores desconocidos con un valor usando la distribución de probabilidad de los valores de los atributos. En el primer enfoque el valor se obtiene a partir de las experiencias que se encuentren en el nodo actual, por ejemplo el valor más probable [Friedman, 1977]. El segundo enfoque es el que se sigue en C4.5 y C5 [Quinlan, 1992] que consiste en la misma idea que ASSISTANT86 [Cestnik y Kononenko, 1987]. Ambos enfoques sólo consideran los valores del atributo.

Desde el punto de vista de análisis estadístico de datos, los valores desconocidos han sido principalmente estudiados por Little y Rubin. En el trabajo [Little y Rubin, 1987] se lleva a cabo un estudio previo de la distribución de los datos y se estiman los parámetros de la distribución. A partir de ese momento los valores desconocidos son reemplazados por los valores estimados. Entonces los parámetros son de nuevo estimados y el proceso iterativo continúa hasta la convergencia. El mismo enfoque considerado en [Little y Rubin, 1987] se usa en redes Bayesianas [Heckerman, 1997].

Prospección en la red: Inferencia de gramáticas

El crecimiento en el uso de la World Wide Web, conocida como Web, ha hecho que exista una gran cantidad de datos almacenados. Por tanto,

se pueden aplicar las técnicas de prospección de datos a la Web en lo que se ha denominado *Web data mining*.

En la prospección de datos en la red existen dos ramas de investigación. En la primera se realiza la prospección para obtener información y se desarrollan técnicas que ayuden al usuario a encontrar la información que está buscando en su navegación. La segunda consiste en hacer la prospección para obtener patrones de navegación del usuario, es decir, desarrollar técnicas para estudiar el comportamiento del usuario cuando está navegando por la Web. Estas rutas de navegación se almacenan en ficheros logs también llamados históricos. Existen dos enfoques para realizar la prospección de los patrones de navegación de los usuarios. En el primero los ficheros logs se transforman en tablas relacionales y se usan las técnicas de prospección de datos usuales [Chen et al., 1998]. En el segundo enfoque se supone que se pueden modelar los datos mediante un modelo formal y se intenta inferir ese modelo para que capture los patrones de conducta del usuario. Nos centramos en el caso en el que se supone que el modelo formal es una gramática regular y nombramos ahora algunos de los trabajos que se han hecho sobre inferencia de gramáticas.

En [Borges y Levere, 1999] se trabaja con ficheros logs y se supone que el modelo del cual provienen los datos es una gramática regular probabilística hipertexto que es una clase de las gramáticas regulares probabilísticas.

En el campo del aprendizaje computacional existen otras aproximaciones para la inferencia de gramáticas. En el caso en que se tiene información adicional se puede usar el modelo PAC (*Probably Aproximately Correct*) que fue introducido por Valiant [Valiant, 1984]. En el modelo, existe un oráculo al que se le hacen preguntas de pertenencia para saber si un elemento pertenece al concepto que queremos aprender. El oráculo nos va respondiendo sí o no, y cuando se tiene una cantidad de información suficiente para determinar el concepto con un cierto grado de confianza establecido, se presenta el concepto aprendido.

Posteriormente Angluin [Angluin, 1987], demostró que se podía aprender la clase de los autómatas finitos deterministas. En 1988, Angluin introdujo las preguntas de equivalencia [Angluin, 1988], que consistían en preguntar al oráculo si el concepto que se presentaba después de las preguntas de pertenencia, era equivalente al que queríamos aprender. Si lo era, habíamos hecho el aprendizaje con éxito, y si no era así, el oráculo proporcionaba un contraejemplo con el que se actualizaba el concepto y se volvía a hacer la pregunta de equivalencia. Se repetía este proceso hasta conseguir el aprendizaje del concepto de una forma exacta.

Más tarde en 1994, Kearns y Vazirani [Kearns y Vazirani, 1994], refinaron el modelo de aprendizaje para los AFD. Conservaron el oráculo, las preguntas de pertenencia y de equivalencia, pero introdujeron una nueva técnica (árboles discriminantes), para almacenar toda la información que se generaba en el proceso de aprendizaje. En 1997 esta técnica fue generalizada por Balcázar et al. [Balcázar et al., 1997].

En este trabajo consideramos que no tenemos información adicional. En ese caso considerando la equivalencia entre gramáticas regulares y autómatas finitos, distintas aproximaciones a este problema ven el autómata como un árbol por donde se va haciendo un recorrido desde la raíz hasta las hojas y la elección de caminar por una rama u otra depende de la estructura que tenga el prefijo de la palabra que estamos considerando como ejemplo [Angluin, 1982], [Carrasco y Oncina, 1994],[Castellanos et al., 1994]. En particular Biermann y Feldman [Biermann y Feldman, 1972] tratan de obtener el autómata finito determinista, estudiando el comportamiento de la cola de una longitud dada del ejemplo considerado en ese momento.

Otros enfoques para extraer patrones a partir de una muestra de datos son los métodos que se encuadran dentro del descubrimiento de modelos de procesos software [Cook y Woolf, 1996]. El descubrimiento de modelos de procesos es esencialmente el análisis de los datos recopilados de un proceso, para producir automáticamente un modelo formal que lo describa. En [Cook y Woolf, 1996] se considera que el proceso se puede describir como

una gramática regular.

Aportaciones

Como aportaciones con respecto a la búsqueda de conjuntos frecuentes con atributos negativos podemos citar:

- **Definición de un lenguaje, una relación de especialización y un predicado adecuados para la formalización del problema de la búsqueda de información negativa frecuente.**
- **Un método que minimiza el esfuerzo computacional en la búsqueda de conjuntos frecuentes obteniendo la frecuencia de itemsets a partir de la frecuencia de determinados itemsets evitando así la evaluación explícita en la base de datos.**
- **Una familia de algoritmos que hemos llamado Neg–Apriori** [Fortes et al., 2001] **para la búsqueda de los conjuntos frecuentes con atributos negativos y una familia de algoritmos llamados Acot–Neg–Apriori para la búsqueda de conjuntos frecuentes con un número acotado de atributos negativos. Los algoritmos desarrollados son de complejidad polinómica en el número de pasadas por la base de datos.**

Respecto a la tarea de la clasificación y concretamente a la generación de árboles de decisión, **desarrollamos técnicas para incluir las experiencias con valores desconocidos en los atributos en el proceso de generación del árbol de decisión** [Fortes et al,. 1999a], [Fortes et al., 2000a]. Las aportaciones en la adaptación de los algoritmos de generación de árboles de decisión para incluir las experiencias con valores desconocidos son:

- **Definición de un criterio general de división para tener en cuenta los valores desconocidos.**

- **Asignación de valores a los valores desconocidos de los atributos teniendo en cuenta la información de los valores del atributo y los valores de la clase.**

- **Predicción de observaciones con valores desconocidos.**

- **Modelado del problema de asignación de valores como un problema de decisión con riesgo. Asignación de parámetros de confianza y de error.**

Por último con respecto a la inferencia de gramáticas nuestras aportaciones son:

- **Un algoritmo de inferencia de gramáticas regulares probabilísticas** [Fortes et al., 1998], [Fortes et al., 1998a] **y gramáticas regulares difusas** [Fortes et al., 1998b], [Fortes et al., 1999]. **Los algoritmos desarrollados son polinómicos en el número de símbolos del alfabeto**.

- **Obtención de la cadena de Markov asociada a una gramática regular probabilística. Estudio estadístico mediante la cadena de Markov de las propiedades del lenguaje generado por la gramática** [Fortes et al., 2000].

Para finalizar esta introducción, detallamos la estructura del trabajo:

En el **capítulo 1** hacemos un rápido recorrido por las nociones básicas utilizadas en los siguientes capítulos. De esta forma intentamos que el trabajo sea autocontenido y que la notación esté introducida desde el primer momento.

En el **capítulo 2** incluimos como primera aportación de este trabajo la generalización de la búsqueda en bases de datos de conjuntos frecuentes con

atributos positivos y negativos. En primer lugar, se formaliza el problema introduciendo un lenguaje, una relación de especialización y un predicado adecuados. A partir de ahí, se introducen las definiciones necesarias para los itemsets con atributos negativos para a continuación presentar las proposiciones que nos llevarán a los algoritmos que se han desarrollado para obtener todos los conjuntos frecuentes con el menor esfuerzo computacional, es decir, evitando evaluar directamente cada conjunto en la base de datos y calculando su frecuencia a partir de las frecuencias ya calculadas de determinados conjuntos.

En el **capítulo 3** adaptamos los algoritmos de aprendizaje de árboles de decisión para considerar las experiencias que tienen valores desconocidos en algún atributo. Definimos un criterio general de división, asignamos valores a los valores desconocidos teniendo en cuenta la información dada por los valores del atributo y de la clase y estudiamos la predicción de observaciones con valores desconocidos. Incluimos resultados experimentales sobre la evaluación de nuestro método en distintas bases de datos. Por último, modelamos el problema de la asignación de valores como un problema de decisión con riesgo para incluir parámetros de confianza y de error en la construcción del árbol de decisión.

En el **capítulo 4** desarrollamos la tercera tarea destacada en esta introducción. Concretamente presentamos los algoritmos de inferencia de gramáticas regulares probabilísticas y gramáticas regulares difusas. También se estudia la obtención de la cadena de Markov asociada a la gramática regular probabilística y cómo a partir de la cadena se realiza un estudio estadístico del lenguaje generado por la gramática. Se incluye un caso práctico donde hemos implementado nuestros algoritmos.

Finalmente se han añadido unos **apéndices** con los algoritmos de búsqueda de conjuntos frecuentes con un número máximo de atributos negativos. También se incluye un algoritmo que organiza la búsqueda de conjuntos frecuentes con atributos negativos de forma diferente a los algoritmos dados en el capítulo 2.

Capítulo 1

Preliminares

En este capítulo introducimos los conceptos básicos utilizados a lo largo del trabajo.

1.1 Conceptos matemáticos

1.1.1 Conjuntos parcialmente ordenados

Para los conjuntos A y B (posiblemente vacíos) se define el *producto cartesiano* de A por B, denotado $A \times B$, como sigue:

$$A \times B = \begin{cases} \emptyset & \text{si } A = \emptyset \text{ ó } B = \emptyset \\ \{(a,b) \mid a \in A, b \in B\} & \text{en otro caso} \end{cases}$$

Dados dos conjuntos A y B, una *relación* de A en B es cualquier subconjunto $\mathcal{R}$ (posiblemente vacío) de $A \times B$. En particular, si $B = A$, una relación de A en A se llama *relación binaria* en A.

Si $\mathcal{R}$ es un relación binaria en A, como alternativa a $(x,y) \in \mathcal{R}$, es frecuente usar la notación $x\mathcal{R}y$ (leída como x está relacionado con y).

Una relación binaria $\mathcal{R}$ sobre un conjunto A puede verificar, entre otras, las siguientes propiedades:

- *Reflexiva:* Para todo $x \in A$, se tiene que $x\mathcal{R}x$.
- *Simétrica:* Para todo $x, y \in A$, se tiene que si $x\mathcal{R}y$ entonces $y\mathcal{R}x$.
- *Antisimétrica:* Para todo $x, y \in A$, si $x\mathcal{R}y$ e $y\mathcal{R}x$, entonces se tiene que $x = y$.
- *Transitiva:* Para todo $x, y, z \in A$, si $x\mathcal{R}y$ e $y\mathcal{R}z$, entonces se tiene que $x\mathcal{R}z$.

Definición 1.1 *Una relación binaria $\mathcal{R}$ sobre un conjunto A se dice de orden parcial si verifica las propiedades reflexiva, antisimétrica y transitiva. En tal caso, el par $(A, \mathcal{R})$ se dice que es un conjunto parcialmente ordenado y se representa por $(A, \leq)$.*

Por ejemplo, para todo conjunto A se tiene que $(\mathcal{P}(A), \subseteq)$ es un conjunto parcialmente ordenado, siendo $\mathcal{P}(A)$ el conjunto de partes de A y $\subseteq$ la inclusión de conjuntos.

En general, un conjunto $(A, \leq)$ parcialmente ordenado se puede representar mediante un grafo dirigido. Sin embargo, este grafo se puede simplificar eliminando todos los lazos (reflexiva) y las aristas que se puedan deducir de otras (transitividad). Además, si ubicamos los vértices de modo que todas las flechas vayan hacia arriba y convenimos leer el grafo desde abajo hacia arriba no es necesario dirigir las aristas y, en tal caso, el grafo recibe el nombre de *Diagrama de Hasse.*

Definición 1.2 *Sea $(A, \leq)$ un conjunto parcialmente ordenado y sea S un subconjunto de A. Decimos que S es cerrado bajo la relación $\leq$ si para todo $s \in S$ y para todo $a \in A$ se cumple que si $a \leq s$ entonces $a \in S$.*

Definición 1.3 *Sea $(A, \leq)$ un conjunto parcialmente ordenado y sea $m \in A$. Decimos que m es un elemento maximal (resp. minimal) de A si para todo $a \in A$ se tiene que si $m \leq a$ (resp. $a \leq m$) entonces $a = m$.*

Con estas definiciones podemos concluir que un subconjunto cerrado de un conjunto parcialmente ordenado se puede representar enumerando sus elementos maximales. Efectivamente, si M es el conjunto de todos los elementos maximales de un subconjunto cerrado S de un conjunto $(A, \leq)$ parcialmente ordenado, entonces $S = \{a \in A \mid \exists m \in M,\ a \leq m\}$.

1.1.2 Probabilidad

Un *experimento científico* es una acción que da lugar a resultados identificables. Este experimento puede ser *determinista* o *aleatorio*. Las características de un experimento aleatorio son las siguientes:

- Los posibles resultados son conocidos previamente.
- El resultado no es predecible.
- Situaciones análogas pueden dar resultados diferentes.

El *espacio muestral* de un experimento aleatorio es el conjunto formado por todos los posibles resultados del experimento. Un *suceso aleatorio* es un subconjunto de elementos del espacio muestral. Los sucesos pueden ser *elementales* si corresponde a un único resultado simple del experimento o *compuestos* si corresponden a un conjunto de sucesos elementales que representa varios resultados posibles del experimento. Decimos que dos sucesos A y B de un espacio muestral son incompatibles si $A \cap B = \emptyset$.

Definición 1.4 *Sea Ω un espacio muestral y sea $\mathcal{A} \subseteq \mathcal{P}(\Omega)$ una familia de sucesos. Decimos que $\mathcal{A}$ es un* álgebra de Boole *de sucesos si verifica las siguientes condiciones:*

1. $\Omega \in \mathcal{A}$
2. *Si* $A \in \mathcal{A} \Rightarrow \bar{A} \in \mathcal{A}$
3. *Si* $A, B \in \mathcal{A} \Rightarrow A \cup B \in \mathcal{A}$

y en tal caso se dice que $(\Omega, \mathcal{A})$ *es un* espacio probabilizable.

Normalmente, en el caso finito, se considera como álgebra el conjunto $\mathcal{P}(\Omega)$ partes de Ω, es decir, el conjunto formado por todos los subconjuntos de elementos de Ω.

Proposición 1.1 *Si* $(\Omega, \mathcal{A})$ *es un espacio probabilizable entonces se verifican las siguientes propiedades:*

1. *Si* $A_i \in \mathcal{A}$ *para todo* $i = 1, \ldots, n$ *entonces* $\cup_{i=1}^{n} A_i \in \mathcal{A}$.
2. *Si* $A_i \in \mathcal{A}$ *para todo* $i = 1, \ldots, n$ *entonces* $\cap_{i=1}^{n} A_i \in \mathcal{A}$.

Definición 1.5 *Sea* $(\Omega, \mathcal{A})$ *un espacio probabilizable. Decimos que una función* $P : \mathcal{A} \to \mathbb{R}$ *es una* probabilidad *si verifica que*

1. $P(\Omega) = 1$,
2. $P(A) \geq 0$ *para todo* $A \in \mathcal{A}$ *y*
3. $P(\bigcup_{i=1}^{n} A_i) = \sum_{i=1}^{n} P(A_i)$ *si* $\forall i = 1, \ldots, n, A_i \in \mathcal{A}$ *y* $\forall i \neq j, A_i \cap A_j = \emptyset$.

y en tal caso, se dice que $(\Omega, \mathcal{A}, P)$ *es un* espacio de probabilidad.

Proposición 1.2 *Si* $(\Omega, \mathcal{A}, P)$ *es un espacio de probabilidad entonces se verifican las siguientes propiedades:*

1. $P(\bar{A}) = 1 - P(A)$.

2. $P(\emptyset) = 0$.

3. *Si* $A \subseteq B$ *entonces* $P(A) \leq P(B)$.

4. $P(A) \in [0, 1]$.

5. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.

La frecuencia de un suceso en un conjunto es el número de veces que se repite este suceso en el conjunto. La frecuencia se puede considerar en términos absolutos o en términos relativos. La frecuencia absoluta coincide estrictamente con la definición mientras que la frecuencia relativa es la proporción de veces que se repite el suceso.

Definición 1.6 *Sea* $(\Omega, \mathcal{A}, P)$ *un espacio de probabilidad. Un conjunto de sucesos* $\{C_i\}_{i \in I}$ *del álgebra* $\mathcal{A}$ *se llama* clase completa de sucesos *de* Ω*, si constituye una partición de* Ω.

Definición 1.7 *Sea* $(\Omega, \mathcal{A}, P)$ *un espacio de probabilidad y sea* A *un suceso del álgebra con* $P(A) > 0$. *Definimos la* probabilidad del suceso B condicionada al suceso A *como:*

$$P_A(B) = P(B/A) = \frac{P(A \cap B)}{P(A)}$$

Teorema 1.1 (de la probabilidad total) *Sea* $(\Omega, \mathcal{A}, P)$ *un espacio de probabilidad y sea* $\mathcal{C} = \{C_i\}_{i \in I}$ *una clase completa de sucesos tal que* $P(C_i) > 0$ *para todo* $i \in I$. *Si* B *es un suceso cualquiera del álgebra* $\mathcal{A}$ *entonces*

$$P(B) = \sum_{i \in I} P(C_i) \cdot P(B/C_i)$$

Teorema 1.2 (de Bayes) *Sea* $(\Omega, \mathcal{A}, P)$ *un espacio de probabilidad y sea* $\mathcal{C} = \{C_i\}_{i \in I}$ *una clase completa de sucesos tal que* $P(C_i) > 0$ *para todo* $i \in I$. *Si* B *es un suceso cualquiera del álgebra* $\mathcal{A}$ *entonces*

$$P(C_i/B) = \frac{P(C_i) \cdot P(B/C_i)}{\sum_{i \in I} P(C_i) \cdot P(B/C_i)}$$

El teorema de Bayes pone de manifiesto el cambio de la probabilidad asignada a un suceso cuando disponemos de información adicional. Por ello, en las condiciones del teorema de Bayes, las probabilidades que intervienen reciben los siguientes nombres:

- Las *probabilidades a priori* son $P(C_i)$ con $i \in I$.
- Las *verosimilitudes* son $P(B/C_i)$ con $i \in I$.
- Las *probabilidades a posteriori* son $P(C_i/B)$ con $i \in I$.

Variable aleatoria

Ahora vamos a definir las variables aleatorias cuyos valores dependen de los resultados de un experimento. La finalidad de estas variables aleatorias consiste en asociar un número real a cada uno de los posibles resultados de un experimento aleatorio.

Definición 1.8 *Sea $(\Omega, \mathcal{A}, \mathcal{P})$ un espacio probabilístico y sea $\{C_i\}_{i=1}^n$ una clase completa de sucesos de Ω formada por sucesos de $\mathcal{A}$.*

Una variable aleatoria simple es una aplicación, $X : \Omega \to \mathbb{R}$ que toma los valores $X(\omega) = x_i$, si $\omega \in C_i$, donde $x_i \neq x_j$ si $i \neq j$.

Definición 1.9 *Sea $(\Omega, \mathcal{A}, \mathcal{P})$ un espacio probabilístico y sea X una variable aleatoria simple que toma los valores $x_1, x_2, \ldots, x_n$. Definimos la distribución de probabilidad de la variable aleatoria X como el conjunto*

$$\{(x_i, P(x_i)) : i = 1, \ldots, n\} \quad \text{donde} \quad P(x_i) = P(\omega \in \Omega : X(\omega) = x_i).$$

Las variables aleatorias y sus distribuciones de probabilidad, pueden considerarse una generalización del concepto frecuentista de probabilidad. Se introducen como el modelo matemático ideal al que se aproximan las distribuciones de frecuencias que se obtendrían en una repetición indefinida de pruebas de este experimento.

Problema de decisión con riesgo

Damos ahora las definiciones básicas relacionadas con la teoría de la decisión [Berger,1988].

En adelante sea $\Lambda = \{\lambda_j \ : \ j = 1, \ldots, m\}$ un espacio paramétrico y sea $A = \{a_i \ : \ i = 1, \ldots, n\}$ el espacio de acciones.

Definición 1.10 *Se define la función de pérdidas como una función* $L : A \times \Lambda \to \mathbb{R}$. *La matriz definida por* L *se llama la matriz de pérdidas. Sus elementos son* $L_{ij} = L(a_i, \lambda_j)$ *con* $i = 1, \ldots, n$, $j = 1, \ldots, m$.

Definición 1.11 *Un problema de decisión se define como la terna* (Λ, A, L) *donde* Λ *es el espacio paramétrico,* A *es el espacio de acciones y* L *es la función de pérdidas.*

Definición 1.12 *Sea* $\Omega = \{x_1, \ldots, x_s\}$ *un espacio muestral finito. Se define un experimento y se denota por* $\mathcal{E}$ *como la terna* $\mathcal{E} = \{\Omega, \mathcal{P}(\Omega), P\}$ *donde* $P : \Omega \times \Lambda \to [0, 1]$ *es una función que verifica*

$$\sum_{k=1}^{s} P(x_k \mid \lambda_j) = 1 \ \ \textit{para todo } j = 1, \ldots, m$$

Definición 1.13 *Para un problema de decisión se define una regla de decisión como la función* $d : \Omega \to A$ *que a cada suceso observable le asocia una acción. Al conjunto de todas las decisiones lo denotamos por* D.

Definición 1.14 *El riesgo de Bayes se define como la función* $R : D \times \Omega \to \mathbb{R}$ *donde*

$$R(a_i \mid x_k) = \sum_{j=1}^{m} L(a_i, \lambda_j) \cdot p(\lambda_j \mid x_k) \qquad \textit{para todo } k = 1, \ldots, s$$

siendo $d(x_k) = a_i$

1.1.3 Cadenas de Markov

Las cadenas de Markov son de gran utilidad para estudiar la evolución de un sistema. Las definiciones dadas en este apartado son las necesarias pasra la comprensión del capítulo cuatro. Para mayor detalle ver [González y Thomason, 1978].

Definición 1.15 *Sea T un conjunto de índices también llamado* espacio paramétrico. *Un* proceso estocástico *es una familia de variables aleatorias $\{X_t : t \in T\}$ que toman valores en el conjunto S llamado* espacio de estados.

En adelante supondremos que el espacio paramétrico T es discreto y el conjunto de estados S es finito.

Definición 1.16 *Se dice que un proceso estocástico es una cadena de Markov (de estado finito) si para todo $x_1, \ldots, x_n \in S$ se verifica la siguiente propiedad:*

$$P[X_n = x_n | X_1 = x_1, \ldots, X_{n-1} = x_{n-1}] = P[X_n = x_n | X_{n-1} = x_{n-1}]$$

Esta propiedad recibe el nombre de propiedad markoviana.

Definición 1.17 *Se define la probabilidad de transición $P_{ij}^{(n,m)}$ como la probabilidad condicionada*

$$P_{ij}^{(n,m)} = P[X_m = j | X_n = i], \quad \text{con } n < m \quad e \quad i, j \in S$$

Definición 1.18 *Decimos que una cadena de Markov es homogénea si verifica que*

$$P[X_{m+n} = j | X_m = i] = P[X_n = j | X_0 = i] \quad \text{para todo } m \in I$$

es decir, las probabilidades de transición $P_{ij}^{(n,m)}$ sólo dependen de la diferencia entre m y n. En tal caso estas probabilidades de transición se dicen que son estacionarias y se denotan por $P_{ij}^{(n)}$ a la probabilidad condicionada

$$P_{ij}^{(n)} = P[X_n = j | X_0 = i] \quad \text{con } n \in I \quad e \quad i, j \in S$$

Definición 1.19 *Se define la matriz de probabilidades de transición* $\mathbf{P}$ *de una cadena de Markov homogénea como la matriz cuyo elemento* (i, j) *es* $P_{ij} = P[X_{n+1} = j | X_n = i]$.

Las matrices de transición son matrices estocásticas:

1. $0 < P_{ij} < 1$ para todo $i, j \in S$
2. $P_{ij} = 1$ para todo $i \in S$

Definición 1.20 *Se define la matriz de transición en* n *etapas como la matriz* $\boldsymbol{P}^{(n)}$ *cuyos elementos son* $P_{ij}^{(n)}$.

En las cadenas de Markov homogéneas, las *ecuaciones de Chapman-Kolmogorov* proporcionan un método para calcular las probabilidades de transición en n etapas y se deduce que $\mathbf{P}^{(n)} = \mathbf{P}^n$.

Definición 1.21 *Sea* $\{X_0, X_1, X_2, \dots\}$ *una cadena de Markov con conjunto de estados* $S = \{1, 2, \dots, n\}$. *Definimos el vector* $\mathbf{P}(0)$ *de probabilidades iniciales como el vector*

$$\boldsymbol{P}(0) = (P[X_0 = 1], P[X_0 = 2], \dots, P[X_0 = n])$$

Para calcular la distribución de probabilidad de la variable X_n de una cadena de Markov $\{X_0, X_1, X_2, \dots\}$ con conjunto de estados $S = \{1, 2, \dots, n\}$ es necesario que se especifique la matriz de probabilidades $\mathbf{P}$ de transición y un vector de probabilidades iniciales $\mathbf{P}(0)$. En tal caso

$$(P[X_k = 1], P[X_k = 2], \dots, P[X_k = n]) = \mathbf{P}(0) \cdot \mathbf{P}^k$$

Definición 1.22 *Se dice que el estado* j *es accesible desde el estado* i *si* $P_{ij}^{(n)} > 0$ *para algún* $n \geq 0$

Definición 1.23 *Se dice que los estados* i *y* j *son comunicantes o que se comunican si el estado* j *es accesible desde el estado* i, *y además, el estado* i *es accesible desde el estado* j.

Una cadena de Markov se puede representar por su diagrama de transición entre estados que es un grafo dirigido donde los vértices del grafo representan a los estados de la cadena y los arcos indican las transiciones entre los estados.

Como resultado de esta definición de comunicación se puede hacer una partición del conjunto de estados en clases, en donde se dice que dos estados que se comunican pertenecen a la misma clase.

Definición 1.24 *Se dice que una cadena de Markov es irreducible si todos los estados se comunican.*

Definición 1.25 *Sea f_{ii} la probabilidad de que el proceso regrese al estado i dado que comienza en el estado i. En tal caso los estados de una cadena de Markov se clasifican en recurrentes si $f_{ii} = 1$ y transitorios si $f_{ii} < 1$. Un caso especial de caso recurrente es el estado absorbente, donde $P_{ii} = 1$*

Definición 1.26 *Una clase comunicante es cerrada si una vez que la cadena alcanza un elemento de ella no se volverá a salir de los elementos de la clase. Una clase comunicante es abierta si transcurrido un tiempo finito la cadena no vuelve a ninguno de los estados de esa clase.*

Teorema 1.3 (de descomposición para cadenas de Markov finitas) *El espacio de estados de una cadena de Markov finita se puede descomponer de manera única como unión disjunta de clases comunicantes o estados sin retorno donde cada clase es recurrente cerrada o es transitoria abierta. Las clases del primer tipo tienen que existir necesariamente.*

1.2 Itemsets con atributos positivos

En el problema clásico de la búsqueda de reglas de asociación se supone una base de datos booleana a modo de tabla. Cada fila de la tabla corresponde con una transacción y cada columna con un atributo. La i-ésima

entrada de una fila puede ser un 1 o un 0, dependiendo si el atributo está o no presente en la transacción. Esto se formaliza en términos de funciones parciales, las cuales en cada atributo, pueden incluir el atributo (valor 1), excluirlo (valor 0), o no considerarlo (indefinido).

Introducimos las definiciones correspondientes al problema de la búsqueda de reglas de asociación formadas por conjuntos de atributos donde todos sus atributos tienen valor 1.

Definición 1.27 *Consideramos un conjunto R de atributos. Llamamos itemset a cualquier subconjunto de atributos, es decir, a cualquier elemento $X \in \mathcal{P}(R)$.*

Sea $\mathcal{T}$ una base de datos sobre un conjunto R de atributos donde las transacciones $t \in \mathcal{T}$ toman valores en el dominio $\{0, 1\}$. En este caso, cada transacción $t \in \mathcal{T}$ se puede considerar como un conjunto de atributos o itemset donde para cada $A \in R$ diremos que $A \in t$ si y sólo si $t(A) = 1$. Es decir, identificamos la transacción t de la base de datos con el itemset $X = \{A \in R : t(A) = 1\}$.

Definición 1.28 *Sea $\{t_1, \ldots, t_N\}$ el conjunto de transacciones de una base de datos $\mathcal{T}$ sobre un conjunto R de atributos. Se define el soporte o frecuencia de un itemset X y se denota por $fr(X)$ como la proporción del número de transacciones que incluyen a X como subconjunto, es decir,*

$$fr(X) = \frac{|\{t \in \mathcal{T} : X \subseteq t\}|}{N} \qquad \text{para todo} \quad X \in \mathcal{P}(R)$$

siendo $\subseteq$ la relación de inclusión entre conjuntos.

Definición 1.29 *Sea $\mathcal{T}$ una base de datos sobre un conjunto R de atributos y sea σ un valor mínimo del soporte especificado por el usuario. Decimos que un itemset X es σ-frecuente o simplemente frecuente si su soporte es mayor que el soporte mínimo, es decir, si $fr(X) > \sigma$. Denotamos por $Fr(\mathcal{T}, \sigma)$ al conjunto formado por todos los itemsets frecuentes, es decir,*

$$Fr(\mathcal{T}, \sigma) = \{X \in \mathcal{P}(R) \ : \ fr(X) > \sigma\}.$$

Una regla de asociación es una expresión de la forma $X \Rightarrow Y$, donde X e Y son conjuntos de attributos. El significado intuitivo de esta regla de asociación es que los datos de la base de datos que contienen a X tienden a contener a Y. Una regla es evaluada por un soporte y una confianza. El soporte es el porcentaje de datos que contienen a X y a Y. La confianza es el porcentaje de los datos que contienen a Y entre los que contienen X.

Definición 1.30 *Dada una regla de asociación $X \Rightarrow Y$, se define su confianza y se denota por $conf(X \Rightarrow Y)$ al cociente entre la frecuencia de la unión y la frecuencia de X, es decir,*

$$conf(X \Rightarrow Y) = \frac{fr(X \cup Y)}{fr(X)}$$

Definición 1.31 *Dado por el usuario un valor de confianza mínimo, denotado por δ, se dice que una regla tiene confianza mayor que δ si su confianza es mayor o igual que δ, es decir, si $conf(X \Rightarrow Y) \geq \delta$.*

El problema de encontrar reglas de asociación que se enuncia de esta forma:

> *"Dados un soporte mínimo σ y una confianza δ, obtener reglas $X \Rightarrow Y$ tales que $fr(X \cup Y) > \sigma$ y $conf(X \Rightarrow Y) \geq \delta$."*

1.3 Árboles de decisión

En este capítulo introducimos algunas nociones básicas sobre árboles de decisión [Quinlan, 1986], [Quinlan, 1986a].

Sea $A = \{X_1, \ldots, X_n, Y\}$ un conjunto de atributos donde el atributo Y es un atributo especial llamado "clase".

Cada atributo X_i toma los valores en $D_i = \{1, 2, \ldots, v(i)\}$ con $i = 1, \ldots, n$, donde v es una función $v : \{1, \ldots, n\} \to \mathbb{N}$. El atributo Y toma valores en $D = \{1, \ldots, h\}$ y se dice entonces que hay h clases.

Definición 1.32 *Se define el universo de experiencias U_E como el producto cartesiano $U_E = D_1 \times \cdots \times D_n \times D$.*

Definición 1.33 *Sea $A = \{X_1, \ldots, X_n, Y\}$ un conjunto de atributos. Se define una experiencia $e \in U_E$ como el vector $e = (X_1(e), X_2(e), \ldots, X_n(e), Y(e))$ donde $X_i(e)$ representa el valor del atributo X_i en la experiencia e, con $i = 1, \ldots, n$ e $Y(e)$ es la clase a la que pertenece esa experiencia.*

Sea $\mathcal{T} = \{e_1, \ldots, e_N\}$ un conjunto finito no vacío de experiencias con el que trabajaremos.

Definimos ahora las observaciones, que tienen la misma estructura que las experiencias sólo que no tienen asignada una clase. Usaremos el árbol de decisión resultante después de aplicar los algoritmos de aprendizaje inductivos de árboles de decisión para predecir a qué clase pertenecen las observaciones.

Definición 1.34 *Se define el universo de observaciones U_B como el producto cartesiano $U_B = D_1 \times \cdots \times D_n$.*

Definición 1.35 *Sea $R = \{X_1, \ldots, X_n\}$ un conjunto de atributos. Se define una observación $b \in U_B$ como el vector $b = (X_1(b), X_2(b), \ldots, X_n(b))$ donde $X_i(b)$ representa el valor del atributo X_i en la observación b.*

Las siguientes definiciones formalizan a qué llamamos de forma genérica una predicción y un sistema de predicción. Ambos conceptos son necesarios para entender cómo usaremos el árbol de decisión obtenido para predecir la clase de las observaciones.

Definición 1.36 *Se define el universo de predicciones y se representa por U_P como el conjunto*

$$U_P = \{(p_1, \ldots, p_h) \in [0,1]^h \mid \sum_{k=1}^{h} p_k = 1\}$$

Definición 1.37 *Se define un sistema de predicción P como la función $P : U_B \to U_P$ donde $P(b) = (P_1(b), \ldots, P_h(b)) \in U_P$ siendo $P_k(b)$ la predicción de que la observación b pertenezca a la clase k con $k = 1, \ldots, h$.*

Un árbol de decisión es un árbol (grafo conexo y sin ciclos) con determinadas características:

- Cada vértice interno (incluida la raíz) del árbol está etiquetado con un atributo del conjunto $\{X_1, \ldots, X_n\}$ y un conjunto de experiencias.
- Cada hoja del árbol tiene asociado un conjunto de experiencias.
- En un camino desde la raíz hasta cualquier hoja, cada atributo aparece a lo sumo una vez.
- Cada vértice etiquetado con un atributo tiene tantos hijos como valores haya dado el test para el atributo.

El proceso de construcción de un árbol de decisión a partir de un conjunto finito de experiencias $\mathcal{T}$ se resume básicamente así:

1. Etiquetar el vértice interno (en el primer paso será la raíz):

   Para cada atributo X_i con $i = 1, \ldots, n$ se elige un test X^i que realiza una partición en los valores del atributo. Esta partición se denota por $\{\theta_j, j = 1, \ldots, u(i) \quad / \quad u(i) \leq v(i); \quad u : \{1, \ldots, n\} \to \mathbb{N}\}$. Los conjuntos θ_j se llamarán también valores. (La versión básica del algoritmo [Quinlan, 1986, Quinlan, 1986a] supone siempre $u(i) = v(i)$ para la partición).

2. Elección de un atributo mediante algún criterio de división para expandir el árbol de decisión

   Supongamos que X_i es el atributo elegido para etiquetar el vértice.

3. Crear hijos y repartir experiencias

Se crean tantos hijos como valores de X_i haya dado el test X^i.

A cada hijo se le asocia el subconjunto correspondiente de la partición $\{T_1, T_2, \ldots, T_{u(i)}\}$ siendo $T_j = \{e \in \mathcal{T} : X_i(e) \in \theta_j\}$ con $j = 1, \ldots, u(i)$.

4. Si existe un hijo que cumpla que todas las experiencias son de la misma clase, o un porcentaje de ellas es de una clase, o hay un número determinado de experiencias o cualquier otra condición hoja que se defina, el árbol no se expandirá por ese vértice. Para los demás vértices que no cumplan la condición hoja se repiten los pasos del 1 al 3 hasta que cumplan la condición hoja.

Definimos ahora el criterio de división para elegir el atributo que etiqueta al vértice.

Para simplificar la notación denotamos por T al conjunto de experiencias en el nodo actual del proceso de construcción. Para el atributo X_i con $i = 1, \ldots, n$, sea $\{\theta_j, j = 1, \ldots, u(i)\}$ la partición de los valores del atributo X_i dada por X^i. Sea $\{T_1, T_2, \ldots, T_{u(i)}\}$ la partición del conjunto de experiencias siendo $T_j = \{e \in T / X_i(e) = \theta_j\}$ con $j = 1, \ldots, u(i)$.

Definición 1.38 *Sea X^i el test asociado con el atributo X_i con $i = 1, \ldots, n$. Dada una función medida $m : \mathcal{P}(\mathcal{T}) \to \mathbb{R}^+$ donde $\mathcal{P}(\mathcal{T})$ es el conjunto partes de $\mathcal{T}$, se define la función $m' : R \times \mathcal{P}(\mathcal{T}) \to \mathbb{R}^+$ donde*

$$m'(X_i, T) = \sum_{j=1}^{u(i)} \frac{\mid T_j \mid}{\mid T \mid} \cdot m(T_j) \quad \textit{para todo } i = 1, \ldots, n$$

y $T = T_1 \cup T_2 \cup \cdots \cup T_{u(i)}$.

Definición 1.39 *Sea X^i el test asociado con el atributo X_i con $i = 1, \ldots, n$. Se define la función gradiente $\Delta : R \times \mathcal{P}(\mathcal{T}) \to \mathbb{R}^+$ donde*

$$\Delta(X_i, T) = m(T) - m'(X_i, T) \quad \textit{para todo } i = 1, \ldots, n$$

Finalmente, el criterio de división sigue la idea de [Quinlan, 1992].

Definición 1.40 *Sea X^i el test asociado con el atributo X_i con $i = 1, \ldots, n$. Se define el criterio de división y se denota por Δ' como:*

$$\Delta'(X_i, T) = \Delta(X_i, T)/m'(X_i, T) \quad \textit{para todo } i = 1, \ldots, n$$

.

Se elige como atributo para etiquetar el nodo actual el atributo X_k que sea máximo en $\{\Delta'(X_i, T) : X_i \in R\}$.

El proceso de construcción de un árbol de decisión intenta revelar la estructura del dominio y por tanto tiene poder de predicción. Para alcanzar esto, se necesita en cada hoja un número significativo de casos; en otras palabras, la partición debe tener el menor número de bloques posibles. El problema de encontrar el árbol de decisión más pequeño consistente a partir de un conjunto de experiencias es NP-completo [Hyalf y Rivest, 1976]. Por lo tanto, la solución para cada problema depende en gran medida de la elección de tests apropiados y de la elección de criterios de división apropiados.

1.4 Gramáticas y Autómatas

Se desarrollan ahora los conceptos básicos necesarios para la inferencia de gramáticas.

Pueden encontrarse desarrollados con más detalle para los autómatas finitos deterministas y las gramáticas regulares en [Hopcroft y Ullman, 1979], para las gramáticas regulares difusas en [Honda et al., 1977, Santos, 1977] y para las relaciones difusas en [Klir y Folger, 1988].

Definición 1.41 *Sea Σ un conjunto no vacío al que llamamos* alfabeto,

- *Sea* $n \in \mathbb{N} - \{0\}$*; una* cadena *sobre el alfabeto* Σ *es toda secuencia finita* $w : \{1, \ldots, n\} \to \Sigma$*. El natural* n *se llama* longitud *de* w*. Las cadenas (también llamadas* palabras*) se denotan por yuxtaposición de sus elementos, es decir,* $w_1 w_2 \ldots w_n$*.*

- *Para cada conjunto* Σ*, denotamos por* ε*, a la función* $\varepsilon : \emptyset \to \Sigma$ *y la denominamos* cadena vacía*. Para cada conjunto* Σ *denotamos por* Σ^0 *al conjunto* $\Sigma^0 = \{\varepsilon\}$*.*

- *Para todo* $n \in \mathbb{N}$*, sea* Σ^n *el conjunto de las cadenas de longitud* n *sobre* Σ*. La unión de la familia* $\{\Sigma^n\}_{n\in\mathbb{N}}$ *se llama* lenguaje universal *sobre* Σ *y se denota por* Σ^**.*

- *Un* lenguaje *es cualquier subconjunto de* Σ^* *y se denota por* $\mathcal{L}$*.*

Gramáticas regulares

Definición 1.42 *Una* gramática regular (por la derecha) *es un vector de cuatro componentes que se denota por* $\mathcal{G} = (V, T, S, r)$ *donde*

- $V = \{A_0 = S, A_1, A_2, \ldots, A_n\}$ *es un conjunto finito no vacío a cuyos elementos se les llama* no terminales*.*

- $T = \{a_1, a_2, \ldots, a_m\}$ *es un conjunto finito no vacío cuyos elementos se les llama* terminales*;* $V \cap T \neq \emptyset$*.*

- $S \in V$ *es el único* símbolo inicial *y*

- r *es un conjunto finito de* reglas de derivación *de la siguiente forma* $A_i \to a_k A_j$ *y* $A_i \to a_k$ *donde* $A_i \in V$*,* $a_k \in T \cup \{\varepsilon\}$*. Al lado izquierdo de la regla se le llama* antecedente *y el lado derecho* consecuente*. Cada regla la representamos por* $r_{i,h}$*, donde el primer índice corresponde con el índice del no terminal del antecedente y el segundo índice corresponde a la posición de la regla en el grupo de reglas con el mismo antecedente.*

Lenguaje generado por una gramática regular

Definición 1.43 (derivación en un paso) *La cadena* $w_2 = x\, a_k\, A_j$ *se deriva desde la* $w_1 = xA_i$ *y se representa por* $d : w_1 \Rightarrow_G w_2$ *si existe una regla de derivación en la gramática de la forma* $A_i \rightarrow a_k A_j y$ *a que al aplicarla a la cadena* w_1 *se obtiene la cadena* w_2.

Definición 1.44 (derivación en n pasos) *Una* derivación en n pasos *es una secuencia finita de derivaciones en un sólo paso donde se deriva la cadena* w_n *a partir de la cadena* w_1. *La derivación en n pasos se representa por* $d : w_1 \Rightarrow_G^n w_n$. *La derivación en cualquier número de pasos se denota por* $\Rightarrow_G^*$.

A las reglas de la forma $A_i \rightarrow a_k$ donde $A_i \in V$, $a_k \in T \cup \{\varepsilon\}$ se les llama reglas terminales porque en ellas termina el proceso de derivación de una cadena.

Definición 1.45 *El* lenguaje generado por una gramática $\mathcal{G}$ *es el conjunto de todas las cadenas formadas por terminales que se pueden obtener a partir de las reglas de derivación de la gramática* $\mathcal{G}$ *y se representa por* $\mathcal{L}(\mathcal{G})$ *o simplemente por* $\mathcal{L}$.

$$\mathcal{L}(\mathcal{G}) = \{w \in T^* / S \Rightarrow_G^* w\}$$

Gramáticas regulares probabilísticas

Definición 1.46 *Una* gramática regular probabilística *denotada por* $\mathcal{G}_P$ *es un par* $< \mathcal{G}, P >$ *donde* $\mathcal{G}$ *es una gramática regular por la derecha y* $p_{i,h} \in P$ *es un conjunto ordenado de probabilidades que se asignan a las reglas* $r_{i,h} \in r$ *de la gramática* $\mathcal{G}$. *En lo que sigue denotaremos* $p(r_{i,h}) = p_{i,h}$.

Definición 1.47 *Una gramática regular probabilística* $\mathcal{G}_P$ *se dice que es* propia *si*

$$\textit{para todo } i \textit{ con } 0 \leq i \leq |V|, \sum_{\{h|P_{i,h} \in r\}} p_{i,h} = 1$$

La probabilidad de una derivación d se calcula como el producto de las probabilidades de las reglas usadas en la derivación de la cadena, es decir, $p(d) = \prod_{1 \leq k \leq n} p(r_k)$ donde $p(r_k)$ es la probabilidad de la derivación k–ésima, $1 \leq k \leq n$. La probabilidad de una cadena viene dada por la suma de las probabilidades de todas sus derivaciones en la gramática, $p(w) = \sum_{d \in \mathcal{G}} p(d)$, $w \in \mathcal{L}(\mathcal{G})$.

Autómatas finitos deterministas

Definición 1.48 *Un* autómata finito determinista *es un vector de la forma* $M = (K, \Sigma, \delta, q_o, F)$ *donde:*

- K *es un conjunto finito no vacío a cuyos elementos se les llama* estados
- Σ *es el alfabeto* $K \cap \Sigma = \emptyset$
- $\delta : (K - F) \times \Sigma \to K$ *es la* función de transición
- q_o *es el* estado inicial, $q_o \in K$
- F *es el conjunto de* estados finales, $F \subseteq K$

Denotamos por $\mathcal{L}(M) = \mathcal{L}$ al lenguaje regular aceptado por M.

Damos la definición de grafo y grafo dirigido para introducir el diagrama de transición de un autómata finito determinista.

Un *grafo*, denotado por $G = (V, E)$, consiste en un conjunto finito de *vértices* (o nodos) V y un conjunto de pares de vértices E llamados aristas.

Un *grafo dirigido* (o *digrafo*), que también se denota por $G = (V, E)$, consiste en un conjunto de vértices V y un conjunto de pares ordenados de vértices E, llamados *arcos*. Denotamos un arco de v a w como $v \to w$.

Un autómata finito determinista se puede representar por su diagrama de transición entre estados que es un grafo dirigido donde los vértices del

grafo representan a los estados del autómata y los arcos se etiquetan con los símbolos del alfabeto.

Autómatas finitos probabilistas

Definición 1.49 *Un* autómata finito probabilista *es un vector de la forma* $N = (K, \Sigma, \delta, \gamma, \pi, F)$ *donde:*

- K *es un conjunto finito no vacío a cuyos elementos se les llama* estados
- Σ *es el alfabeto* $K \cap \Sigma = \emptyset$
- $\delta : (K - F) \times \Sigma \to K$ *es la* función de transición
- $\gamma : (K - F) \times \Sigma \to [0, 1]$ *es la función de probabilidad del próximo símbolo*
- $\pi : (K - F) \to [0, 1]$ *es la distribución de probabilidad de los estados iniciales*
- F *es el conjunto de* estados finales, $F \subseteq K$

Además se cumple que para todo $q \in K - F$, $\sum_{\sigma \in \Sigma} \gamma(q, \sigma) = 1$ *y* $\sum_{q \in K} \pi(q) = 1$

Existe una equivalencia entre los gramáticas regulares y los autómatas finitos deterministas y las gramáticas regulares probabilísticas y los autómatas finitos probabilistas [Hopcroft y Ullman, 1979].

Gramáticas regulares difusas

Sea X un conjunto llamado conjunto universal. La pertenencia de un elemento a al subconjunto A de X es vista como una función característica

μ_A definida desde X a $\{0,1\}$ tal que

$$\mu_A(x) = \begin{cases} 1 & \text{si} \quad x \in A \\ 0 & \text{si} \quad x \notin A \end{cases}$$

A $\mu_A(x)$ se le llama grado de pertenencia del elemento x al conjunto A. El conjunto A se llama *conjunto difuso* si la imagen de la función de pertenencia μ_A se define en el intervalo $[0,1]$.

El valor 1 representa que el elemento pertenece nítidamente al conjunto A, el valor 0, representa la no pertenencia absoluta al conjunto A, y los demás valores indican una pertenencia parcial al conjunto A.

Definición 1.50 (de relaciones difusas) *Dados dos conjuntos X e Y. Sea el conjunto universal $X \times Y$, se define una* relación difusa *como un subconjunto de $X \times Y$ que tiene asociado una función de pertenencia.*

En particular se pueden considerar relaciones binarias difusas sobre un mismo conjunto (relación binaria que se establece sobre el producto cartesiano de dos conjuntos donde ambos coinciden).

Definición 1.51 *Una* gramática regular difusa *denotada por $\mathcal{G}_\mu$ es un par $< \mathcal{G}, \mu >$ donde $\mathcal{G}$ es una gramática regular y μ es una función de pertenencia que asocia grados a las reglas de la gramática. La expresión de las reglas es de la forma $\mu_p(t) : A_i \to a_k A_j, \mu_p(r) : A_i \to a_k$ donde μ_p es el grado de pertenencia.*

Definición 1.52 (Grado de pertenencia de una secuencia en $\mathcal{G}$) *Sea α_i con $i = 1, \ldots, m$, y x secuencias de $(V \cup T)^*$. Sea i una norma triangular y u su co-norma, entonces el grado de pertenencia de una secuencia es*

$$\mu_{\mathcal{G}}(x) = u(i(\mu(S \to \alpha_1), \mu(\alpha_1 \to \alpha_2), \ldots, \mu(\alpha_m \to x)))$$

Capítulo 2

Información negativa frecuente

En este capítulo desarrollamos un método para la búsqueda de conjuntos frecuentes donde la aparición de atributos negativos también es importante. El problema de la búsqueda de las reglas de asociación se encuadra dentro del aprendizaje no supervisado porque los datos de los que disponemos no están clasificados de ninguna forma.

Formalizamos el problema de la búsqueda de información negativa frecuente, definiendo un lenguaje, una relación de especialización y un predicado adecuados para este caso. Definimos la representación del conjunto de itemsets que dará la clave del orden de búsqueda de los itemsets frecuentes. Desarrollamos unas proposiciones que nos llevarán a los algoritmos para la búsqueda de itemsets frecuentes con atributos positivos y negativos.

La búsqueda de los itemsets frecuentes se organiza dando prioridad a los atributos positivos frente a los negativos para que no se obtenga una cantidad muy grande de información irrelevante. Para ello también se presentan algoritmos que sólo buscan itemsets frecuentes con un número acotado de atributos negativos.

Los algoritmos desarrollados son una variante del Apriori porque siguen siendo algoritmos del tipo bottom–up pero son más eficientes que la aplicación directa de Apriori en nuestro problema. En nuestros algoritmos se evita la evaluación explícita de los itemsets en la base de datos siempre que sea posible. Esto se consigue a partir de unas relaciones que damos sobre el cálculo de la frecuencia de un itemset a partir de las frecuencias de determinados itemsets. Con esto se minimiza el número de itemsets a evaluar en cada pasada por la base de datos. A estos nuevos algoritmos los hemos denominado Neg–Apriori y Acot–Neg–Apriori.

2.1 Planteamiento teórico

Sea $\mathcal{T}$ una base de datos, $\mathcal{L}$ el lenguaje formado por todos los subconjuntos de datos de $\mathcal{T}$ y q un predicado que evalúa si una palabra $\varphi \in \mathcal{L}$ es interesante o no. Queremos calcular el conjunto de todas las palabras interesantes de $\mathcal{L}$ sobre $\mathcal{T}$, es decir, el conjunto de palabras tales que $q(\varphi)$ sea cierto. A este conjunto lo llamamos Teoría de $\mathcal{L}$, $\mathcal{T}$ y q y lo denotamos por $Th(\mathcal{L}, \mathcal{T}, q)$.

Dada esta descripción general, existen dos subtareas a desarrollar:

1. Elaborar un método para descubrir palabras φ candidatas a ser interesantes y

2. comprobar si estas palabras φ son realmente interesantes.

Dada una relación de especialización $\preccurlyeq$ en $\mathcal{L}$, el conjunto $Th(\mathcal{L}, \mathcal{T}, q)$ se puede representar enumerando sólo sus elementos maximales, es decir, a partir del conjunto

$$MTh(\mathcal{L}, \mathcal{T}, q) = \{\varphi \in Th(\mathcal{L}, \mathcal{T}, q) \mid \forall\, \theta \in Th(\mathcal{L}, \mathcal{T}, q)\,, \varphi \preccurlyeq \theta \Rightarrow \varphi = \theta\}$$

se pueden obtener todos los elementos del conjunto $Th(\mathcal{L}, \mathcal{T}, q)$.

Obviamente, si $\mathcal{L}$ es infinito y $q(\varphi)$ se satisface para infinitas palabras, no se puede calcular una representación explícita de $Th(\mathcal{L}, \mathcal{T}, q)$.

Por lo tanto el problema de clasificar las palabras en interesantes o no, se reduce a calcular los elementos maximales de $Th(\mathcal{L}, \mathcal{T}, q)$. La complejidad de cálculo que requiere esta tarea se demuestra en un teorema en [Mannila y Toivonen, 1997] y el resultado se basa en el concepto de frontera.

Definición 2.1 *Sea S un conjunto de palabras de $\mathcal{L}$ cerrado bajo la relación $\preccurlyeq$. Se define la frontera positiva como*

$$Bd^+(S) = \{\varphi \in \mathcal{S} \wedge q(\varphi) \,/\, \forall\, \theta \in \mathcal{L},\ \varphi \preccurlyeq \theta \Rightarrow \neg q(\theta)\}$$

Paralelamente se define la frontera negativa

$$Bd^-(S) = \{\varphi \in (\mathcal{L} - S) \wedge \neg q(\varphi) \,/\, \forall \theta \in \mathcal{L}, \theta \preccurlyeq \varphi \Rightarrow q(\theta)\}$$

y la frontera del conjunto S de palabras

$$Bd(S) = Bd^+(S) \cup Bd^-(S)$$

En [Mannila y Toivonen, 1997] se desarrolla el algoritmo levelwise para calcular $Th(\mathcal{L}, \mathcal{T}, q)$, y se demuestra que cualquier algoritmo que realice preguntas a un oráculo para ver si $q(\varphi)$ es interesante realizará al menos $|Bd(Th(\mathcal{L}, \mathcal{T}, q))|$ preguntas para calcular $Th(\mathcal{L}, \mathcal{T}, q)$.

Además, también se demuestra que el problema de encontrar conjuntos frecuentes es un ejemplo del problema de encontrar $MTh(\mathcal{L}, \mathcal{T}, q)$ para un lenguaje y un predicado adecuados. En este caso, el predicado "ser interesante" se interpreta como "ser frecuente".

El algoritmo levelwise y los algoritmos de la familia de Apriori son esencialmente el mismo a la hora de organizar la búsqueda de palabras interesantes y conjuntos frecuentes respectivamente. Con respecto a la complejidad, desde el punto de vista práctico, las preguntas al oráculo se

sustituyen por pasadas a la base de datos. En nuestro algoritmo seguimos la filosofía del algoritmo Apriori organizando la búsqueda de conjuntos frecuentes de primero en amplitud y con recuento. Por lo tanto, aunque la complejidad del algoritmo levelwise y del algoritmo Apriori está en función de parámetros diferentes se puede demostrar [Baixeries et al., 2000] que la complejidad es realmente la misma.

Vamos ahora a trasladar esta formalización a nuestro caso de búsqueda de información positiva y negativa. Primero, definimos un lenguaje con una relación de especialización y un predicado adecuados para formalizar la búsqueda de conjuntos frecuentes. A partir de ahí, mostramos que la teoría de las fronteras del conocimiento se puede trasladar fácilmente a la búsqueda de reglas de asociación en nuestro caso. La relación de especialización que definamos nos será útil más adelante para organizar la búsqueda de nuestros conjuntos frecuentes.

2.1.1 Conjunto de atributos: Itemset

En esta sección, vamos a establecer un nueva definición de itemset que generaliza el concepto de conjunto de atibutos. Nuestra intención es poder distinguir dos tipos de atributos en el conjunto.

En adelante, sea $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$ una base de datos con N filas o transacciones sobre un conjunto $R = \{A_i : i \in I\}$ de atributos que toman valores en el dominio $\{0, 1\}$ y sea $I = \{1, 2, \ldots, n\}$ el conjunto de índices.

Para cada subconjunto p de I denotamos por A^p al conjunto de atributos de R cuyos índices están en p, es decir, para cada $p \in \mathcal{P}(I)$

$$A^p = \{A_i \in R \,/\, i \in p\}$$

Cada transacción $t \in \mathcal{T}$ de la base de datos tiene un único identificador y asigna un 0 o un 1 a cada elemento de R. Por tanto, las transacciones se pueden considerar funciones sobre el conjunto de atributos e identificar los subconjuntos de atributos como funciones parciales.

El objetivo es encontrar en la base de datos las funciones parciales más frecuentes y para ello identificamos cada evaluación de estas funciones parciales con los conjuntos de atributos que se definen a continuación.

Definición 2.2 *Para cada* $p \in \mathcal{P}(I)$ *y* $s \in \mathcal{P}(I-p)$ *llamamos* itemset *al conjunto de atributos* $A^{p,s}$ *que se identifica con la función parcial que evalúa cada atributo del conjunto* A^p *a* 1*, cada atributo del conjunto* A^s *a* 0 *e indefinido para el resto de atributos de* R*.*

Si $A^{p,s}$ es un itemset, entonces los atributos de A^p se denominan *atributos positivos* y los atributos de A^s se denominan *atributos negativos.* Además, a los itemsets de cardinal ℓ los llamamos ℓ-itemsets y si $|s| = f$ el conjunto $A^{p,s}$ recibe el nombre de *itemset* f*-negativos.* La igualdad de itemsets queda determinada por la igualdad de los conjuntos correspondientes de atributos positivos y negativos.

De esta forma, cada transacción $t \in \mathcal{T}$ se puede considerar como un itemset donde para cada $A \in R$ diremos que $A \in t$ si $t(A) = 1$ o bien diremos que $\overline{A} \in t$ si $t(A) = 0$. Es decir, identificamos la transacción t de la base de datos con el itemset $A^{p,s}$ tal que $t(A) = 1$ para todo $A \in A^p$, $t(A) = 0$ para todo $A \in A^s$ y $p \cup s = I$.

Obsérvese que el caso estudiado hasta ahora donde todos los atributos de los itemsets eran positivos es un caso particular del nuestro para $s = \emptyset$.

Definición 2.3 *Sea* $R = \{A_i : i \in I\}$ *con* $I = \{1, \ldots, n\}$ *un conjunto de atributos. Llamamos* $\mathcal{I}$ *al conjunto formado por todos los itemsets, es decir,*

$$\mathcal{I} = \{A^{p,s} \,/\, p \in \mathcal{P}(I),\ s \in \mathcal{P}(I-p)\}$$

Ejemplo 2.1 Sea $\mathcal{T}$ una base de datos sobre el conjunto de atributos $R = \{A_1, A_2, A_3, A_4\}$. Sean $p = \{1, 2\}$ y $s = \{4\}$ dos subconjuntos de índices. Llamamos $A^{p,s}$ y lo denotamos por $\{A_1, A_2, \overline{A}_4\}$ al subconjunto de atributos que se identifica con la función parcial sobre R que asigna el valor 1 a los atributos (positivos) del conjunto $A^p = \{A_1, A_2\}$ y 0 al único atributo (negativo) del conjunto $A^s = \{A_4\}$. ∎

Definimos ahora el lenguaje cuyas palabras representarán a los itemsets.

Definición 2.4 *Consideramos el alfabeto* $\Sigma = \{A_1, \overline{A}_1, \ldots, A_n, \overline{A}_n\}$ *con los símbolos ordenados según una relación que denotamos por* $<$ *de esta forma* $A_1 < \overline{A}_1 < A_2 < \overline{A}_2 < \cdots < A_n < \overline{A}_n$. *Definimos en* Σ^* *el lenguaje* $\mathcal{L}$ *formado por las palabras* $w = a_1 \ldots a_k$ *que cumplan estas condiciones:*

1. $a_i \in \Sigma$ *para todo* $i = 1, \ldots, k$.
2. $a_i < a_{i+1}$ *para todo* $i = 1, \ldots, k-1$.
3. *Si* $A_j \in w \Rightarrow \overline{A}_j \notin w$ *para cada* $j = 1, \ldots, n$.

De la definición anterior se desprende que la longitud máxima de cualquier palabra del lenguaje $\mathcal{L}$ es n y además, la igualdad de palabras (símbolo a símbolo) se reduce a la igualdad de los conjuntos de símbolos del alfabeto que las componen.

Proposición 2.1 *Sean* v *y* w *dos palabras del lenguaje* $\mathcal{L}$ *y consideramos los conjuntos de símbolos de estas palabras:* $V = \{a \in \Sigma \,/\, a \in v\}$ *y* $W = \{b \in \Sigma \,/\, b \in w\}$. *Entonces* $v = w$ *si y sólo si* $V = W$

Demostración: Si $v = w$ entonces la igualdad símbolo a símbolo determina que $V = W$. El recíproco se demuestra aplicando la condición de ordenación de los símbolos en una palabra. □

Ahora vamos establecer una biyección entre el lenguaje $\mathcal{L}$ y el conjunto $\mathcal{I}$ formado por todos los itemsets de tal manera que cada itemset $A^{p,s}$ corresponda a una palabra del lenguaje $w = a_1 a_2 \ldots a_\ell$ donde $\ell = |p \cup s|$ y cada a_i de w es el símbolo A_j o bien el símbolo $\overline{A}_j$ para algún $j \in I$. Por lo tanto, en el ejemplo 2.1, el conjunto $\{A_1, A_2, \overline{A}_4\}$ se puede identificar con la palabra $A_1 A_2 \overline{A}_4$ del lenguaje.

Teorema 2.1 *La aplicación* $b : \mathcal{L} \to \mathcal{I}$ *que a cada elemento* $w \in \mathcal{L}$ *le asigna el itemset* $A^{p,s} \in \mathcal{I}$ *siendo* p *el conjunto de índices* $\{i \in I \,/\, A_i \in w\}$ *y* s *el conjunto de índices* $\{j \in I \,/\, \overline{A}_j \in w\}$, *es una biyección.*

Demostración: Veamos que la aplicación es inyectiva, es decir, que si $b(w_1) = b(w_2)$ entonces $w_1 = w_2$.

Para ello, sean $b(w_1) = A^{p,s}$ y $b(w_2) = A^{q,t}$ dos itemsets. La igualdad de itemset determina que $p = q$ y $s = t$ y como

$$\begin{array}{rcl} p = q & \Rightarrow & \{i \in I \,/\, A_i \in w_1\} = \{i \in I \,/\, A_i \in w_2\} \quad \Rightarrow \\ & \Rightarrow & \{A \in \Sigma \,/\, A \in w_1\} = \{A \in \Sigma \,/\, A \in w_2\}, \\ s = t & \Rightarrow & \{j \in I \,/\, \overline{A}_j \in w_1\} = \{j \in I \,/\, \overline{A}_j \in w_2\} \quad \Rightarrow \\ & \Rightarrow & \{\overline{A} \in \Sigma \,/\, \overline{A} \in w_1\} = \{\overline{A} \in \Sigma \,/\, \overline{A} \in w_2\}, \end{array}$$

entonces, $w_1 = w_2$ por la proposición 2.1 sobre la igualdad de palabras en un lenguaje.

Probemos ahora la sobreyectividad viendo que para todo $A^{p,s} \in \mathcal{I}$ existe un $w \in \mathcal{L}$ tal que $f(w) = A^{p,s}$. Sea $A^{p,s}$ un itemset con $|p \cup s| = \ell$ y consideramos el conjunto de símbolos del alfabeto Σ dado por

$$W = \{A_i \in \Sigma \,/\, i \in p\} \cup \{\overline{A}_j \in \Sigma \,/\, j \in s\}.$$

Consideramos la palabra $w \in \mathcal{L}$ que verifica:

$$\begin{array}{l} \text{que } w = a_1 a_2 \ldots a_\ell \text{ con } a_i \in W \text{ para todo } i = 1, \ldots \ell, \text{ y} \\ \text{que } a_i < a_{i+1} \text{ para todo } i = 1, \ldots, \ell - 1. \end{array}$$

y por lo tanto, $f(w) = A^{p,s}$. □

Este teorema es fundamental para identificar nuestro problema con el planteamiento teórico y poder estudiar su complejidad.

2.1.2 Frecuencia de un itemset

Para definir la frecuencia de un itemset necesitamos definir una relación de especialización entre los itemsets del conjunto $\mathcal{I}$.

Definición 2.5 *En el conjunto $\mathcal{I}$ definimos la relación de orden parcial $\preccurlyeq$ de la siguiente manera: si $X = A^{p,s}$ e $Y = A^{q,t}$ son dos itemsets de $\mathcal{I}$,*

decimos que $X \preccurlyeq Y$ *si y sólo si* $p \subseteq q$ *y* $s \subseteq t$. *En tal caso, diremos que* X *es un subconjunto de* Y.

Una transacción de la base de datos se puede ver como una función total sobre el conjunto de atributos y un itemset o subconjunto de estos atributos puede ser visto como una función parcial. Si estas funciones parciales se pueden extender a alguna función total correspondiente a una transacción entonces decimos que los itemsets son subconjuntos de las transacciones y empleamos el símbolo $\preccurlyeq$ para representar dicha relación.

Obsérvese también que cada transacción se pueden ver como un elemento de $\mathcal{I}$, identificando cada valor igual a 0 de la transacción con su correspondiente atributo negativo e identificando cada valor igual a 1 con su correspondiente atributo positivo.

Definición 2.6 *Sea* $R = \{A_i : i \in I\}$ *un conjunto finito de atributos y sea* $\mathcal{T} = \{t_1, t_2 \ldots, t_N\}$ *el conjunto de transacciones de una base de datos. Para cada itemset* $X \in \mathcal{I}$ *definimos el conjunto* T_X *de todas las transacciones que contiene a* X, *es decir,*

$$T_X = \{t \in \mathcal{T} \,/\, X \preccurlyeq t\} \qquad \textit{para todo} \quad X \in \mathcal{I}$$

Ahora estamos en condiciones de extender la definición de frecuencia para estos nuevos itemsets que consideran atributos negativos.

Definición 2.7 *Sea* $R = \{A_i : i \in I\}$ *un conjunto finito de atributos y sea* $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$ *el conjunto de transacciones de una base de datos. Se define el soporte o frecuencia de un itemset* X *y se denota por* $fr(X)$ *como la proporción del número de transacciones que incluyen a* X *como subconjunto, es decir,*

$$fr(X) = \frac{|T_X|}{N} \qquad \textit{para todo} \quad X \in \mathcal{I}$$

Proposición 2.2 *La definición de frecuencia para los itemsets verifica las siguientes propiedades:*

1. $fr(X) \in [0,1]$ *para todo* $X \in \mathcal{I}$.

2. $fr(\emptyset) = fr(A^{\emptyset,\emptyset}) = 1$.

Demostración: Para demostrar la propiedad *1* debemos tener en cuenta que $T_X \subseteq \mathcal{T}$, con lo cual $0 \leq |T_X| \leq |\mathcal{T}| = N$ y, por lo tanto, $fr(X) \in [0,1]$. La propiedad *2* es una consecuencia de que $T_\emptyset = \{t \in \mathcal{T} : \emptyset \preccurlyeq t\} = \mathcal{T}$ y, por tanto, $fr(\emptyset) = N/N = 1$. □

Ejemplo 2.2 Sea $R = \{A, B, C, D\}$ el conjunto de atributos correspondiente a la base de datos $\mathcal{T}$ cuyas transacciones aparecen en la tabla de la figura 2.1. Para calcular la frecuencia del itemset $X = A\,\overline{B}\,\overline{C}$, recorremos la

	A	B	C	D
t_1	1	0	0	0
t_2	1	1	0	0
t_3	0	1	0	1
t_4	1	0	0	1
t_5	1	1	1	0
t_6	1	0	1	0
t_7	1	1	1	0
t_8	0	1	1	0
t_9	1	0	0	0
t_{10}	1	1	0	0

Figura 2.1: Base de datos del ejemplo 2.2

base de datos contando todas las transacciones que contengan a X, es decir, buscamos transacciones que asignen 1 al atributo A y 0 a los atributos B y C. En este caso, el conjunto de estas transacciones es $\{t_1, t_4, t_9\}$ y aplicando la fórmula de la frecuencia obtenemos

$$fr(X) = \frac{|\{t_1, t_4, t_9\}|}{10} = \frac{3}{10} = 0,3 \qquad \blacksquare$$

Veamos que la frecuencia definida sobre el conjunto de itemsets $\mathcal{I}$ se puede considerar como una función de probabilidad.

Si definimos los conjuntos $A_i = \{t \in \mathcal{T} \,/\, t(A_i) = 1\}$ y $\overline{A}_j = \{t \in \mathcal{T} \,/\, t(A_j) = 0\}$, entonces para cada $p \in \mathcal{P}(I)$ y $s \in \mathcal{P}(I-p)$ podemos definir $A^{p,s} = \left(\bigcap_{i\in p} A_i\right) \bigcap \left(\bigcap_{j\in s} \overline{A}_j\right)$ que identificaremos con el correspondiente itemset $A^{p,s}$ de $\mathcal{I}$.

Consideramos el espacio muestral $\Omega = \{A^{p,s} : p \cup s = I\}$ y sea $\mathcal{A} = \mathcal{P}(\Omega)$ un álgebra de Boole. Sobre el espacio probabilizable $(\Omega, \mathcal{A})$ definimos la función de probabilidad $P : \mathcal{A} \to \mathbb{R}$ como

$$P(X) = \frac{|X|}{N} \qquad \text{para todo } X \in \mathcal{A}$$

que corresponde con la frecuencia del itemset X de $\mathcal{I}$.

Definición 2.8 *Sea $\mathcal{T}$ una base de datos sobre un conjunto R de atributos y sea σ un valor mínimo del soporte especificado por el usuario. Decimos que un itemset X es* σ-frecuente *o simplemente* frecuente *si su soporte es mayor que el soporte mínimo, es decir, si $fr(X) > \sigma$. Denotamos por $Fr(\mathcal{T}, \sigma)$ al conjunto formado por todos los itemsets frecuentes que se encuentran en $\mathcal{T}$.*

Con respecto a la relación de especialización definida antes tenemos la propiedad de antimonotonía respecto a los conjuntos de atributos frecuentes.

Proposición 2.3 *Sea $\mathcal{T}$ una base de datos sobre un conjunto R de atributos y sean X e Y dos itemsets. Si $X \preccurlyeq Y$ entonces $fr(X) \geq fr(Y)$.*

Demostración: Por la transitividad de la relación, si $X \preccurlyeq Y$ entonces cualquier transacción $t \in \mathcal{T}$ que verifique la condición $Y \preccurlyeq t$ también verifica la condición $X \preccurlyeq t$. Con esto hemos comprobado que el cardinal del conjunto $\{t \in \mathcal{T} \,/\, X \preccurlyeq t\}$ es mayor que el cardinal del conjunto $\{t \in \mathcal{T} \,/\, X \preccurlyeq t\}$ y por tanto, $fr(X) \geq fr(Y)$. □

Teorema 2.2 *Sea $\mathcal{T}$ una base de datos sobre el conjunto $R = \{A_i : i \in I\}$ de atributos y sea $A^{p,s}$ un itemset. Si u es un conjunto de índices tal que $u \subseteq I - (p \cup s)$ entonces el conjunto $\{T_X \,/\, X \in I_u^{p,s}\}$ es una partición del conjunto de transacciones $T_{A^{p,s}}$.*

Demostración: Hay que comprobar que la intersección dos a dos es vacía, es decir, $T_X \cap T_Y = \emptyset$ si $X \neq Y$ y que la unión es el total, es decir, $\bigcup_{X \in I_u^{p,s}} T_X = T_{A^{p,s}}$.

Por un lado, si $X, Y \in I_u^{p,s}$ y $X \neq Y$ entonces existe $i \in u$ tal que $A_i \in X$ y $\overline{A}_i \in Y$ o bien $\overline{A}_i \in X$ y $A_i \in Y$. Sin pérdida de generalidad, supongamos que $A_i \in X$ y $\overline{A}_i \in Y$. Ahora $t(A_i) = 1$ para todo $t \in T_X$ y $t(A_i) = 0$ para todo $t \in T_Y$ y, por lo tanto, $T_X \cap T_Y = \emptyset$.

Por otro lado, para comprobar la igualdad $\bigcup_{X \in I_u^{p,s}} T_X = T_{A^{p,s}}$ vamos a demostrar que se verifica la doble inclusión.

$\subset$) Si $t \in \bigcup_{X \in I_u^{p,s}} T_X$ entonces existe un $X \in I_u^{p,s}$ tal que $t \in T_X$ y se verifica

$$\left.\begin{array}{lcl} t \in T_X & \stackrel{(1)}{\Longrightarrow} & X \preccurlyeq t \\ X \in I_u^{p,s} & \stackrel{(2)}{\Longrightarrow} & A^{p,s} \preccurlyeq X \end{array}\right\} \quad \Rightarrow \quad A^{p,s} \preccurlyeq t \quad \Rightarrow \quad t \in T_{A^{p,s}}.$$

donde (1) es una consecuencia de la definición del conjunto T_X y (2) se deduce del lema 2.1.

$\supset$) Sea $t \in T_{A^{p,s}}$. Si $q = \{i \in u \,/\, t(A_i) = 1\}$ y $t = \{j \in u \,/\, t(A_j) = 0\}$ entonces $X = A^{p \cup q, s \cup t} \preccurlyeq t$, es decir, $t \in T_X$. Y como $X \in I_u^{p,s}$ entonces $t \in \bigcup_{X \in I_u^{p,s}} T_X$. □

Proposición 2.5 *Sea $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$ una base de datos sobre el conjunto de atributos $R = \{A_i : i \in I\}$ y sea $A^{p,s}$ un itemset. Si u es un conjunto de índices tal que $u \subseteq I - (p \cup s)$ entonces*

$$fr(A^{p,s}) = \sum_{X \in I_u^{p,s}} fr(X) \quad \textit{para todo } A^{p,s} \in \mathcal{I}.$$

Demostración: Partiendo de la definición de frecuencia del itemset $A^{p,s}$

obtenemos

$$fr(A^{p,s}) = \frac{|T_{A^{p,s}}|}{N} \overset{(*)}{=} \frac{1}{N} \sum_{X \in I_u^{p,s}} |T_X| = \sum_{X \in I_u^{p,s}} \frac{|T_X|}{N} = \sum_{X \in I_u^{p,s}} fr(X)$$

aplicando en $(*)$ que, por el teorema 2.2, el conjunto $\{T_X \,/\, X \in I_u^{p,s}\}$ es una partición del conjunto de transacciones $T_{A^{p,s}}$. □

Ahora obtenemos un caso particular de descomposición de frecuencias utilizando itemsets con un único atributo añadido. En el algoritmo de búsqueda de conjuntos frecuentes, esta descomposición permitirá calcular las frecuencias de algunos itemsets sin necesidad de rastrear la base de datos.

Proposición 2.6 *Sea $\mathcal{T}$ una base de datos sobre el conjunto de atributos $R = \{A_i : i \in I\}$ y sean $p \in \mathcal{P}(I)$ y $s \in \mathcal{P}(I - p)$ dos subconjuntos de índices de I. Entonces para cada índice $i \in I - (p \cup s)$ se verifica que*

$$fr(A^{p,s}) = fr(A^{p \cup \{i\},s}) + fr(A^{p,s \cup \{i\}})$$

Demostración: Consideramos el conjunto $u = \{i\}$ y aplicamos la proposición 2.5. □

Por último, y como caso particular del teorema 2.2, veamos que el conjunto de todos los itemsets con los mismos atributos (sin distinguir su carácter positivo o negativo) constituye una partición en la base de datos y por tanto, la suma de sus frecuencias es 1.

Definición 2.10 *Sea $\mathcal{T}$ una base de datos sobre el conjunto de atributos $R = \{A_i : i \in I\}$. Para cada $u \in \mathcal{P}(I)$ con $|u| = \ell$ llamamos I_ℓ^u al conjunto $I_u^{\emptyset,\emptyset}$ de todos los itemsets con los mismos ℓ atributos correspondientes al conjunto u de índices, es decir,*

$$I_\ell^u = \{A^{p,s} \in \mathcal{I} \,/\, p \cup s = u,\ |u| = \ell\}.$$

Proposición 2.7 *Sea $\mathcal{T}$ una base de datos sobre el conjunto de atributos $R = \{A_i : i \in I\}$. Para cada $u \in \mathcal{P}(I)$ con $|u| = \ell$ se verifica que*

1. *el conjunto $\{T_X \,/\, X \in I_\ell^u\}$ es una partición de $\mathcal{T}$*
2. *y, además, $\sum_{X \in I_\ell^u} fr(X) = 1$.*

Demostración: Consideramos el itemset $A^{\emptyset,\emptyset}$ y el conjunto de índices $u = I$. En ese caso, la propiedad *1* se obtiene aplicando directamente el teorema 2.2 y la propiedad *2* se deduce de la proposición 2.5 sabiendo que $fr(A^{\emptyset,\emptyset}) = 1$. En ambos casos se tiene en cuenta que $I_u^{p,s} = I_\ell^u$. □

2.1.3 Reglas de asociación

Para definir las reglas de asociación formadas por estos nuevos itemsets con atributos positivos y negativos, necesitamos definir la unión de itemsets. Para las definiciones y proposiciones de esta sección vamos a considerar una base de datos $\mathcal{T}$ sobre un conjunto R de atributos y sea $\mathcal{I}$ el conjunto de itemsets.

Definición 2.11 *Sean $X = A^{p,s}$ e $Y = A^{q,t}$ dos itemsets que verifican que $p \cap t = \emptyset$ y $q \cap s = \emptyset$. Se define la unión $X \cup Y$ y la intersección $X \cap Y$ a partir de la unión e intersección de los atributos que los forman, es decir,*

$$X \cup Y = A^{p \cup q, s \cup t} \qquad y \qquad X \cap Y = A^{p \cap q, s \cap t}.$$

Ejemplo 2.3 Sea $\mathcal{T}$ una base de datos sobre el conjunto de atributos $R = \{A, B, C, D\}$. Para calcular la unión e intersección de dos conjuntos X e Y es necesario que ningún atributo positivo de un conjunto sea negativo en el otro y viceversa. Por tanto, si $X = A\,B\,\overline{C}$ e $Y = A\,\overline{C}\,\overline{D}$, podemos obtener la unión $X \cup Y = A\,B\,\overline{C}\,\overline{D}$ y la intersección $X \cap Y = A\,\overline{C}$. Sin embargo, no es posible calcular ninguna de estas operaciones entre los itemset $X = A\,B\,\overline{C}$ e $Y = A\,\overline{B}\,\overline{D}$ porque el atributo B aparece como positivo en X y como negativo en Y. ■

Nótese que la unión de dos itemset es realmente una conjunción en el sentido de que las transacciones deben contener simultáneamente a X y a Y.

Proposición 2.8 *Para dos itemsets X e Y, se tiene la siguiente relación entre conjuntos de transacciones*

$$\{t \in \mathcal{T} \,/\, X \cup Y \preccurlyeq t\} = \{t \in \mathcal{T} \,/\, X \preccurlyeq t\} \cap \{t \in \mathcal{T} \,/\, Y \preccurlyeq t\}$$

Demostración: Se sigue directamente de que $X \cup Y \preccurlyeq t$ es equivalente a $(X \preccurlyeq t) \wedge (Y \preccurlyeq t)$. □

Las reglas de asociación formadas por los nuevos itemsets se definen de igual manera, es decir, una *regla de asociación* es una expresión de la forma $X \Rightarrow Y$ donde X e Y son itemsets. Sin embargo, los conjuntos X e Y deben cumplir las condiciones necesarias para poder definirse la unión de ambos.

Definición 2.12 *Dada una regla de asociación $X \Rightarrow Y$, se define su confianza denotada por $conf(X \Rightarrow Y)$ como el cociente entre la frecuencia de la unión y la frecuencia de X, es decir,*

$$conf(X \Rightarrow Y) = \frac{fr(X \cup Y)}{fr(X)}$$

Definición 2.13 *Dado por el usuario un valor de confianza mínimo denotado por δ, se dice que una regla $X \Rightarrow Y$ tiene confianza mayor que δ si $conf(X \Rightarrow Y) \geq \delta$.*

Ya tenemos todo los elementos para plantear el problema de encontrar reglas de asociación que se enuncia de esta forma:

> *"Dados un soporte mínimo σ y una confianza δ, se quieren obtener reglas $X \Rightarrow Y$ tales que $fr(X \cup Y) > \sigma$ y, además, $conf(X \Rightarrow Y) \geq \delta$."*

Básicamente, el proceso para generar las reglas de asociación consiste en:

1. Encontrar itemsets o conjuntos de atributos que sean frecuentes,

2. a partir de cada itemset frecuente Z, obtener todas las posibles particiones formadas por dos subconjuntos X e Y y

3. determinar la confianza de las dos reglas que se obtienen para cada partición.

Si Z es un conjunto frecuente de cardinal ℓ, el número de reglas posible es

$$\sum_{i=1}^{\ell-1} \binom{\ell}{i} = 2^{\ell} - 2 = 2 \cdot \left(2^{\ell-1} - 1\right)$$

una vez eliminada la partición trivial que no genera ninguna regla propia. Como vemos, en la generación de reglas se tiene en principio un número exponencial de subconjuntos. Sin embargo, realmente se obtienen pocos conjuntos frecuentes porque acotamos el número de atributos negativos que pueden aparecer en los itemsets frecuentes.

Ejemplo 2.4 Sea $\mathcal{T}$ una base de datos sobre el conjunto $R = \{A, B, C, D\}$ de atributos y supongamos que el itemset $Z = A\,B\,\overline{D}$ es frecuente y de cardinal 3. Para calcular todas las reglas posibles que se generan a partir Z consideramos todos los subconjuntos no triviales de Z que determinan 3 particiones distintas. Cada una de ellas da lugar a dos posibles reglas

según vemos en la tabla

Subconjuntos propios de Z	Partición $\{X, Y\}$	Regla $X \Rightarrow Y$
A	$\{A, B\,\overline{D}\}$	$A \Rightarrow B\,\overline{D}$
B	$\{B, A\,\overline{D}\}$	$B \Rightarrow A\,\overline{D}$
$\overline{D}$	$\{\overline{D}, A\,B\}$	$\overline{D} \Rightarrow A\,B$
$A\,B$	$\{A\,B, \overline{D}\}$	$A\,B \Rightarrow \overline{D}$
$A\,\overline{D}$	$\{A\,\overline{D}, B\}$	$A\,\overline{D} \Rightarrow B$
$B\,\overline{D}$	$\{B\,\overline{D}, A\}$	$B\,\overline{D} \Rightarrow A$

que hacen un total de 6 posibles reglas cuya confianza pueda ser mayor que un cierto valor mínimo prefijado. ■

El último paso para evaluar si una regla tiene confianza mayor o igual que δ requiere conocer las frecuencias de todos los subconjuntos de Z para realizar el correspondiente cociente. Hemos demostrado que si Z es frecuente entonces todos sus subconjuntos también lo son y por lo tanto conocemos sus frecuencias. Por tanto, vamos a centrarnos en el problema de encontrar todos los conjuntos frecuentes porque es la fase que conlleva mayor coste computacional.

2.1.4 Complejidad de la búsqueda de itemsets frecuentes

Para nosotros, que una palabra del lenguaje sea interesante es lo mismo que decir que un itemset sea frecuente, es decir, nuestro predicado q evalúa si el itemset es frecuente. Por lo tanto, aplicando el teorema 2.1, se puede establecer una biyección entre el conjunto $Fr(\mathcal{T}, \sigma)$ de itemsets frecuentes y la teoría $Th(\mathcal{L}, \mathcal{T}, q)$.

Veamos que la complejidad del problema de obtener todos los conjuntos frecuentes vuelve a ser la misma que en el algoritmo teórico levelwise.

Definición 2.14 *Sea σ un soporte mínimo y sea $Fr(\mathcal{T}, \sigma)$ un subconjunto de itemsets de $\mathcal{I}$ que es cerrado bajo $\preccurlyeq$. Definimos la frontera positiva de*

$Fr(\mathcal{T},\sigma)$ como el conjunto

$$Bd^{+}(Fr(\mathcal{T},\sigma)) = \{X \in Fr(\mathcal{T},\sigma) / \forall\, Y \in \mathcal{I}, X \preccurlyeq Y \Rightarrow fr(Y) \leq \sigma\}$$

Paralelamente se define la frontera negativa de $Fr(\mathcal{T},\sigma)$ como el conjunto

$$Bd^{-}(Fr(\mathcal{T},\sigma)) = \{X \in \mathcal{I} - Fr(\mathcal{T},\sigma) / \forall\, Y \in \mathcal{I}, Y \preccurlyeq X \Rightarrow fr(Y) > \sigma\}$$

y la frontera del conjunto $Fr(\mathcal{T},\sigma)$

$$Bd(Fr(\mathcal{T},\sigma)) = Bd^{+}(Fr(\mathcal{T},\sigma)) \cup Bd^{-}(Fr(\mathcal{T},\sigma))$$

Con esta definición, el problema de obtener todos los itemsets frecuentes se reduce a determinar los elementos maximales del conjunto $Fr(\mathcal{T},\sigma)$ que constituyen la frontera positiva. Por tanto, el problema es equivalente a encontrar la frontera positiva de la teoría $Th(\mathcal{L},\mathcal{T},q)$ para un lenguaje y un predicado adecuados. En este caso, el predicado "ser interesante" se interpreta como "ser frecuente".

En [Mannila y Toivonen, 1997] se determina que la complejidad de cualquier algoritmo para calcular $Th(\mathcal{L},\mathcal{T},q)$ es $|Bd(Th(\mathcal{L},\mathcal{T},q))|$. Por tanto, la complejidad del problema de encontrar todos los conjuntos frecuentes vuelve a ser $|Bd(Fr(\mathcal{T},\sigma))|$.

La proposición 2.4 justifica la búsqueda de los itemsets frecuentes por niveles como se define en el algoritmo levelwise. Si la enumeración de los itemsets se hace de abajo arriba pararemos cuando todos los itemsets frecuentes maximales hayan sido generados. Simétricamente la enumeración se parará cuando se hayan generado todos los itemsets no frecuentes minimales.

La aportación de nuestro algoritmo es la disminución del número de itemsets para los que hay que calcular su frecuencia mediante pasadas por la base de datos. Esto se logra procesando las frecuencias en un orden adecuado. Además, para calcular el resto de las frecuencias establecemos relaciones que permiten determinar unas frecuencias a partir de otras.

2.2 Consideraciones de búsqueda

Como en el planteamiento teórico, consideramos dos tareas en el cálculo de itemsets frecuentes:

1. Determinar los itemsets candidatos a ser frecuentes y
2. comprobar si estos candidatos son realmente frecuentes.

En esta sección vamos a establecer las propiedades que permiten establecer un método para realizar estas dos tareas y diseñar el algoritmo correspondiente.

Sea $R = \{A_i : i \in I\}$ un conjunto de atributos y sea $\mathcal{I}$ el correspondiente conjunto de itemsets. Si el cardinal de R es n, se puede comprobar que existen 3^n itemsets distintos. Sin embargo, para el problema de buscar conjuntos frecuentes podemos prescindir del itemset impropio $\emptyset = A^{\emptyset,\emptyset}$ y del itemset $A^{\emptyset,I}$ que en la práctica tiene frecuencia 0. Por tanto, a lo sumo hay $3^n - 2$ conjuntos frecuentes y el objetivo es proporcionar un algoritmo que permita encontrarlos evitando evaluar todos los conjuntos mediante pasadas por la base de datos.

2.2.1 Representación de los itemsets

La relación de especialización $\preccurlyeq$ definida en el conjunto de itemsets se puede representar mediante un diagrama de Hasse o diagrama de un conjunto parcialmente ordenado. Para visualizar mejor el proceso de búsqueda, vamos a considerar este diagrama como una estructura espacial que dispone los itemsets en niveles y plantas. Cada nivel contiene todos los itemsets con igual número de atributos y cada planta contiene todos los itemsets con el mismo número de atributos negativos.

Definición 2.15 *Sea $R = \{A_i : i \in I\}$ un conjunto de atributos y sea $\mathcal{I}$ el conjunto de todos los itemsets. Llamamos $I_{f,\ell}$ al conjunto de los ℓ-itemsets*

f-negativos de $\mathcal{I}$, es decir,

$$I_{f,\ell} = \{A^{p,s} \in \mathcal{I} \,/\, |p \cup s| = \ell,\ |s| = f\}$$

Observemos que el conjunto $I_{f,\ell}$ corresponde al elemento de la estructura situado en el nivel ℓ (*level*) de la planta f (*floor*). Además, estos elementos de la estructura determina una partición en el conjunto $\mathcal{I}$ de todos los itemsets.

Si $|R| = n$ entonces la estructura consta de $n+1$ plantas que se enumeran de 0 a n. Todas las plantas no tienen el mismo número de niveles de tal manera que los itemsets de la planta f se organizan en $n - f + 1$ niveles. En cada nivel ℓ hay $\binom{n}{\ell} \cdot 2^{\ell}$ itemsets (ℓ-itemsets) y en cada planta f se encuentran todos los itemsets con f atributos negativos. De esta manera, pueden existir a lo sumo $\binom{n}{f} \cdot 2^{n-f}$ itemsets frecuentes con f atributos negativos y por lo tanto,

$$\sum_{f=0}^{k} \binom{n}{f} \cdot 2^{n-f}$$

determina el número máximo de itemsets frecuentes con a lo sumo k atributos negativos. Veamos un ejemplo sencillo para mostrar esta estructura de itemsets.

Ejemplo 2.5 Sea $R = \{A, B, C, D\}$ un conjunto de cuatro atributos. En la figura 2.2 se representa en el plano la estructura que contiene todos los posibles itemsets. Cada elemento de la estructura se ha representado por un rectángulo que corresponde al conjunto $I_{f,\ell}$ en función de la planta f (columna) y del nivel ℓ (fila) al que pertenece. En cada rectángulo se representan algunos o todos los itemsets correspondientes a ese conjunto y, junto a ellos, el par (f, ℓ) que indica la planta f y el nivel ℓ al que pertenecen. ■

La importancia de esta estructura radica en la relación entre sus elementos. Por un lado, si consideramos una determinada planta, cada itemset de

$ABCD$ (0,4)	$AB\overline{C}D$,..... (1,4)	$\overline{A}\overline{B}CD$,..... (2,4)	$\overline{A}\overline{B}\overline{C}D$,..... (3,4)	$\overline{A}\overline{B}\overline{C}\overline{D}$ (4,4)
ABC, BCD,..... (0,3)	$A\overline{B}D$,..... (1,3)	$\overline{A}\overline{B}D$,..... (2,3)	$\overline{B}\overline{C}D$,..... (3,3)	
AB, BC,CD,..... (0,2)	$A\overline{B}$,.... (1,2)	$\overline{B}\overline{D}$,... (2,2)		
A,B,C,D (0,1)	$\overline{A},\overline{B},\overline{C},\overline{D}$ (1,1)			
$\emptyset$ (0,0)				

Figura 2.2: Estructura del espacio de itemsets

un nivel está relacionado (en el sentido $\preccurlyeq$) con algunos de los itemsets de los niveles consecutivos en esa misma planta. Por otro lado, si nos fijamos en dos plantas consecutivas, cada itemset de un nivel de la planta inferior está relacionado con algunos de los itemsets del siguiente nivel de la planta superior.

Proposición 2.9 *Sea* $R = \{A_i : i \in I\}$ *un conjunto de atributos y consideramos los itemsets* $X \in I_{f,\ell}$ *e* $Y \in I_{g,\ell+1}$ *con* $\ell < |I|$. *Si* $X \preccurlyeq Y$ *entonces* $g = f$ *o bien* $g = f + 1$.

Demostración: Consideramos las siguientes implicaciones:

$$\begin{array}{llllll} \text{- Si} & X = A^{p,s} \in I_{f,\ell} & \text{entonces} & |p \cup s| = \ell & \text{y} & |s| = f. \\ \text{- Si} & Y = A^{q,t} \in I_{f,\ell+1} & \text{entonces} & |q \cup t| = \ell + 1 & \text{y} & |t| = g. \end{array}$$

Como $X \preccurlyeq Y$ entonces $p \subseteq q$ y $s \subseteq t$ y, por lo tanto, se verifica alguna de estas condiciones:
o bien $p = q$ y $t = s \cup \{i\}$ con $i \in I - (p \cup s)$ y entonces $g = f + 1$
o bien $s = t$ y $q = p \cup \{i\}$ con $i \in I - (p \cup s)$ y entonces $g = f$. □

Además, si consideramos un determinado nivel, cualquier itemset de ese nivel sólo está relacionado (en el sentido $\preccurlyeq$) consigo mismo.

Proposición 2.10 *Sea $R = \{A_i : i \in I\}$ un conjunto de atributos. Para cada $\ell \leq |I|$ consideramos el conjunto $I_\ell = \{A^{p,s} \in \mathcal{I} \,/\, |p \cup s| = \ell\}$ de todos los ℓ-itemsets. Sean X e Y dos itemsets del conjunto I_ℓ. Si $X \preccurlyeq Y$ entonces $X = Y$, es decir, en I_ℓ no hay dos elementos distintos que estén relacionados.*

Demostración: Sean $X = A^{p,s}$ e $Y = A^{q,t}$. Si $X \preccurlyeq Y$ entonces sabemos que se cumple:

$$\begin{array}{llll} \text{que} & p \subseteq q & \text{y por tanto} & |p| \leq |q| \quad \text{y además} \\ \text{que} & s \subseteq t & \text{y por tanto} & |s| \leq |t|. \end{array}$$

Como X e Y pertenecen a I_ℓ sabemos que $|p \cup s| = |q \cup t| = \ell$, es decir,

$$|p| + |s| = |q| + |t|$$

y por lo tanto no queda más remedio que cumplirse que $p = q$ y $s = t$ y entonces $X = Y$. □

Obsérvese que el conjunto I_ℓ de la proposición anterior es igual a la unión de todos los elementos de la estructura situados en el nivel ℓ, es decir,

$$I_\ell = \bigcup_{f \leq \ell} I_{f,\ell} \qquad \text{para todo} \quad \ell = 0, 1, \ldots n$$

y por tanto, como explicábamos antes, no existen dos itemsets distintos del mismo nivel que estén relacionados.

Por último, veamos que para cada nivel podemos elegir itemsets por plantas de tal manera que el conjunto de estos itemsets determina una partición en la base de datos y, por lo tanto, la suma de todas sus frecuencias es 1.

Proposición 2.11 *Sea $\mathcal{T}$ una base de datos sobre el conjunto de atributos $R = \{A_i : i \in I\}$. Para cada $\ell \leq |I|$ consideramos un subconjunto de índices $u \in \mathcal{P}(I)$ tal que $|u| = \ell$. Entonces el conjunto $I_\ell^u = \{A^{p,s} \in \mathcal{I} \,/\, p \cup s = u\}$ verifica las siguientes propiedades:*

1. *Para cada* $I_{f,\ell}$ *con* $f \leq \ell$ *existe al menos un* $X \in I_{f,\ell}$ *tal que* $X \in I^u_\ell$.
2. *La suma de las frecuencias es 1, es decir,* $\sum_{X \in I^u_\ell} fr(X) = 1$.

Demostración: Para comprobar la propiedad *1* vamos a construir el correspondiente itemset X para cada conjunto $I_{f,l}$. Sea $u = \{i_1, i_2, \ldots i_\ell\}$ el subconjunto de índices. Para cada $I_{f,\ell}$ con $f \leq \ell$ consideramos los conjuntos $s = \{i_j \ / \ j \leq f\}$ y $p = u - s$. Ahora, el itemset $X = A^{p,s}$ pertenece al conjunto I^u_ℓ porque $p \cup s = u$ y, además, $X \in I_{f,\ell}$ porque $|p \cup s| = |u| = \ell$ y $|s| = f$.

La propiedad *2* presenta el mismo resultado obtenido en la proposición 2.7. □

Como veremos, esta estructura resulta útil para describir el procedimiento de búsqueda de conjuntos frecuentes; tanto en la generación de candidatos, como en el cálculo directo de algunas frecuencias sin mirar en la base de datos.

2.2.2 Cálculo de frecuencias

Vamos a determinar la relación entre las frecuencias de itemsets de plantas consecutivas para evitar la evaluación explícita (pasada por la base de datos) de muchas de ellas. Además, obtenemos resultados que permiten reducir el número de posibles itemsets frecuentes en el proceso de generación de candidatos cuando σ es mayor que 0,5.

Consideramos dos formas duales de atacar el problema. La primera consiste en empezar considerando los itemsets de atributos positivos e ir añadiendo atributos negativos progresivamente. En este caso, si se procesan en un orden adecuado, las frecuencias de los nuevos itemsets se obtienen a partir de las frecuencias de algunos itemsets de la planta anterior. La forma dual consiste en empezar considerando los itemsets con todos los atributos negativos e ir añadiendo atributos positivos progresivamente.

En este trabajo, consideramos la primera forma que consiste en ir añadiendo información negativa (es decir atributos negativos) de forma progresiva y controlada. Para ello, utilizamos una descomposición de las frecuencias de un itemset en términos de las frecuencias de los itemsets de la planta anterior. En general, a partir de los niveles $\ell - 1$ y ℓ de la planta $f - 1$, podemos obtener información sobre la frecuencia de los itemsets del nivel ℓ de la planta f.

Teorema 2.3 *Sea $p \in \mathcal{P}(I)$ y $s \in \mathcal{P}(I - p)$ tal que $|s| \geq 1$. Entonces para todo $j \in s$ se verifica que*

$$fr(A^{p,s}) = fr(A^{p,s-\{j\}}) - fr(A^{p\cup\{j\},s-\{j\}})$$

Demostración: Por la proposición 2.6 sabemos que

$$fr(A^{p,s}) = fr(A^{p\cup\{i\},s}) + fr(A^{p,s\cup\{i\}}).$$

Considerando los conjuntos $q = p$ y $t = s \cup \{i\}$ tenemos

$$fr(A^{q,t-\{i\}}) = fr(A^{q\cup\{i\},t-\{i\}}) + fr(A^{q,t})$$

y despejando $fr(A^{q,t})$ tenemos el resultado

$$fr(A^{q,t}) = fr(A^{q,t-\{j\}}) - fr(A^{q\cup\{j\},t-\{j\}})$$

para $p = q$ y $s = t$. □

Esta proposición nos permite calcular la frecuencia de cualquier itemset $A^{p,s} \in I_{f,\ell}$ que no esté en la planta 0, como diferencia de las frecuencias de dos itemsets que están respectivamente en $I_{f-1,\ell-1}$ y en $I_{f-1,\ell}$. Además, hay $|s|$ posibles formas de descomponer $fr(A^{p,s})$ y todas ellas nos proporcionan el mismo resultado.

Ejemplo 2.6 Sea $\mathcal{T}$ una base de datos sobre el conjunto de atributos $R = \{A, B, C, D\}$ y supongamos que se han calculado la frecuencia de

los itemsets A, C y AC de la planta 0. A partir de ellos, podemos calcular las frecuencias de los itemsets $\overline{A}$, $\overline{C}$, $A\,\overline{C}$ y $\overline{A}\,C$ de la planta 1.

$$\begin{array}{lcll} fr(\overline{A}) & = & fr(\emptyset) - fr(A) & = & 1 - fr(A) \\ fr(\overline{C}) & = & fr(\emptyset) - fr(C) & = & 1 - fr(C) \\ fr(A\,\overline{C}) & = & fr(A) - fr(AC) & & \\ fr(\overline{A}\,C) & = & fr(C) - fr(AC) & & \end{array}$$

Incluso podemos calcular la frecuencia del itemset $\overline{A}\,\overline{C}$ de la planta 2 a partir de las frecuencias calculadas y, en este caso, de dos formas distintas:

$$fr(\overline{A}\,\overline{C}) = fr(\overline{A}) - fr(\overline{A}\,C) = 1 - fr(\overline{A}) - fr(C) + fr(AC)$$

o bien

$$fr(\overline{A}\,\overline{C}) = fr(\overline{C}) - fr(A\,\overline{C}) = 1 - fr(\overline{C}) - fr(A) + fr(AC)$$

■

2.2.3 Generación de candidatos

Desarrollamos ahora las proposiciones que aplicaremos en la generación de candidatos.

Proposición 2.12 *Sea $p \in \mathcal{P}(I)$ y $s \in \mathcal{P}(I-p)$ tal que $|s| \geq 1$. Si σ es el soporte mínimo entonces se verifica que*

1. *$fr(A^{p,s}) > \sigma$ si y sólo si existe $j \in s$ tal que $fr(A^{p,s-\{j\}}) > \sigma + fr(A^{p\cup\{j\},s-\{j\}})$.*
2. *Si existe $j \in s$ tal que $fr(A^{p,s-\{j\}}) < \sigma$ entonces $fr(A^{p,s}) < \sigma$.*

Demostración: Esta proposición es una consecuencia inmediata del teorema 2.3. Para demostrar las dos propiedades basta sustituir la frecuencia del itemset $A^{p,s}$ por su correspondiente descomposición en diferencia de frecuencias. En la propiedad *2*, además hay que tener en cuenta que la frecuencia de cualquier itemset es siempre mayor o igual que 0. □

Esta proposición establece los criterios necesarios para determinar si un itemset es frecuente en función de la información sobre la frecuencia de los itemsets de la planta anterior.

Ejemplo 2.7 Sea $\mathcal{T}$ una base de datos sobre el conjunto $R = \{A, B, C, D\}$ de atributos y supongamos que se ha calculado la frecuencia del itemset $A\overline{C}$ del nivel 2 en la planta 1 y ha resultado ser no frecuente. En este caso, sabemos que los itemsets $A\overline{B}\,\overline{C}$ y $A\overline{C}\,\overline{D}$ de la planta 2 son no frecuentes y por supuesto cualquier itemset que sea mayor que $A\overline{C}$ en en el sentido dado por $\preccurlyeq$. ∎

Para los resultados anteriores sobre la relación entre las frecuencias de itemsets podemos considerar sus resultados duales intercambiando los atributos positivos con los negativos.

Proposición 2.13 *Sea $p \in \mathcal{P}(I)$ con $|p| \geq 1$ y $s \in \mathcal{P}(I - p)$. Si σ es el soporte mínimo entonces se verifica que*

1. $fr(A^{p,s}) = fr(A^{p-\{i\},s}) - fr(A^{p-\{i\},s\cup\{i\}}), \quad \forall i \in p.$
2. $fr(A^{p,s}) > \sigma$ *si y sólo si existe* $i \in p$ *tal que* $fr(A^{p-\{i\},s}) > \sigma + fr(A^{p-\{i\},s\cup\{i\}})$.
3. *Si existe* $i \in p$ *tal que* $fr(A^{p-\{i\},s}) < \sigma$ *entonces* $fr(A^{p,s}) < \sigma$.

Demostración: Identificar los itemsets $A^{p,s}$ con los correspondientes itemsets $A^{s,p}$ y aplicar el teorema 2.3 para demostrar la propiedad *1* y los resultados obtenidos en la proposición 2.12 para demostrar las propiedades *2* y *3*. □

Utilizando los resultados de estas proposiciones, existen distintas formas de recorrer la estructura. Por ejemplo, podemos calcular primero las frecuencias de los itemsets con el mayor número posible de atributos, obtener

así los itemsets frecuentes en ese lugar y determinar todos sus subconjuntos para usar esta información en la búsqueda de los demás conjuntos frecuentes.

En la práctica, este procedimiento no sería muy práctico ya que en las últimas plantas de la estructura es dificil que haya conjuntos frecuentes porque las frecuencias de esos itemsets son pequeñas. En este caso, las proposiciones no son útiles. En el ejemplo del supermercado, es poco probable que haya muchos clientes que realicen exactamente las mismas compras.

En la práctica, la frecuencia del itemset $A^{\emptyset,I}$ es 0 y resulta útil aplicar el siguiente resultado.

Proposición 2.14 *Si* $fr(A^{\emptyset,I}) = 0$ *entonces para cada* $i \in I$ *se verifica que*

1. $fr(A^{\{i\},I-\{i\}}) = fr(A^{\emptyset,I-\{i\}})$.
2. *Si* $fr(A^{\emptyset,I-\{i\}}) > \sigma$ *entonces para todo* $j \in I - \{i\}$ *se verifica que* $fr(A^{\{i\},I-\{i,j\}}) > \sigma$.

Demostración: La propiedad *1* es una aplicación directa del teorema 2.3 mientras que la propiedad *2* se deduce de la propiedad *1* y de que en este caso $A^{\{i\},I-\{i,j\}} \preccurlyeq A^{\{i\},I-\{i\}}$. Aplicando la proposición 2.3 se obtiene el resultado. □

También usaremos las siguientes propiedades que tienen en cuenta si el soporte mínimo exigido por el usuario σ es mayor o no que el 50%. Así podemos obtener una poda extra cuando se piden frecuencias muy altas (aunque este caso es poco habitual en la práctica).

Proposición 2.15 *Para todo atributo* $A \in R$ *se verifican las siguientes propiedades:*

1. $|fr(A) - 0,5| < |\sigma - 0,5| \Leftrightarrow |fr(\overline{A}) - 0,5| < |\sigma - 0,5|, \quad \forall \sigma \in [0,1]$.

2. *Si* $\sigma < 0,5$ *entonces*

$$fr(A) \leq \sigma \Rightarrow fr(\overline{A}) > \sigma \quad y \quad fr(A) > 1 - \sigma \Leftrightarrow fr(\overline{A}) < \sigma.$$

3. *Si* $\sigma = 0,5$ *entonces*

$$fr(A) \geq \sigma \Leftrightarrow fr(\overline{A}) \leq \sigma \quad y \quad fr(A) \leq \sigma \Leftrightarrow fr(\overline{A}) \geq \sigma.$$

4. *Si* $\sigma > 0,5$ *entonces*

$$fr(A) \geq \sigma \Rightarrow fr(\overline{A}) < \sigma \quad y \quad fr(A) < 1 - \sigma \Leftrightarrow fr(\overline{A}) > \sigma.$$

Demostración: La propiedad *1* se demuestra a partir de la definición de frecuencia de un atributo negativo y de las propiedades de los valores absolutos. El resto de las propiedades se deducen de los casos particulares de la propiedad para los distintos valores de σ. □

Para los demás itemsets de cardinal mayor que uno tenemos la siguiente proposición en el caso en que σ sea mayor que 0,5.

Proposición 2.16 *Sean* $p \in \mathcal{P}(I)$ *y* $s \in \mathcal{P}(I - p)$ *dos subconjuntos disjuntos de índices del conjunto* I. *Si* $\sigma > 0,5$ *se verifican las siguientes propiedades*

1. $\forall j \in s$, *si* $fr(A^{p,s-\{j\}}) > \sigma + fr(A^{p\cup\{j\},s-\{j\}})$ *entonces* $fr(A^{p\cup\{j\},s-\{j\}}) < \sigma$.

2. *Si* $\exists j \in s \,/\, fr(A^{p\cup\{j\},s-\{j\}}) > 1 - \sigma > \sigma$ *entonces* $fr(A^{p,s}) < \sigma$.

Demostración: La propiedad *1* es una consecuencia inmediata del teorema 2.3 y la propiedad *2* se deduce a partir de la contraposición de la propiedad *1*. □

La proposición anterior determina condiciones para la generación de candidatos y la determinación de la frecuencia de un itemset a partir de

determinadas frecuencias. Podemos generalizar este resultado considerando todos los itemsets correspondientes de las plantas anteriores. Esto se establece en la siguiente proposición.

Proposición 2.17 *Sean $p \in \mathcal{P}(I)$ y $s \in \mathcal{P}(I-p)$ dos subconjuntos disjuntos de índices del conjunto I. Si $\sigma > 0,5$ se verifica que*

$$\sum_{x \subseteq s, x \neq \emptyset} fr(A^{p \cup x, s-x}) > 1 - \sigma > \sigma \quad \Longrightarrow \quad fr(A^{p,s}) < \sigma$$

Demostración: Consideramos el conjunto $A^{p,\emptyset}$ y sea $u = s$. Por la proposición 2.5 tenemos que

$$fr(A^{p,\emptyset}) = \sum_{X \in I_s^{p,\emptyset}} fr(X) = fr(A^{p,s}) + \sum_{x \subseteq s, x \neq \emptyset} fr(A^{p \cup x, s-x})$$

y como $0 \leq fr(A^{p,\emptyset}) \leq 1$ se deduce que si $\sum fr(A^{p \cup x, s-x}) > 1 - \sigma$ entonces $fr(A^{p,s}) < \sigma$. □

Esta proposición determina condiciones para la generación de candidatos. Según esta condición para que un itemset sea candidato puede que tengamos que llegar a comprobar las frecuencias de un número demasiado alto de itemsets. Por este motivo, en el algoritmo usaremos la proposición 2.16.

2.2.4 Proceso de búsqueda

Básicamente, nuestro algoritmo usa la filosofía del algoritmo Apriori calculando los itemsets candidatos y frecuentes por niveles sucesivos. Nosotros generalizamos este procedimiento para incorporar el recorrido por las distintas plantas en la estructura espacial de itemsets.

Como hemos visto antes, las frecuencias de los itemsets de la planta f con $f \geq 1$, se pueden calcular a partir de determinadas frecuencias de itemsets de las plantas f y $f-1$. Por lo tanto, en un principio intentaremos que los únicos itemsets para los que se calcule su frecuencia directamente,

haciendo una pasada por la base de datos, sean aquellos que estén en la planta cero. Llegado a un cierto nivel, en la planta cero no habrá más itemsets frecuentes pero en las demás plantas sí puede seguir habiendo conjuntos frecuentes. Entonces otra planta tomará el papel de planta cero. A estas plantas que actúan como planta cero las llamamos planta base. Ahora desarrollamos con más detalle esta idea.

Supongamos que estamos en el nivel $\ell - 1$ y todos los $(\ell - 1)$-itemsets candidatos han sido clasificados en frecuentes y no frecuentes, veamos entonces como el algoritmo pasará a generar candidatos en el nivel ℓ.

En la planta $k = 0$, por la proposición 2.4, $A^{p,\emptyset} \in I_{0,\ell}$ será candidato si todos sus correspondientes subconjuntos del nivel $\ell - 1$ son frecuentes, es decir, si $A^{p-\{i\},\emptyset}$ es frecuente para todo $i \in p$. Para el resto de las plantas, el itemset $A^{p,s} \in I_{f,\ell}$ con $f \geq 1$ será candidato si para cada $i \in p$, $A^{p-\{i\},s} \in I_{f,\ell-1}$ es frecuente y si para cada $j \in s$, $A^{p,s-\{j\}} \in I_{f-1,\ell-1}$ es frecuente. De esta manera, para generar los candidatos de un nivel de la planta f sólo necesitamos considerar los itemsets del nivel anterior en las plantas $f - 1$ y f.

Para determinar si estos candidatos $A^{p,s} \in I_{f,\ell}$ son frecuentes sólo hay que comprobar, aplicando la proposición 2.12, si $\exists\, j \in s$, $fr(A^{p,s-\{j\}}) > \sigma + fr(A^{p\cup\{j\},s-\{j\}})$. Para ello, antes hemos tenido que calcular la frecuencia de los itemsets $A^{p\cup\{j\},s-\{j\}} \in I_{f-1,\ell}$.

Diseñamos un algoritmo que sea capaz de calcular las frecuencias de los itemsets en un orden adecuado sin aumentar el número de pasadas por la base de datos y evaluando sólo las frecuencias de las plantas que actúan como planta cero.

En cualquiera de las versiones que presentamos, el algoritmo siempre comienza calculando la frecuencia de los itemsets con un sólo atributo y, además, positivo, es decir, los itemsets del conjunto $I_{1,1}$. La proposición 2.15 permite calcular la frecuencia de los itemsets formados por un sólo atributo a partir de la frecuencia de su complementario y, por lo tanto,

podemos saber la frecuencia de todos los itemsets con un único atributo. Una vez procesado el nivel $\ell = 1$, se saben cuáles son los únicos atributos que pueden incorporarse a los itemsets para generar candidatos. Esta información obtenida del primer nivel se usa en el algoritmo para refinar el conjunto de índices asociado a los atributos negativos.

Además, una vez fijado un nuevo nivel, el recorrido por las distintas plantas nos proporciona otro refinamiento en el algoritmo para la generación de itemsets candidatos. Supongamos que todos los itemsets $A^{p,s} \in I_{f,\ell}$ son no frecuentes. Entonces, mediante la relación de orden sabemos que $A^{q,t}$ no es frecuente para todo $p \subseteq q$ y para todo $s \subseteq t$. Visualicemos este resultado en la estructura: si consideramos que cada rectángulo $I_{f,\ell}$ de la estructura espacial de itemsets se identifica con el punto (f, l) del plano cartesiano XY, entonces la región en la cual sabemos que no existirán conjuntos frecuentes es la intersección de las regiones $x \geq f$ e $y \geq x+(\ell-f)$. En el algoritmo, esta información se almacena en el conjunto de índices $J \in I$. Para cada nivel, en el momento de procesarlo, el conjunto J contiene los índices correspondientes a las plantas que no contienen ningún candidato.

Ejemplo 2.8 Sea $\mathcal{T}$ una base de datos sobre el conjunto de atributos $R = \{A, B, C, D\}$ y supongamos que $\sigma = 0.4$. Vamos a explicar el proceso de generación de candidatos y el camino que sigue nuestro algoritmo para la búsqueda de conjuntos frecuentes. Supongamos que los itemsets frecuentes maximales que encontraremos son ABC, $AB\overline{C}$, and $A\overline{B}$.

Por lo tanto, A, B, C son atributos frecuentes, y también $\overline{B}$ y $\overline{C}$ son atributos negativos frecuentes.

Al comienzo, encontramos que D, $\overline{A}$, y $\overline{D}$ no aparecerán en ningún itemset frecuente. El algoritmo almacena esta información mediante el nuevo conjunto $\overline{I}$. En el siguiente paso, tomamos como candidatos, primero los itemsets correspondientes en $I_{0,2}$, segundo en $I_{1,2}$, y por último en $I_{2,2}$ que cumplan las condiciones.

Encontramos como itemsets frecuentes a AB, AC, BC, $A\overline{B}$, $A\overline{C}$, y $B\overline{C}$.

En este momento, sabemos que no existen itemsets frecuentes en $I_{2,2}$. Por lo tanto, no existirán itemsets frecuentes en (f, ℓ) con $f \geq 2$, $\ell > 2$ y $\ell \geq f$. Esta información la usa el algoritmo a través del conjunto J para refinar la búsqueda en la generación de candidatos. En el siguiente paso buscamos los itemsets frecuentes en $I_{0,3}$ y $I_{1,3}$ y ABC, $AB\overline{C}$ son itemsets frecuentes, y la búsqueda en el siguiente nivel, prueba que junto con $A\overline{B}$, ellos son los itemsets frecuentes maximales.

A lo largo de este ejemplo queda claro como actuaría el algoritmo en el caso que diéramos una cota sobre los atributos negativos en los itemsets: descartando plantas que no cumplan esta limitación. ■

2.3 Algoritmos de búsqueda de itemsets frecuentes

Presentamos ahora los algoritmos para la búsqueda de itemsets frecuentes. Damos primero un algoritmo donde se explica cuál es el método que hemos usado para alcanzar nuestro objetivo de hacer las mismas pasadas que Apriori por la base de datos pero sin tener que evaluar todas las frecuencias. Para ello, recorremos la estructura de itemsets para procesar las frecuencias en un orden adecuado y poder obtener unas en función de otras.

La idea básica es realizar el recorrido por niveles (de esta forma nos aseguramos que en la generación de los candidatos del siguiente nivel tenemos la información sobre la frecuencia de todos sus subconjuntos). Además, en cada nivel se irán calculando las frecuencias de los itemsets progresivamente por plantas, es decir, comenzaremos calculando en la base de datos, la frecuencia de los itemsets de la planta base. Con estas frecuencias podremos calcular la frecuencia de los itemsets de la siguiente planta mediante las diferencias definidas en el teorema 2.3. Sucesivamente continuamos este proceso hasta que terminemos con todas las plantas de ese nivel donde haya candidatos.

Primero presentamos un algoritmo más teórico donde se integran los casos $\sigma \leq 0,5$ y $\sigma > 0,5$ y donde suponemos que siempre que necesitemos una frecuencia para realizar la diferencia la tenemos calculada. Posteriormente, presentaremos dos algoritmos, uno para cada condición de corte de σ donde vamos obteniendo las frecuencias de forma ordenada para disponer de ellas en el momento de calcular frecuencias por diferencias.

Hasta ahora hemos dicho que seguiríamos la filosofía de Apriori de recorrer la estructura de itemsets por niveles, pero debido a las particularidades de nuestro caso también se podría pensar recorrer la estructura por plantas para encontrar los itemsets frecuentes. El algoritmo que describe ese proceso se encuentra en el apéndice B.

2.3.1 Algoritmo Neg–Apriori

Presentamos ahora los algoritmos de una forma más precisa. En un principio no se hace ninguna restricción sobre el número de atributos negativos que pueden aparecer en los itemsets frecuentes. En el apéndice A se muestran todos los algoritmos que calculan los itemsets frecuentes con a lo sumo un número determinado de atributos negativos.

Entrada del algoritmo:

- Un conjunto R de atributos o su correspondiente conjunto I de índices.
- Una base de datos $\mathcal{T}$.
- Un soporte mínimo $\sigma \in (0, 1)$.

Salida del algoritmo:

- El conjunto de todos los itemsets frecuentes.

Parámetros del algoritmo:

- f es la planta base y su valor indica el número de atributos negativos que tiene el itemset. Corresponde a la planta donde hay que evaluar las frecuencias directamente; la primera es la planta cero y progresivamente se va cambiando la planta base a medida que llegamos a un nivel vacío en la planta base.

- ℓ es el nivel y su valor indica el cardinal del itemset.

- k es la variable interna utilizada para recorrer las distintas plantas de un mismo nivel.

- $C_{k,\ell}$ es el conjunto de itemsets candidatos del nivel ℓ de la planta k.

- $L_{k,\ell}$ es el conjunto de itemsets frecuentes del nivel ℓ de la planta k.

- $\overline{I}$ es el subconjunto de índices cuyos atributos son los únicos que podrán aparecer en los itemsets frecuentes con atributos negativos. Si $\overline{I} = \emptyset$ no se tendrán itemsets frecuentes con atributos negativos y por lo tanto se puede usar cualquier algoritmo clásico (p.e. Apriori) que sólo considere atributos positivos.

- J indica, en cada nivel, las plantas donde no habrá itemsets frecuentes.

El algoritmo recorre la estructura espacial en el orden primero en nivel y segundo en planta, es decir, realiza el siguiente camino:

$$(0,1),(1,1);(0,2),(1,2),(2,2);(0,3),(1,3),(2,3),(3,3);(0,4),\ldots$$

donde las comas separan las plantas y los punto y comas separan los niveles.

En los conjuntos $C_{k,\ell}$ con $k \geq 1$ del algoritmo se explicitan las condiciones de pertenencia al conjunto pero se ha omitido su construcción. El conjunto $A^{p,s} \in C_{k,\ell}$ se genera como candidato a partir de los conjuntos $A^{p,s'} \in L_{k-1,\ell-1}$ con $m \in \overline{I} - (p \cup s')$ y $s = s' \cup \{m\}$. Inicialmente, suponemos que todos los conjuntos $C_{f,\ell}$ y $L_{f,\ell}$ para $f \leq \ell \leq n$ son vacíos.

Ahora presentamos los algoritmos en pseudocódigo. Cada uno de ellos incluye uno o varios módulos para que su lectura sea más sencilla. Al finalizar, incluimos comentarios adicionales sobre ciertas instrucciones.

Algoritmo Neg-Apriori

1. $L_{0,1} := \{A^{\{i\},\emptyset},\ \forall i \in I/\ fr(A^{\{i\},\emptyset}) > \sigma\}$;
   $L_{1,1} := \{A^{\emptyset,\{i\}},\ \forall i \in I/\ fr(A^{\emptyset,\{i\}}) = 1 - fr(A^{\{i\},\emptyset}) > \sigma\}$;
   $\overline{I} := \{i \in I/A^{\emptyset,\{i\}} \in L_{1,1}\}$;
   if $\overline{I} = \emptyset$ **then** Apriori **else**

2. $C_{0,2} := \{A^{p,\emptyset} \in I_{0,2}/\ \forall i \in p,\ A^{p-\{i\},\emptyset} \in L_{0,1}\}$;
   $L_{0,2} := \{A^{p,\emptyset} \in C_{0,2}/\ fr(A^{p,\emptyset}) > \sigma\}$;

3. $f := 0$; $\ell := 2$; $J := \emptyset$;

4. **while** $f \leq \ell$ and $\ell \leq n$ **do**
   while $L_{f,\ell} \neq \emptyset$ and $\ell \leq n$ **do**
   $k := f + 1$;
   $L_{\ell,\ell-1} := \emptyset$;
   Cand-Freq
   if $\ell = n$ **then** output;
   $\ell := \ell + 1$;
   $J := J \cup \{k + 1/\ k \in J, k < n\}$;
   if $f = 0$ **then**
   $C_{0,\ell} := \{A^{p,\emptyset} \in I_{0,\ell}/\ \forall i \in p,\ A^{p-\{i\},\emptyset} \in L_{0,\ell-1}\}$;
   $L_{0,\ell} := \{A^{p,\emptyset} \in C_{0,\ell}/\ fr(A^{p,\emptyset}) > \sigma\}$
   else
   $L_{f,\ell} := \emptyset$
   fi
   od (while ℓ)
   if $f = \ell$ **then** output;
   $f := f + 1$;
   $C_{f,\ell} := \{A^{p,s} \in I_{f,\ell}/\ \forall i \in p, A^{p-\{i\},s} \in L_{f,\ell-1},$

$\forall j \in s,\ A^{p,s-\{j\}} \in L_{f-1,\ell-1}\}$;
$L_{f,\ell} := \{A^{p,s} \in C_{f,\ell} / \exists j \in s, fr(A^{p,s-\{j\}}) > \sigma + fr(A^{p\cup\{j\},s-\{j\}})\}$;
$J := J \cup \{f-1\}$
od (while f)

5. output $\bigcup_{k\leq\ell\leq n} L_{k,\ell}$
fi

En el grupo de instrucciones etiquetadas con 1, el algoritmo calcula primero los conjuntos frecuentes formados por un único atributo positivo y, a partir de ellos, el conjunto de itemsets frecuentes formados por un único atributo negativo. Una vez obtenida esta información se sabe cuando se suba a las demás plantas en el desarrollo del algoritmo con cuáles únicos atributos se pueden buscar conjuntos frecuentes con atributos negativos y cuando se aumente de nivel, con cuáles únicos atributos se pueden buscar conjuntos frecuentes con atributos positivos.

En el grupo de instrucciones etiquetadas con 2 y 3 se calculan los itemsets frecuentes del nivel 2 en la planta 0 y se inicializan las variable respectivamente. La última instrucción del algoritmo etiquetada con 5 proporciona la salida.

El grupo de instrucciones etiquetadas con 4 constituyen el cuerpo principal del algoritmo y es donde se produce el recorrido por la estructura en busca de los conjuntos frecuentes. La primera sentencia *while* establece la condición de finalización del proceso y la segunda sentencia *while* que se anida a ésta, corresponde a los cambios de planta base. El módulo **Cand-Freq** permite subir de planta a partir de la planta base para calcular los itemsets candidatos e itemsets frecuentes de cada nivel. El pseudocódigo correspondiente a este módulo es

Cand-Freq

while $k \leq \ell$ **do**
 if $k \notin J$ **then**
  $L_{k,\ell} := \emptyset$;
  if $\sigma \leq 0,5$ **then**
   $C_{k,\ell} := \{A^{p,s} \in I_{k,\ell} / \forall i \in p, A^{p-\{i\},s} \in L_{k,\ell-1},$
   $\forall j \in s, A^{p,s-\{j\}} \in L_{k-1,\ell-1}\}$
  else
   $C_{k,\ell} := \{A^{p,s} \in I_{k,\ell} / \forall i \in p, A^{p-\{i\},s} \in L_{k,\ell-1},$
   $\forall j \in s, A^{p,s-\{j\}} \in L_{k-1,\ell-1},$
   $\forall j \in s, fr(A^{p\cup\{j\},s-\{j\}}) < \sigma\}$
  fi
  $L_{k,\ell} := \{A^{p,s} \in C_{k,\ell} / \exists j \in s,\ fr(A^{p,s-\{j\}}) > \sigma + fr(A^{p\cup\{j\},s-\{j\}})\}$;
  if $L_{k,\ell} = \emptyset$ **then** $J := J \cup \{k\}$ **fi**
 fi
 $k := k + 1$;
od (while k)

Después de ejecutar este módulo, el algoritmo sube de nivel y se calculan los itemsets frecuentes del nivel siguiente de la planta base ($L_{f,\ell}$). Si este conjunto no es vacío volvemos a repetir el proceso (while ℓ) para volver a recorrer todas las plantas de ese nivel. Si el conjunto $L_{f,\ell}$ es vacío entonces cambiamos de planta base ($f := f + 1$) para calcular el nuevo conjunto $L_{f,\ell}$ y poder repetir de nuevo el proceso.

En todo este proceso se intenta aprovechar la mayor información posible entre las plantas y minimizar la evaluación de la frecuencia de los itemsets en la base de datos. Cuando $\sigma > 0,5$, en la generación de candidatos se tiene en cuenta que ningún itemset relacionado con el candidato en el mismo nivel y en la planta anterior sea frecuente. Incluso, usando la proposición 2.16, se puede exigir algo más restrictivo como que sea menor que $1 - \sigma$.

2.3.2 Algoritmo Neg–Apriori–1

Cuando se generan candidatos y se calculan frecuencias, se supone siempre que se tienen calculadas todas las frecuencias necesarias. Estos son los casos en los que hacen falta determinadas frecuencias:

- En $C_{k,\ell}$ cuando $\sigma > 0,5$ se necesita que $fr(A^{p\cup\{j\},s-\{j\}}) < \sigma$ para todo $j \in s$.
- Para cualquier σ se necesita que exista $j \in s$ tal que $fr(A^{p,s-\{j\}}) > \sigma + fr(A^{p\cup\{j\},s-\{j\}})$.

La frecuencia $fr(A^{p,s-\{j\}})$ siempre la tenemos porque para que $A^{p,s}$ sea candidato $A^{p,s-\{j\}}$ tiene que haber sido frecuente. Sin embargo, no siempre conocemos $fr(A^{p\cup\{j\},s-\{j\}})$ y esto puede deberse a que el itemset $A^{p\cup\{j\},s-\{j\}} \in I_{k-1,\ell}$ no haya sido candidato y por tanto, su frecuencia no ha sido calculada. Este conjunto puede no haber sido candidato por dos razones: porque no todos sus subconjuntos son frecuentes o porque caiga dentro del conjunto formado por J.

Por lo tanto, si queremos usar la diferencia de frecuencias, tendremos que calcular la frecuencia de todos estos itemsets directamente en la base de datos. Los siguiente algoritmos incorporan unos nuevos conjuntos de candidatos ($\tilde{C}_{k,\ell}$) correspondientes a estos itemsets. Para mantener el número de pasadas en la base de datos, las frecuencias de estos conjuntos se calculan a la vez que se calculan las frecuencias de los itemsets de la planta base.

El siguiente algoritmo se utiliza en el caso de que $\sigma \leq 0,5$ y obtiene los conjuntos frecuentes recorriendo la estructura por niveles completos.

Algoritmo Neg-Apriori-1 ($\sigma \leq 0,5$)

1. $L_{0,1} := \{A^{\{i\},\emptyset},\ \forall i \in I /\ fr(A^{\{i\},\emptyset}) > \sigma\}$;
   $L_{1,1} := \{A^{\emptyset,\{i\}},\ \forall i \in I /\ fr(A^{\emptyset,\{i\}}) = 1 - fr(A^{\{i\},\emptyset}) > \sigma\}$;

$\overline{I} := \{i \in I \,/ A^{\emptyset,\{i\}} \in L_{1,1}\}$;
if $\overline{I} = \emptyset$ **then** Apriori **else**

2. $f := 0$; $\ell := 1$; $J := \emptyset$;

3. **while** $f \leq \ell$ and $\ell < n$ **do**
   while $L_{f,\ell} \neq \emptyset$ and $\ell < n$ **do**
   $\ell := \ell + 1$;
   $L_{-1,\ell-1} := \emptyset$; $L_{\ell,\ell-1} := \emptyset$;
   if $f \notin J$ **then**
   $C_{f,\ell} := \{A^{p,s} \in I_{f,\ell} / \forall i \in p, A^{p-\{i\},s} \in L_{f,\ell-1},$
   $\forall j \in s, A^{p,s-\{j\}} \in L_{f-1,\ell-1}\}$
   else
   $L_{f,\ell} := \emptyset$
   fi
   Cand-Freq-level
   $J := J \cup \{k+1 / k \in J, k < n\}$;
   od (while ℓ)
   while $L_{f,\ell} = \emptyset$ and $f \leq \ell$ and $\ell < n$ **do**
   $J := J \cup \{f\}$;
   $f := f + 1$
   od(while f)
   od(while f and ℓ)

4. output $\bigcup_{k \leq \ell \leq n} L_{k,\ell}$
   fi

Al igual que el algoritmo anterior, en el grupo de instrucciones etiquetadas con 1 se calculan los conjuntos frecuentes formados por un único atributo y se refina el conjunto de índices para la generación de candidatos. Las instrucciones etiquetadas con 2 y 4 inicializan las variables y proporcionan la salida respectivamente.

El grupo de instrucciones etiquetadas con 3 constituyen el cuerpo principal del algoritmo y es donde se produce el recorrido por la estructura en busca de los conjuntos frecuentes. Igualmente, la primera sentencia *while* establece la condición de finalización del proceso y la segunda sentencia *while* que se anida a esta corresponde a los cambios de planta base. El módulo **Cand-Freq-level** permite subir de planta a partir de la planta base para calcular los itemsets candidatos e itemsets frecuentes de cada nivel. El pseudocódigo correspondiente a este módulo es

Cand-Freq-level

$k := f + 1;$
$L_{\ell,\ell-1} := \emptyset;$
while $k \leq \ell$ **do**
 if $k \notin J$ **then**
  $C_{k,\ell} := \{A^{p,s} \in I_{k,\ell} / \forall i \in p, A^{p-\{i\},s} \in L_{k,\ell-1},$
   $\forall j \in s, A^{p,s-\{j\}} \in L_{k-1,\ell-1}\};$
  $\tilde{C}_{k,\ell} := \{A^{p,s} \in C_{k,\ell} / \forall j \in s, A^{p\cup\{j\},s-\{j\}} \notin C_{k-1,\ell}\}$
 fi
 $k := k + 1$
od (while k)
$\tilde{C}_\ell := C_{f,\ell} \cup \left(\bigcup_{k:=f+1,k\notin J}^{\ell} \tilde{C}_{k,\ell}\right);$
$\tilde{L}_\ell := \{A^{p,s} \in \tilde{C}_\ell / \ fr(A^{p,s}) > \sigma\};$
$L_{f,\ell} := \tilde{L}_\ell \cap C_{f,\ell};$
$k := f + 1;$
while $k \leq \ell$ **do**
 if $k \notin J$ **then**
  $L_{k,\ell} := \{A^{p,s} \in C_{k,\ell} - \tilde{C}_{k,\ell} / \exists j \in s, \ fr(A^{p,s-\{j\}}) > \sigma + fr(A^{p\cup\{j\},s-\{j\}})\};$
  $L_{k,\ell} := L_{k,\ell} \cup (\tilde{L}_\ell \cap \tilde{C}_{k,\ell});$
  if $L_{k,\ell} = \emptyset$ **then** $J := J \cup \{k\}$ **fi**
 else $L_{k,\ell} := \emptyset$
 fi

$k := k + 1$
od(while k)

En el módulo Cand-Freq-level, la primera sentencia *while* genera todos los candidatos en el nivel ℓ. Además, comprueba qué candidatos necesitan calcularse directamente por la base de datos y va creando los subconjuntos $\tilde{C}_{k,\ell}$ de la siguiente forma

$$\tilde{C}_{k,\ell} := \{A^{p,s} \in C_{k,\ell} / \forall j \in s, A^{p\cup\{j\},s-\{j\}} \notin C_{k-1,\ell}\}$$

Después realiza la unión de todos estos conjuntos en ese nivel y crea el conjunto $\tilde{C}_\ell$. En este conjunto están todas los itemsets para los que hay que calcular su frecuencia directamente en la base de datos.

La segunda sentencia *while* calcula los itemsets frecuentes que restan mediante las diferencias y distribuye los itemsets frecuentes obtenidos en el conjunto anterior en cada conjunto de frecuentes correspondiente.

Después de esto, el algoritmo seguirá subiendo de nivel en el bucle *while* f *and* ℓ hasta que en la planta base exista un conjunto $L_{f,\ell} = \emptyset$ y sea necesario cambiar de planta base. Si es así, fijado un nivel, el bucle *while* f va aumentando de planta buscando el primer conjunto de itemsets frecuentes que sea no vacío para tomarlo como nueva planta base.

2.3.3 Algoritmo Neg–Apriori–2

Este algoritmo híbrido se utiliza en el caso de que $\sigma > 0,5$ y obtiene los conjuntos frecuentes recorriendo la estructura de las dos formas descritas en los algoritmos anteriores.

Algoritmo Neg-Apriori-2 $(\sigma > 0,5)$

1. $L_{0,1} := \{A^{\{i\},\emptyset}, \forall i \in I / fr(A^{\{i\},\emptyset}) > \sigma\}$;
   $L_{1,1} := \{A^{\emptyset,\{i\}}, \forall i \in I / fr(A^{\emptyset,\{i\}}) = 1 - fr(A^{\{i\},\emptyset}) > \sigma\}$;

$\overline{I} := \{i \in I \,/ A^{\emptyset,\{i\}} \in L_{1,1}\}$;
if $\overline{I} = \emptyset$ **then** Apriori **else**

2. $C_{0,2} := \{A^{p,\emptyset} \in I_{0,2} / \forall i \in p,\ A^{p-\{i\},\emptyset} \in L_{0,1}\}$;
   $L_{0,2} := \{A^{p,\emptyset} \in C_{0,2} / \ fr(A^{p,\emptyset}) > \sigma\}$;

3. $f := 0$; $\ell := 2$; $J := \emptyset$;

4. **while** $L_{0,\ell} \neq \emptyset$ and $\ell < n$ **do**
   Cand-level
   $C_{0,\ell+1} := \{A^{p,\emptyset} \in I_{0,\ell+1} / \forall i \in p,\ A^{p-\{i\},\emptyset} \in L_{0,\ell}\}$;
   $\tilde{C}_\ell := C_{0,\ell+1} \cup (\cup_{k:=1,k\notin J}^{\ell} \tilde{C}_{k,\ell})$;
   $\tilde{L}_\ell := \{A^{p,s} \in \tilde{C}_\ell \,/ \ fr(A^{p,s}) > \sigma\}$;
   $L_{0,\ell+1} := \tilde{L}_\ell \cap C_{0,\ell+1}$;
   Freq-level
   $J := J \cup \{k+1 / \, k \in J, k < n\}$;
   $\ell := \ell + 1$
   od (while ℓ)

5. **if** $L_{0,\ell} \neq \emptyset$ and $\ell = n$ **then** output;
   Cand-level
   $\tilde{C}_\ell := \left(\bigcup_{k:=1,k\notin J}^{\ell} \tilde{C}_{k,\ell}\right)$;
   $\tilde{L}_\ell := \{A^{p,s} \in \tilde{C}_\ell \,/ \ fr(A^{p,s}) > \sigma\}$;
   Freq-level
   if $\ell = n$ then output;
   while $L_{f,\ell} = \emptyset$ and $f \leq \ell$ **do**
   $J := J \cup \{f\}$;
   $f := f + 1$
   od(while f)

6. **Punto 3 del algoritmo de** $\sigma \leq 0,5$

7. output $\bigcup_{k\leq\ell\leq n} L_{k,\ell}$
   fi

Al principio el algoritmo se comporta como Neg-Apriori, de hecho, las instrucciones etiquetadas con 1, 2 y 3 son las mismas. Al llegar a las sentencia 6 se aplica el correspondiente punto 3 del algoritmo Neg-Apriori-1 y se finaliza el proceso con la salida que proporciona la instrucción etiquetada como 7. Pero, veamos qué ocurre en los grupos de instrucciones 4 y 5.

Mientras que la planta cero sea la planta base se sigue el recorrido que hace el algoritmo Neg-Apriori, es decir, por niveles obtiene los itemsets de cada planta que sean frecuentes y los itemsets frecuentes del siguiente nivel en la planta cero (Módulos: Cand-level y Freq-level). De esta forma se aprovecha la información de la planta anterior para generar los candidatos. Este proceso se corresponde al punto 4.

Cuando la planta cero deja de ser la planta base hay que volver al algoritmo Neg-Apriori-1 porque no podemos generar los candidatos del siguiente nivel ya que no se han calculado las frecuencias de los conjuntos $\tilde{C}_{k,\ell}$ del nivel anterior. En el punto 5 del algoritmo se calculan las frecuencias de los itemsets de estos conjuntos que quedan en ese nivel.

Cand-level

$k := 1$; $L_{\ell,\ell-1} := \emptyset$; $\tilde{C}_{0,\ell} = \emptyset$;
while $k \leq \ell$ **do**
 if $k \notin J$ **then**
  $C_{k,\ell} := \{A^{p,s} \in I_{k,\ell} \,/\, \forall i \in p, A^{p-\{i\},s} \in L_{k,\ell-1},$
  $\forall j \in s, A^{p,s-\{j\}} \in L_{k-1,\ell-1},$
  $\exists j \in s, A^{p\cup\{j\},s-\{j\}} \in C_{k-1,\ell} - \tilde{C}_{k-1,\ell},$
  $fr(A^{p\cup\{j\},s-\{j\}}) < 1 - \sigma\}$;
  $\tilde{C}_{k,\ell} := \{A^{p,s} \in C_{k,\ell} / \forall j \in s, A^{p\cup\{j\},s-\{j\}} \notin C_{k-1,\ell}\}$;
  $L_{k,\ell} := \{A^{p,s} \in C_{k,\ell} - \tilde{C}_{k,\ell} / \exists j \in s,\ fr(A^{p,s-\{j\}}) > \sigma + fr(A^{p\cup\{j\},s-\{j\}})\}$;
 $k := k + 1$
 fi
od (while k)

Freq-level

$k := 1;$
while $k \leq \ell$ **do**
 if $k \notin J$ **then**
  $L_{k,\ell} := L_{k,\ell} \cup (\tilde{L}_\ell \cap \tilde{C}_{k,\ell});$
  if $L_{k,\ell} = \emptyset$ **then** $J := J \cup \{k\}$ **fi**
 else $L_{k,\ell} := \emptyset$
 fi
 $k := k + 1$
od(while k)

2.3.4 Complejidad

Con respecto a la complejidad del algoritmo, desde un punto de vista teórico se consideran dos aspectos: la generación de candidatos y el cálculo de frecuencias de los itemsets.

El peor caso en la generación de candidatos se presenta cuando el soporte mínimo σ es menor o igual que 0,5. En este caso, dos itemsets, uno de ellos con un atributo particular positivo y el otro itemset con el mismo atributo negativo pueden ser frecuentes simultáneamente. Sin embargo, si $\sigma > 0,5$ por la proposición 2.16 se puede refinar la generación de candidatos. Además, independientemente del valor de σ los conjuntos I y J refinan la generación de candidatos.

Por otro lado, se pueden obtener las frecuencias de determinados itemsets como diferencia de frecuencias ya calculadas de itemsets de la planta anterior. Aquí es donde realmente se reduce la complejidad porque permite obtener la frecuencia de itemsets sin tener que calcularla directamente en

la base de datos.

Además, en todos los casos y como ocurre en el algoritmo Apriori, el número de pasadas por la base de datos es $m + 1$ siendo m el cardinal del itemset frecuente con mayor número de atributos. Supongamos que en el proceso de ir avanzando por niveles, hemos alcanzado (sin saber) el itemset frecuente de mayor cardinal y éste tiene m atributos. Entonces, en el caso peor, podría pasar que generásemos los itemsets de cardinal $m + 1$ y así, dando una última pasada por la base de datos comprobaríamos que ninguno era frecuente y por lo tanto el algoritmo acaba.

Capítulo 3

Generación de árboles de decisión con valores desconocidos

El problema de la generación de árboles de decisión se encuadra dentro del aprendizaje supervisado, sin conocimiento adicional y sin modelo. Es supervisado porque las experiencias están marcadas, es decir, conocemos a qué clase pertenecen, pero no tenemos ninguna información más. Tampoco podemos suponer un modelo que genere esas experiencias.

En este capítulo, desarrollamos nuestro enfoque para considerar las experiencias con valores desconocidos en la ejecución de los algoritmos de aprendizaje inductivo de árboles de decisión. La motivación de ello es la predicción de nuevas observaciones. Una vez generado el árbol de decisión, éste se convierte en una serie de reglas llamadas reglas inductivas con un parámetro de validez asociado. Este parámetro depende del número de experiencias que componen cada una de las hojas del árbol. Si existe un número mayor de experiencias, la predicción de la clase de nuevas observaciones será más precisa. Seguiremos la filosofía del algoritmo C4.5 en la

generación del árbol.

Desarrollamos tres puntos básicos para adaptar los pasos del algoritmo donde es necesario conocer el valor del atributo. Primero, modificamos el criterio de división para incluir los valores desconocidos. Segundo, explicamos cómo asignar valores a las experiencias con valores desconocidos en los atributos teniendo en cuenta los valores del atributo y los de la clase. Tercero, explicamos como se incluyen las experiencias en los subconjuntos de la partición del nodo. Por último, tratamos la predicción de la clase de las observaciones con valores desconocidos usando árboles de decisión con todas las variantes. Además, presentamos resultados experimentales donde se muestra que nuestro método obtiene mejores resultados que C4.5. Esto se debe a que nosotros tenemos en cuenta más información cuando se incluyen las experiencias en los subconjuntos de la partición.

En la segunda parte de este capítulo trataremos el problema de valores desconocidos como un problema de decisión con riesgo. De esta forma asignaremos un parámetro de confianza y un parámetro de error a cada asignación de valores que hagamos. También se los asignaremos a la predicción. Con ello, damos dos medidas informativas en la asignación de valores generalizando el método seguido por C4.5.

Experiencias y observaciones con valores desconocidos

Sea $\mathcal{T} = \{e_1, \ldots, e_N\}$ un conjunto finito no vacío de experiencias. En adelante consideramos el conjunto de atributos $A = \{X_1, \ldots, X_n, Y\}$ donde el atributo Y es un atributo especial llamado "clase".

Cada atributo X_i toma valores en $D_i = \{1, 2, \ldots, v(i)\}$ con $i = 1, \ldots, n$, donde v es una función $v : \{1, \ldots, n\} \to \mathbb{N}$. El atributo Y toma valores en $D = \{1, \ldots, h\}$ y se dice entonces que hay h clases.

Definición 3.1 *Sea $A = \{X_1, \ldots, X_n, Y\}$ un conjunto de atributos. Se define una* experiencia con valores desconocidos *y se denota por $e-$, como*

el vector $e- = (X_1(e-), X_2(e-), \ldots, X_n(e-), Y(e-))$ *donde el valor de algún atributo* X_i *es desconocido. Este valor desconocido lo representamos con el símbolo "?".*

Definición 3.2 *Sea* $R = \{X_i : i = 1, \ldots, n\}$ *un conjunto de atributos. Se define una* observación con valores desconocidos *y se denota por* $b-$*, como el vector* $b- = (X_1(b-), X_2(b-), \ldots, X_n(b-))$ *donde el valor de algún atributo* X_i *es desconocido. Este valor desconocido lo representamos con el símbolo "?".*

3.1 Criterio general de división

Definimos ahora un criterio general donde los valores desconocidos también se consideran al seleccionar un atributo en el nodo actual durante el proceso de construcción del árbol de decisión. Suponemos que los valores desconocidos se han perdido de forma aleatoria. El valor desconocido se considera como un nuevo valor del atributo y el criterio de división tendrá en cuenta este nuevo valor. El atributo con valores desconocidos se penaliza mediante un peso. Mostramos cómo cada criterio de división definido para trabajar con atributos sin valores desconocidos se puede adaptar para trabajar con valores desconocidos en los atributos.

Consideremos que el nodo actual es un nodo intermedio. Para simplificar la notación denotamos por T al conjunto de experiencias en el nodo actual.

Sea m cualquier medida y m' la función definidas sólo para experiencias con valores conocidos en los atributos . Sea $e-$ una experiencia que tiene desconocido el valor del atributo X_i, es decir, $X_i(e-) = ?$.

A cada atributo X_i de A se le asocia un test X^i que realiza una partición en los valores conocidos del atributo X_i y sea $\{\theta_j, j = 1, \ldots, u(i)\}$ esa partición.

Por tanto, los valores conocidos dados por X^i y el valor desconocido de X_i crean la partición $\{T_1, T_2, \ldots, T_{u(i)}, T_{u(i)+1}\}$ en el conjunto de experiencias actual donde $T_j = \{e \in T / X_i(e) = \theta_j\}$ con $j = 1, \ldots, u(i)$ y $T_{u(i)+1} = \{e- \in T / X_i(e-) = ?\}$

Sea $f_i = |\{e \in T \ / \ X_i(e) \neq ?\}|/|T|$. La función gradiente que definimos ahora tiene en cuenta esta proporción.

Definición 3.3 *Sea X^i el test asociado al atributo X_i con $i = 1, \ldots, n$. Se define la función gradiente $\Delta : R \times \mathcal{P}(\mathcal{T}) \rightarrow \mathbb{R}^+$ donde*

$$\Delta(X_i, T) = f_i \cdot (m(T) - m'(X_i, T)) \quad \textit{para todo } i = 1, \ldots, n$$

Definición 3.4 *Sea X^i el test asociado al atributo X_i con $i = 1, \ldots, n$. Sea $\{T_1, T_2, \ldots, T_{u(i)}, T_{u(i)+1}\}$ la partición del actual conjunto de experiencias. Se define un criterio (con valores conocidos y desconocidos) $m'' : R \times \mathcal{P}(\mathcal{T}) \rightarrow \mathbb{R}^+$ donde*

$$m''(X_i, T) = \sum_{j=1}^{u(i)+1} \frac{|\, T_j \,|}{|\, T \,|} \cdot m(T_j) \quad \textit{para todo } i = 1, \ldots, n$$

siendo $T = T_1 \cup T_2 \cup \cdots \cup T_{u(i)} \cup T_{u(i)+1}$

Finalmente, el criterio de división sigue la idea de [Quinlan, 1992].

Definición 3.5 *Sea X^i el test asociado al atributo X_i con $i = 1, \ldots, n$. Se define el criterio de división (con valores conocidos y desconocidos) y se denota por Δ' como:*

$$\Delta'(X_i, T) = \Delta(X_i, T)/m''(X_i, T) \quad \textit{para todo } i = 1, \ldots, n$$

Supongamos que X_k es el atributo que toma valor máximo en $\{\Delta'(X_i, T) : X_i \in R\}$. Entonces el atributo X_k es seleccionado para expandir el árbol en el nodo actual (esto es, para obtener nuevos nodos hijos y repartir las experiencias).

Ejemplo 3.1 Este ejemplo está tomado de [Quinlan, 1992] y se usará en todo el capítulo. El problema de clasificación que se tiene es el siguiente: se observan algunas características climatológicas y a partir de ellas se decide si jugar o no a un determinado deporte. Sea X_1 el atributo que refleja los estados del cielo, X_2 es el atributo que toma los valores de la temperatura, X_3 el que toma los valores de la humedad, X_4 los del viento, e, Y la clase. En la tabla 3.1 se presenta el conjunto de experiencias $\mathcal{T}$. Para cada atri-

$\mathcal{T}$	$X_1(e)$	$X_2(e)$	$X_3(e)$	$X_4(e)$	$Y(e)$
e_1	soleado	75	70	sí	jugar
e_2	soleado	80	90	sí	no jugar
e_3	soleado	85	85	no	no jugar
e_4	soleado	72	95	no	no jugar
e_5	soleado	69	70	no	jugar
e_6-	?	72	90	sí	jugar
e_7	cubierto	83	78	no	jugar
e_8	cubierto	64	65	sí	jugar
e_9	cubierto	81	75	no	jugar
e_{10}	lluvioso	71	80	sí	no jugar
e_{11}	lluvioso	65	70	sí	no jugar
e_{12}	lluvioso	75	80	no	jugar
e_{13}	lluvioso	68	80	no	jugar
e_{14}	lluvioso	70	96	no	jugar

Tabla 3.1: Conjunto de experiencias $\mathcal{T}$

buto X_i se eligen los tests X^i . Suponemos que para X_1 la partición dada por X^1 es θ_1 =soleado, θ_2 =cubierto, θ_3=lluvioso. En adelante denotamos indistintamente "soleado" o "sol", "cubierto" o "cub" y "lluvioso" o "ll". También en adelante denotamos indistintamente "jugar" o "j" y "no jugar" o "no j". Suponemos que la medida m es la razón de ganancia. Entonces:

$$m(T) = -\frac{8}{13}\log_2\left(\frac{8}{13}\right) - \frac{5}{13}\log_2\left(\frac{5}{13}\right) = 0,961 \quad \text{bits}$$

$$
\begin{aligned}
m'(X_1,T) &= \frac{5}{13}\left[\frac{-2}{5}\log_2\left(\frac{2}{5}\right)-\frac{3}{5}\log_2\left(\frac{3}{5}\right)\right]+ \\
&+ \frac{3}{13}\left[\frac{-3}{3}\log_2\left(\frac{3}{3}\right)-\frac{0}{3}\log_2\left(\frac{0}{3}\right)\right]+ \\
&+ \frac{5}{13}\left[\frac{-3}{5}\log_2\left(\frac{3}{5}\right)-\frac{2}{5}\log_2\left(\frac{2}{5}\right)\right] = 0,747 \quad \text{bits}
\end{aligned}
$$

$$
\Delta(X_1,T) = \frac{13}{14}(0,961-0,747) = 0,199 \quad \text{bits}
$$

$$
\begin{aligned}
m''(X_1,T) &= \frac{-5}{14}\log_2\left(\frac{5}{14}\right)-\frac{3}{14}\log_2\left(\frac{3}{14}\right)- \\
&- \frac{5}{14}\log_2\left(\frac{5}{14}\right)-\frac{1}{14}\log_2\left(\frac{1}{14}\right) = 1,809 \quad \text{bits}
\end{aligned}
$$

$$
\Delta'(X_1,T) = \frac{0,199}{1,809} = 0,110 \quad \text{bits}
$$

Para los demás atributos el proceso se hace de igual forma y suponemos que X_1 hace máxima a $\{\Delta'(X_i,T) : X_i \in R\}$. Entonces el siguiente paso es la partición del conjunto de experiencias. ■

Después de seleccionar el atributo X_i para expandir un nodo del árbol de decisión actual, se genera un hijo por cada valor conocido del atributo seleccionado.

Antes de repartir las experiencias en cada subconjunto de la partición asignamos valores a los valores desconocidos. Las experiencias que tengan valores conocidos en el atributo X_i seleccionado, se reparten entre los nodos hijos. A las experiencias que tengan valores desconocidos en X_i les asignaremos valores y luego distribuiremos las experiencias en cada subconjunto de la partición.

3.2 Asignación de valores múltiples

El proceso que vamos a describir ahora, se realiza en la construcción del árbol en cada nodo donde haya un conjunto de experiencias T que tenga

experiencias con valores desconocidos. Para asignar los valores tenemos en cuenta la información de los valores de ese atributo y los valores de la clase.

En la asignación de valores generalizamos el método propuesto por C4.5 en el siguiente sentido. Consideramos el atributo X_i que tiene valor desconocido en la experiencia $e-$ como una variable aleatoria simple que toma los valores $\Omega = \{\theta_j : j = 1, ..., u(i)\}$. También consideramos el atributo Y como una variable aleatoria que toma los valores $\{y_k : k = 1, \ldots, h\}$.

Sobre los posibles valores del atributo tenemos una información a priori que nos la proporciona la frecuencia de los valores del atributo en el conjunto de experiencias del nodo actual. Consideramos estas frecuencias como las probabilidades a priori $p(X_i(e) \in \theta_j, e \neq e-) = p(\theta_j)$ con $j = 1, \ldots, u(i)$. El método propuesto en C4.5 asigna estas probabilidades a los valores desconocidos indicando con ello la creencia de que el atributo pueda tomar cada uno de esos valores.

Nosotros tenemos en cuenta información adicional para asignar valores a los atributos con valores desconocidos. Para obtener esta información adicional nos fijamos en el atributo clase Y que también lo podemos considerar como una variable aleatoria simple. La justificación de la elección de este atributo es que la tarea de clasificación intenta descubrir cómo los valores de los atributos X_i influyen en los valores de la clase Y. Nosotros recogemos esta información adicional considerando para cada $k = 1, \ldots, h$ las siguientes probabilidades $p(Y(e) = y_k \mid X_i(e) \in \theta_j) \quad j = 1, ..., u(i)$.

Aplicamos el teorema de Bayes y obtenemos para cada $j = 1, \ldots, u(i)$ las probabilidades a posteriori $p(X_i(e) \in \theta_j \mid Y(e) = y_k)$ para cada $k = 1, \ldots, h$. Estas probabilidades a posteriori indicarán la creencia que tenemos acerca de que el atributo X_i tome esos valores dado el valor y_k de la clase con $k = 1, \ldots, h$.

A cada experiencia con valores desconocidos le asociamos un vector con h componentes donde cada componente es la probabilidad a posteriori correspondiente.

Explicamos ahora por qué nuestro método generaliza a C4.5. Fijado el valor θ_j de X_i y el valor y_k de la clase Y, si consideramos la distribución uniforme

$$p(Y(e) = y_k \mid X_i(e) \in \theta_j) = \frac{1}{u(i)}$$

entonces por el teorema de Bayes

$$p(X_i(e) \in \theta_j \mid Y(e) = y_k) = \frac{p(Y(e) = y_k \mid X_i(e) \in \theta_j) \cdot p(\theta_j)}{\sum_{\ell=1}^{u(i)} p(Y(e) = y_k \mid X_i(e) \in \theta_\ell) \cdot p(\theta_\ell)}$$

tenemos entonces que $p(X_i(e) \in \theta_j \mid Y(e) = y_k) = p(\theta_j)$.

Ejemplo 3.2 Volvamos a los datos del ejemplo 3.1. El primer atributo seleccionado para expandir el primer nodo del árbol de decisión fue X_1. El test X^1 para el atributo X_1 nos dio la partición más fina. La experiencia e_6- tiene un valor desconocido en X_1. Entonces asignamos valores al valor desconocido calculando las probabilidades a posteriori.

Sea p(sol)=5/13, p(cub)=3/13, p(ll)=5/13 la información a priori.

Las probabilidades a posteriori se encuentran en la siguiente tabla

p(sol | j)=0,25	p(sol | no j)=0,6
p(cub| j)=0,375	p(cub| no j)=0
p(ll | j)=0,375	p(ll | no j)=0,4

La probabilidad p(sol | j)=0,25 se obtiene de esta forma:

$$p(\text{sol} \mid j) = \frac{p(\text{j} \mid \text{sol}) \cdot p(\text{sol})}{p(\text{j} \mid \text{sol}) \cdot p(\text{sol}) + p(\text{j} \mid \text{cub}) \cdot p(\text{cub}) + p(\text{j} \mid \text{ll}) \cdot p(\text{ll})}$$

$$p(\text{sol} \mid j) = \frac{\frac{2}{5} \cdot \frac{5}{13}}{\frac{2}{5} \cdot \frac{5}{13} + 1 \cdot \frac{3}{13} + \frac{3}{5} \cdot \frac{5}{13}}$$

■

3.3 Clasificar experiencias y asignar parámetros

Sea X^i el test asociado a X_i que realiza la partición $\{\theta_1, \theta_2, .., \theta_{u(i)}\}$ en los valores de X_i. Además, X_i maximiza a $\{\Delta'(X_k, T) : X_k \in R\}$. Entonces, X_i es el atributo que etiqueta al nodo actual. Ahora, hacemos la partición del actual conjunto de experiencias T. Este conjunto T se divide en $\{T_1, \ldots, T_{u(i)}\}$.

Ahora, a cada experiencia le asociamos un vector con h componentes (una por cada clase) y se incluyen en el correspondiente correspondiente T_j de la siguiente manera:

- Si el valor del atributo X_i en la experiencia es conocido, entonces e es incluida en el correspondiente T_j y, en el vector asociado a e, la clase de e tiene asociada probabilidad 1. Las componentes de las otras clases tienen asociada probabilidad 0.

- Si el valor del atributo X_i en la experiencia es desconocido, entonces $e-$ se incluye en cada T_j. En el conjunto $T_j = \{e \in T : X_i(e) \in \theta_j\}$, la experiencia $e-$ tiene asociado el vector donde sus componentes son las probabilidades $p(X_i(e) \in \theta_j \mid Y(e) = y_k)$ para $k = 1, \ldots, h$.

En la construcción del árbol, las experiencias irán descendiendo por las ramas del árbol y se irán distribuyendo por los correspondientes T_j. Entonces las probabilidades se acumularán multiplicando.

En el árbol de decisión que se va generando cada hoja tiene asociado el vector (N/E) donde:

- N es el número de experiencias que están en esa hoja (N es la suma de las probabilidades de las experiencias que están en esa hoja).

- E es un vector de dimensión h donde cada componente es el número de experiencias de cada clase (E_k es la suma de las probabilidades de

las experiencias asociadas a la clase y_k). La clase que etiqueta la hoja es la clase que tiene mayor valor en E_k.

Ejemplo 3.3 Consideremos de nuevo el ejemplo 3.1. Una vez seleccionado el atributo X_1, tenemos que expandir el árbol y repartir las experiencias en los tres subconjuntos de la partición del conjunto de experiencias.

$$\begin{aligned} T_1 &= \{e \in T : X_1(e) = \text{sol}\} \\ T_2 &= \{e \in T : X_1(e) = \text{cub}\} \\ T_3 &= \{e \in T : X_1(e) = \text{ll}\} \end{aligned}$$

Las trece experiencias con valor en X_1 conocido son asignadas de esta forma: $T_1 = \{e_1, e_2, e_3, e_4, e_5\}, T_2 = \{e_7, e_8, e_9\}, T_3 = \{e_{10}, e_{11}, e_{12}, e_{13}, e_{14}\}$ La experiencia e_6- aparecerá en cada T_j con $j = 1, 2, 3$.

Cada experiencia tiene asociado un vector con dos componentes (dos probabilidades a posteriori, una por cada clase:"jugar" o "no jugar"). Entonces, cada vector de cada experiencia es de la forma: (p_1, p_2).

Por lo tanto la partición del conjunto de experiencias es:

Para T_1

	$X_3(e)$	$Y(e)$	p_1	p_2
e_1	70	jugar	1	0
e_2	90	no jugar	0	1
e_3	85	no jugar	0	1
e_4	95	no jugar	0	1
e_5	70	jugar	1	0
e_6-	90	jugar	0,25	0,6

En el nodo actual están las experiencias de T_1. En T_1 hay que hacer todo el proceso de nuevo (elegir tests para la partición de los valores de los atributos, aplicar el criterio de división para elegir el atributo por el que clasificar, asignar probabilidades y repartir las experiencias). Supongamos que el atributo elegido es X_3 y el test X^3 divide en humedad ≤ 75

y humedad $>$ 75. Repartimos experiencias y supongamos que los nodos que nos salen son ya hojas. Las reglas inductivas que se generan a partir de ese subárbol son:
cielo=soleado
humedad$\leq$ 75$(N = 2/E = (2;0))$
humedad$>$ 75$(3,25/(0,25;3))$

Para T_2

	$Y(e)$	p_1	p_2
e_6-	jugar	0,375	0
e_7	jugar	1	0
e_8	jugar	1	0
e_9	jugar	1	0

En T_2 también se cumple la condición hoja, por lo que la regla obtenida es:
cielo=cubierto (3,375/(3,375;0))

Para T_3

	$X_4(e)$	$Y(e)$	p_1	p_2
e_6-	sí	jugar	0,375	0,4
e_{10}	sí	no jugar	0	1
e_{11}	sí	no jugar	0	1
e_{12}	no	jugar	1	0
e_{13}	no	jugar	1	0
e_{14}	no	jugar	1	0

Por último en T_3 hay que hacer de nuevo todo el proceso descrito anteriormente para T_1 y supongamos que las reglas inductivas que se derivan son:
cielo=lluvioso
viento=sí $(2,375/(0,375;2))$
viento=no $(3/(3;0))$

Por lo tanto, las reglas inductivas que se derivan del árbol de decisión final son:

cielo=soleado:
  humedad$\leq$ 75: jugar $(N = 2/E = (2; 0))$
  humedad$>$ 75: no jugar $(3,25/(0,25; 3))$
cielo=cubierto: jugar $(3,375/(3,375; 0))$
cielo=lluvioso:
  viento=sí: no jugar $(2,375/(0,375; 2))$
  viento=no: jugar $(3/(3; 0))$

■

3.4 Predicción

El árbol de decisión obtenido se puede usar para predecir la clase de una observación que puede tener valores desconocidos. Entonces existen cuatro perspectivas para la predicción. Se estudian las tres perspectivas con valores desconocidos en los atributos y además se incluyen resultados experimentales en la siguiente sección. En todas estas predicciones se dan parámetros de confianza calculados a partir del número de experiencias asociadas a cada nodo y a partir de las probabilidades acumuladas asociadas a cada nodo del árbol de decisión.

3.4.1 Predicción para observaciones sin valores desconocidos

Explicamos ahora cómo predecir la clase de una observación sin valores desconocidos en árboles de decisión construidos con experiencias con valores desconocidos en los atributos. Supongamos una observación del tipo b (con todos sus valores conocidos). Entonces, b desciende por el árbol por un

único camino. La observación b alcanza una hoja t. Seleccionamos la clase que aparece el mayor número de veces en las experiencias de la hoja t. Supongamos que es por ejemplo y_c. Entonces, predecimos que b pertenece a y_c. Esta predicción se completa con un parámetro de confianza. Decimos que b pertenece a $Y(b) = y_c$ con probabilidad p=(número de experiencias en t clase asociada y_c/número de experiencias en t).

Ejemplo 3.4 Usando el árbol de decisión del ejemplo 3.3, vamos a predecir la clase de varias observaciones, para poner de manifiesto que aunque la observación tenga todos los valores conocidos no tiene por qué predecirse la clase con probabilidad 1. Esto es debido a que el árbol de decisión que usamos para predecir está construido a partir de experiencias con valores desconocidos.

Para la observación b=(soleado, 70, 80, no) la clase que se asigna es $Y(b)$ = "no jugar" con probabilidad 3/3,25=0,92.

Para la observación b=(lluvioso, 68, 81, no) la predicción es $Y(b)$="jugar" con probabilidad 1.

Para el caso b=(cubierto, 70, 65, no), la predicción es $Y(b)$="jugar" con probabilidad 1. ■

Si el árbol de decisión ha sido construido a partir de experiencias con todos los valores conocidos, la observación desciende por un único camino y la probabilidad asociada a la clase elegida se calcula igual que en el caso anterior.

3.4.2 Predicción para observaciones con valores desconocidos

Mostraremos ahora cómo predecir la clase de una observación del tipo $b-$ tanto en el caso de árboles de decisión construidos con experiencias con valores desconocidos en los atributos como en el caso de árboles de decisión

construidos con experiencias sin valores desconocidos en los atributos.

La filosofía para la predicción de la clase de la observación del tipo $b-$ es la misma tanto para árboles de decisión construidos con experiencias del tipo e como para árboles de decisión construidos con experiencias del tipo $e-$. La observación $b-$ comienza a descender por el árbol y en un momento dado la observación $b-$ llega a un nodo que tiene etiquetado el atributo donde $b-$ tiene valor desconocido. Entonces, $b-$ baja por todas las ramas y se sigue el camino de descenso por el árbol con este criterio hasta llegar a las hojas. Esto origina que la observación $b-$ aparezca en varias de las hojas del árbol de decisión. El enfoque de Friedman [Friedman, 1977] asigna la observación $b-$ a la clase con mayor representación en la unión de esas hojas.

Nosotros seguiremos el enfoque de [Quinlan, 1992]. En cada nodo donde la observación $b-$ descienda por todas las ramas, calculamos la proporción de experiencias dirigidas por cada rama de la totalidad de experiencias en el nodo. (Por ejemplo, si el nodo tiene s experiencias y tres ramas con s_1, s_2, s_3 experiencias, respectivamente, la proporción para cada rama es s_i/s con $i = 1, 2, 3$.).

Se calculan las probabilidades de las clases que aparecen en las hojas. Finalmente calculando las probabilidades condicionadas, se calcula la distribución de probabilidad sobre $Y(b-)$. La clase que se predice para $b-$ es la que tenga un valor de probabilidad más alto.

Ejemplo 3.5 Volvemos a usar el árbol de decisión del ejemplo 3.3 para predecir la clase de las siguientes observaciones.

Sea $b-$=(soleado, 70, ?, no). Entonces $X_1(b-)$=sol y $X_3(b-)$= ?. Por tanto,

- Si humedad $\leq$75 entonces $Y(b-)$="jugar" con probabilidad 1 e $Y(b-)$= ="no jugar" con probabilidad 0.
- Si humedad>75, entonces $Y(b-)$="no jugar" con probabilidad 3/3,25

e $Y(b-)$="jugar" con probabilidad 0,25/3,25.

El conjunto de experiencias se divide en 2 y 3,25 experiencias para los valores de X_3=humedad. Por lo tanto:

- $Y(b-)$="jugar" con probabilidad (2/5,25).1+(3,25/5,25).(0,25/3,25)= =0,43
- $Y(b-)$="no jugar" con probabilidad (3,25/5,25).(3/3,25)=0,57

Elegimos por tanto $Y(b-)$="no jugar".

Para la observación $b-$=(?, 72, 90, sí) la predicción es:

- Si es soleado entonces $Y(b-)$="jugar" con probabilidad 0,25/3,25 e $Y(b-)$ = "no jugar" con probabilidad 3/3,25.
- Si es cubierto entonces $Y(b-)$="jugar" con probabilidad 1 e $Y(b-)$= ="no jugar" con probabilidad 0.
- Si es lluvioso, $Y(b-)$="jugar" con probabilidad 0,375/2,375 e $Y(b-)$= ="no jugar" con probabilidad 2/2,375.

La división de las 14 experiencias fue: 5,25, 3,375 y 5,375 para los valores de X_1=cielo. Entonces:

- $Y(b-)$="jugar" con probabilidad (5,25/14).(0,25/3,25)+(3,375/14).1+(5,375/14).(0,375/2,375)=0,33
- $Y(b-)$="no jugar" con probabilidad (5,25/14).(3/3,25)+(3,375/14).0+(5,375/14).(2/2,375)=0,67

Por lo tanto elegimos $Y(b-)$="no jugar" ■

3.5 Resultados experimentales

Todo lo expuesto en este capítulo se puede integrar de forma directa en cualquier software desarrollado en aprendizaje inductivo como software académico [Ramos y Morales, 1999], [Ramos et al., 2000] o software comercial [Quinlan, 1992].

Hemos implementado nuestro método y hemos usado bases de datos estándar para conseguir resultados experimentales. Hemos usado las conocidas bases de datos de Holte [Holte, 1993] que contienen los 16 problemas que Holte usó para comparar su método 1R con el C4.5 de Quinlan [Quinlan, 1992]. Este conjunto de problemas intenta representar problemas de la vida real. Holte exponía que los resultados experimentales obtenidos para estos problemas mediante sistemas de aprendizaje podrían dar luz para la ejecución de cada sistema de aprendizaje en una situación real.

Todos los problemas seleccionados cumplen dos condiciones: representar un problema de la vida real que no haya sido construido de forma artificial; y los ejemplos son descritos mediante atributos usados de forma natural en la vida real. La única base de datos que no cumple con la segunda condición es CH. Esta base de datos representa finales de partida en ajedrez. De acuerdo con Holte esta base de datos está diseñada para adaptarse muy bien al ID3 de Quinlan.

3.5.1 Descripción de las bases de datos

Existen varias versiones de los problemas de Holte. Básicamente, las diferencias están en el número de experiencias usadas o en el número de atributos usados. Nosotros hemos usado la versión almacenada en el "UCI Repository" de Automated Learning Repository in California University, Irvine [Blake et al., 1998].

En la tabla 3.2 presentamos un breve resumen de las características de estas bases de datos: la primera columna muestra el nombre de la base

Prob	Tamaño	Val. Desc.	Clase predom.	Contin	Discr.	N.Clases
BC	286	9	70.3	0	9	2
CH	3196	0	52.2	0	36	2
GL	214	0	35.3	9	0	7
G2	163	0	53.4	9	0	2
HD	303	7	54.5	5	8	2
HE	155	167	79.4	6	13	2
HO	368	1927	63.0	7	15	2
HY	3163	5329	95.2	7	18	2
IR	150	0	33.3	4	0	3
LA	57	326	64.9	8	8	2
LY	148	0	56.7	2	16	4
MU	8124	2480	51.8	0	22	2
SE	3163	5329	90.7	7	18	2
SO	47	0	36.2	0	35	4
VO	435	381	61.4	0	16	2
V1	435	392	61.4	0	16	2

Tabla 3.2: Bases de datos

de datos, la segunda el número de experiencias que tiene, la tercera el número de valores desconocidos, la cuarta el porcentaje de ocurrencia de la clase más frecuente, la quinta y sexta columnas muestran el número de atributos continuos y discretos respectivamente, y la última columna muestra el número de clases diferentes.

Para los experimentos, se han implementado dos algoritmos para construir los árboles de decisión. En el primer algoritmo, las experiencias con valores desconocidos son distribuidas siguiendo la filosofía de C4.5; es decir, la distribución de probabilidad del atributo considerado se usa para distribuir las experiencias con valores desconocidos en ese atributo. En el segundo algoritmo llamado PostP, las experiencias con valores desconocidos se distribuyen teniendo en cuenta la distribución de probabilidad obtenida

usando el atributo y la clase.

En la primera etapa, se usaron las bases de datos originales, y luego se introdujo de forma aleatoria un porcentaje de valores desconocidos (10%, 20%, y 30%, respectivamente). Este proceso de introducir valores desconocidos se hizo 10 veces para cada porcentaje. De esta forma se minimiza la posibilidad de sesgo. Por lo tanto se han hecho 31 experimentos para cada base de datos.

3.5.2 Resultados

Para los 31 árboles de decisión estimados, se ha calculado el error de clasificación. Para cada grupo de árboles de decisión relativos al mismo porcentaje de valores desconocidos se ha calculado el error medio y la desviación típica.

Los resultados obtenidos se muestran en la tabla 3.3. La primera columna muestra el nombre de la base de datos, la segunda el error en la muestra usando C4.5 (primer valor) y el error usando nuestro enfoque PostP (segundo valor). En las columnas 3, 4, y 5, se muestran los resultados obtenidos para las bases de datos con 10%, 20%, y 30% de valores desconocidos, respectivamente. Para cada porcentaje en la primera fila de cada base de datos se muestra el error medio de los experimentos, y en la segunda fila aparece la desviación típica.

En general, el tamaño de los árboles en ambos métodos es similar, salvo en el caso de que el porcentaje de los valores desconocidos sea grande (30%). En ese caso los árboles obtenidos mediante PostP son un poquito mayores que los obtenidos mediante C4.5. De todas formas este resultado es razonable porque el método PostP distribuye las experiencias usando los valores del atributo y los de la clase. Sin embargo, en el caso de los árboles grandes obtenidos mediante PostP, el error es substancialmente menor. El error medio para todos los experimentos con el 30% de valores desconocidos para C4.5 es 11.67 y con PostP es 7.82. Por tanto nuestro método es un

Nombre		Original C4.5-PostP	10% C4.5-PostP	20% C4.5-PostP	30% C4.5-PostP
BC	(error)	12.24-11.54	9.16-8.64	12.97-11.50	15.87-13.95
	(std d)	0.0-0.0	1.17-1.14	0.86-1.22	1.33-1.37
CH	(error)	0.38-0.38	3.00-2.84	6.74-5.45	11.92-9.27
	(std d)	0.0-0.0	0.18-0.17	0.31-0.25	0.40-.023
GL	(error)	6.07-6.07	11.78-10.14	17.24-13.55	28.08-18.64
	(std d)	0.0-0.0	2.70-2.29	2.12-1.77	1.54-2.90
G2	(error)	7.97-7.97	11.41-10.61	15.77-12.82	18.83-14.48
	(std d)	0.0-0.0	1.68-1.45	2.69-2.15	2.26-2.07
HD	(error)	4.29-3.63	7.03-6.53	10.33-8.15	12.61-10.66
	(std d)	0.0-0.0	0.77-1.18	1.84-1.43	1.40-1.63
HE	(error)	1.94-2.58	6.84-4.84	7.48-5.55	10.32-7.55
	(std d)	0.0-0.0	1.79-0.88	0.76-1.09	1.09-1.52
HO	(error)	9.24-5.98	10.22-7.55	11.85-9.27	13.64-12.04
	(std d)	0.0-0.0	1.29-0.65	1.00-1.01	0.92-0.97
HY	(error)	0.35-0.35	0.74-0.65	1.13-0.94	1.62-1.19
	(std d)	0.0-0.0	0.08-0.06	0.12-0.11	0.16-0.11
IR	(error)	2.67-2.67	3.80-3.67	4.80-4.00	9.27-6.00
	(std d)	0.0-0.0	0.73-0.77	1.52-0.98	1.48-1.69
LA	(error)	3.51-1.75	9.65-4.03	21.93-3.86	24.21-7.72
	(std d)	0.0-0.0	8.14-2.14	7.03-0.96	6.66-2.51
LY	(error)	6.08-6.08	4.86-4.32	6.82-4.86	10.74-6.62
	(std d)	0.0-0.0	1.17-1.14	1.78-1.14	2.05-1.59
MU	(error)	0.0-0.0	0.17-0.21	0.65-0.40	1.55-0.78
	(std d)	0.0-0.0	0.05-0.02	0.06-0.05	0.12-0.05
SE	(error)	1.01-1.01	2.24-2.10	3.64-3.32	4.75-4.39
	(std d)	0.0-0.0	0.23-0.21	0.22-0.26	0.30-0.34
SO	(error)	0.0-0.0	2.55-1.06	5.53-0.64	10.0-1.91
	(std d)	0.0-0.0	1.43-1.35	3.14-0.94	3.36-1.21
VO	(error)	1.84-2.30	2.62-2.62	3.77-3.22	5.70-4.18
	(std d)	0.0-0.0	0.46-0.51	0.46-0.48	0.48-0.45
V1	(error)	4.60-4.37	4.71-4.07	6.23-5.03	7.56-5.77
	(std d)	0.0-0.0	0.65-0.46	0.57-0.28	0.85-0.76

Tabla 3.3: Resultados

33% mejor con respecto a C4.5.

Podemos concluir que nuestros resultados son apreciablemente mejores en muchos casos cuando se usa el algoritmo PostP.

Para varias bases de datos el error decrece hasta el 80%. Por ejemplo, con la base de datos SO, el error medio con C4.5 es 4.52 y con nuestro método es 0.90.

Queremos hacer notar que los experimentos con la base de datos LA han dado muy buenos resultados: en los experimentos con un 30% de valores desconocidos, el error se reduce en más del 69%. El mejor resultado obtenido con el algoritmo PostP con respecto a C4.5, es con la base de datos SO.

Resumiendo, C4.5 obtiene mejores resultados únicamente en el 6% de todos los experimentos. En el 3% de los casos los resultados son similares para ambos métodos, en el 82% nuestro método mejora la clasificación y el restante 15%, pertenece a los experimentos sin valores desconocidos.

3.5.3 Otras bases de datos

También hemos llevado a cabo otros experimentos con otras bases de datos tomadas del MLRepository [Blake et al., 1998]. Se eligieron bases de datos que tuvieran valores desconocidos. En la tabla 3.4 se muestra un breve resumen de sus características.

Los resultados que se han obtenido y que se muestran en la tabla 3.5 son similares a las bases de datos previas. En todos los casos con valores desconocidos el error con PostP es menor que el error con C4.5. Además, si existe un alto porcentaje de valores desconocidos el error con PostP decrece más rápido que con C4.5. También la desviación estándar con nuestro método es menor que con C4.5.

Prob	Tamaño	Desco.	Contin	Disc.	N.Clas
Adult	3000	759	6	8	2
Bands	400	986	20	17	2
Dermatology	250	1	34	2	6
Horse-colic	250	67	10	17	2
House-votes	315	395	0	17	2
Post-operative	70	3	0	9	3

Tabla 3.4: Otras bases de datos

3.6 Problema de decisión con riesgo

Generalizamos ahora el método C4.5 usando nuestro método PostP para asignar valores desconocidos y además asignaremos un parámetro de error a cada clase en cada asignación de valores que hagamos. Modelamos el problema como un problema de decisión con riesgo.

Supongamos la experiencia $e-$ que tiene valor desconocido en el atributo X_i, es decir $X_i(e-) = ?$. Identificamos el espacio paramétrico Λ con los valores del atributo, esto es, $\Lambda = \{\theta_j \quad j = 1, ..., u(i)\}$. La decisión que hay que tomar es el valor del atributo X_i en $e-$. Por tanto el conjunto de acciones A coincide con el espacio paramétrico.

Para calcular el riesgo de Bayes son necesarias las probabilidades a posteriori y las pérdidas. Para calcular las probabilidades a posteriori se usa el mismo procedimiento que explicamos anteriormente. A partir de la información a priori $p(X_i(e) \in \theta_j, e \neq e-) = p(\theta_j)$ y de $p(Y(e) = y_k \mid X_i(e) \in \theta_j) \quad j = 1, ..., u(i), \quad k = 1, \ldots, h$, aplicamos el teorema de Bayes y obtenemos las probabilidades a posteriori $p(X_i(e) \in \theta_j \mid Y(e) = y_k)$.

La decisión sobre el valor del atributo tiene asociado un riesgo que indica la pérdida ponderada causada por esta decisión. Se intentan minimizar las pérdidas. Estas pérdidas se representan en una matriz llamada la matriz de pérdidas. Cada elemento de la matriz de pérdidas mide nuestra creencia

Nomb.		Original C4.5-PostP	10% C4.5-PostP	20% C4.5-PostP	30% C4.5-PostP
Adult	(error)	7.00-7.23	9.25-8.72	11.43-10.33	13.88-12.57
	(std d)	0.00-0.00	0.37-0.28	0.62-0.39	0.32-0.36
Bands	(error)	4.25-4.25	5.95-5.10	10.42-8.40	15.47-12.22
	(std d)	0.00-0.00	0.63-0.63	0.97-1.24	0.77-0.82
Dermat.	(error)	2.00-2.00	2.68-2.64	4.96-4.32	9.00-5.36
	(std d)	0.00-0.00	0.57-1.35	0.76-1.08	1.22-1.31
Horse.	(error)	6.05-6.85	13.87-8.31	18.47-12.06	23.27-15.93
	(std d)	0.00-0.00	2.04-1.33	2.89-1.84	3.78-1.15
House.	(error)	1.90-1.90	2.29-1.94	3.56-2.57	4.70-3.27
	(std d)	0.00-0.00	0.55-0.31	0.59-0.25	0.68-0.63
Post.	(error)	14.29-14.29	15.29-15.57	18.86-17.57	21.43-20.00
	(std d)	0.00-0.00	2.05-2.86	1.32-1.65	2.41-2.41

Tabla 3.5: Más resultados

(un número real en $[0, 1]$) sobre lo lejos o cerca que el valor asignado está del verdadero valor del atributo. Esta idea está relacionada con el concepto de distancia. Para calcular la matriz de pérdidas tendremos en cuenta todos los tipos de atributos. Clasificamos los atributos en atributos cualitativos (ordenados o no ordenados) y atributos cuantitativos (continuos o discretos).

Consideramos que tenemos dos conjuntos de valores θ_i y θ_j. Considerando todos los tipos de atributos, definimos una función de pérdidas como sigue:

$$L(\theta_i, \theta_j) = \begin{cases} d(\theta_i, \theta_j) & \text{cuant.} \\ 1/u(i) & \text{cual. no ord. y } \theta_i \cap \theta_j = \emptyset \\ 0 & \text{cual. no ord. y } \theta_i \cap \theta_j \neq \emptyset \\ \mid ord(i) - ord(j) \mid /u(i) & \text{cual. ord. y } \theta_i \cap \theta_j = \emptyset \\ 0 & \text{cual. ord. y } \theta_i \cap \theta_j \neq \emptyset \end{cases}$$

donde $ord(i)$ es la posición ordinal del valor θ_i.

Por último, es necesario definir la distancia entre dos conjuntos cuando el atributo es cuantitativo, es decir, $d(\theta_i, \theta_j)$. En [Wilson y Martinez, 1997] se pueden encontrar varias definiciones para calcular distancias entre valores del atributo. Nosotros hemos adaptado y normalizado estas funciones distancias a nuestro caso para medir distancias entre conjuntos de valores de un atributo.

Posibles definiciones de la función d son:

$$d_1(\theta_i, \theta_j) = \frac{|| \theta_i, \theta_j ||}{\sum_{k=1, k\neq i}^{n(i)} || \theta_i, \theta_k ||}$$

$$d_2(\theta_i, \theta_j) = \frac{||\theta_i, \theta_j||^2}{\sum_{k=1, k\neq i}^{n(i)} ||\theta_i, \theta_k||^2}$$

donde $||\theta_i, \theta_j||$ se define como

$$||\theta_i, \theta_j|| = min_{o\in\theta_i, o'\in\theta_j} |o - o'|$$

Ejemplo 3.6 Volvamos a los datos del ejemplo 3.1. El primer atributo seleccionado para expandir el primer nodo del árbol de decisión fue X_1. El test X^1 para el atributo X_1 nos dio la partición más fina. Las probabilidades a posteriori para la experiencia e_6- ya se han calculado anteriormente. Tenemos un atributo cualitativo ordenado con lo que definimos la matriz de pérdidas como aparece en la tabla 3.6. Por lo tanto los riesgos de Bayes

A | Ω	θ_1 =sol	θ_2 =cub	θ_3 =ll
sol	0	1/3	2/3
cub	1/3	0	1/3
ll	2/3	1/3	0

Tabla 3.6: La matriz de pérdidas

asociados a estas decisiones calculados a partir de la tabla 3.6 son:

$$\begin{aligned}
\text{R(sol | j)} &= \text{p(sol | j)L(sol,sol)} + \\
&+ \text{p(cub | j) L(sol,cub)} + \\
&+ \text{p(ll | j)L(sol,ll)} = 0,375 \\
\text{R(cub | j)} &= 0,20 \\
\text{R(ll | j)} &= 0,29 \\
\text{R(sol | no j)} &= 0,26 \\
\text{R(cub | no j)} &= 0 \\
\text{R(ll | no j)} &= 0,4
\end{aligned}$$

■

En un paso intermedio de la construcción del árbol se asocia cada experiencia al correspondiente T_j de la siguiente manera:

- Si el valor del atributo X_i en la experiencia es conocido, entonces e se incluye en el correspondiente T_j y en el vector asociado a e la clase de e tiene asociada probabilidad 1 y riesgo 0. Las componentes de las otras clases tienen asociado probabilidad 0 y riesgo 0.
- Si el valor del atributo X_i en la experiencia es desconocido, entonces $e-$ se incluye en cada T_j y se le asocia en cad T_j un vector. En el conjunto $T_j = \{e \in T \,/\, X_i(e) \in \theta_j\}$, la experiencia $e-$ tiene asociado el vector donde sus componentes son las probabilidades $p(X_i(e) \in \theta_j \mid Y(e) = y_k)$ para $k = 1, \ldots, h$ y los riesgos $R(\theta_j \mid y_k)$ para $k = 1, \ldots, h$.

Cuando clasificamos por el atributo X_i, las experiencias se incluyen en el T_j apropiado y se multiplican las probabilidades y se suman los riesgos.

En el árbol de decisión que se va generando cada hoja tiene asociado el vector $(N/E/R)$ donde

- N y E se construyen de la misma forma que explicamos anteriormente.

- R es un vector de dimensión h correspondiente a los riesgos asociados a cada clase.

Ejemplo 3.7 Consideremos de nuevo nuestro ejemplo. Los valores del atributo X_1 dados por X^1 inducen la partición T_1, T_2 y T_3 en el actual conjunto de experiencias T.

Las trece experiencias restantes con valor en X_1 conocido son asignadas de esta forma:
$T_1 = \{e_1, e_2, e_3, e_4, e_5\}, T_2 = \{e_7, e_8, e_9\}, T_3 = \{e_{10}, e_{11}, e_{12}, e_{13}, e_{14}\}$

Cada experiencia tiene asociado un vector con cuatro componentes (dos probabilidades y dos riesgos, uno por cada clase: "jugar" o "no jugar"). Entonces, cada vector de cada experiencia es de la forma: (p_1, p_2, R_1, R_2).

Por lo tanto la partición del conjunto de experiencias es:
Para T_1

	$X_3(e)$	$Y(e)$	p_1	p_2	R_1	R_2
e_1	70	jugar	1	0	0	0
e_2	90	no jugar	0	1	0	0
e_3	85	no jugar	0	1	0	0
e_4	95	no jugar	0	1	0	0
e_5	70	jugar	1	0	0	0
e_6-	90	jugar	0,25	0,6	0,375	0,26

cielo=soleado
humedad$\leq$ $75(N = 2/E = (2;0)/R = (0;0))$
humedad> $75(3,25/(0,25;3)/(0,375;0,26))$
Para T_2

	$Y(e)$	p_1	p_2	R_1	R_2
e_6-	jugar	0,375	0	0,2	0,33
e_7	jugar	1	0	0	0
e_8	jugar	1	0	0	0
e_9	jugar	1	0	0	0

cielo=cubierto (3,375/(3,375;0)/(0,2;0,33))
Para T_3

	$X_4(e)$	$Y(e)$	p_1	p_2	R_1	R_2
e_6-	sí	jugar	0,375	0,4	0,29	0,4
e_{10}	sí	no jugar	0	1	0	0
e_{11}	sí	no jugar	0	1	0	0
e_{12}	no	jugar	1	0	0	0
e_{13}	no	jugar	1	0	0	0
e_{14}	no	jugar	1	0	0	0

cielo=lluvioso
viento=sí $(2,375/(0,375;2)/(0,29;0,4))$
viento=no $(3/(3;0)/(0;0))$

Las reglas inductivas que se derivan del árbol de decisión final son:

cielo=soleado:
  humedad$\leq$ 75: jugar $(N = 2/E = (2;0)/R = (0;0))$
  humedad$>$ 75: no jugar $(3,25/(0,25;3)/(0,375;0,26))$
cielo=cubierto: jugar $(3,375/(3,375;0)/(0,2;0,33))$
cielo=lluvioso:
  viento=sí: no jugar $(2,375/(0,375;2)/(0,29;0,4))$
  viento=no: jugar $(3/(3;0)/(0;0))$

■

3.7 Predicción

El árbol de decisión obtenido se puede usar para predecir la clase de una observación que puede tener valores desconocidos. En todas estas predicciones se dan parámetros de confianza y de error calculados a partir de

las probabilidades y riesgos acumulados asociados a cada hoja del árbol de decisión.

3.7.1 Predicción para observaciones sin valores desconocidos

En el caso de árboles de decisión construido a partir de experiencias con valores desconocidos, la predicción se realiza de la misma forma que hemos explicado antes. Pero ahora añadimos a la predicción el riesgo asociado a la clase mayoritaria.

Ejemplo 3.8 Usando el árbol de decisión del ejemplo 3.7, la clase que se predice para la observación b=(soleado, 70, 80, no) es $Y(b)$ = "no jugar" con probabilidad 3/3,25=0,92 y riesgo 0,26.

Para el ejemplo b=(lluvioso, 68, 81, no) la predicción es $Y(b)$="jugar" con probabilidad 1 y riesgo 0.

y para el caso b=(cubierto, 70, 65, no), la predicción $Y(b)$="jugar" con probabilidad 1 y riesgo 0,2. ■

Para el caso en que los árboles de decisión se hayan construido con experiencias sin valores desconocidos la predicción tendrá asociada una probabilidad que se calculará igual que se ha explicado antes y riesgo 0 asociado.

3.7.2 Predicción para observaciones con valores desconocidos

Para este caso, la probabilidad asociada a la predicción se calcula de la misma forma que antes. En la predicción, a cada clase se le asocia un riesgo que se calcula sumando para cada camino del árbol que lleva a las hojas donde ha llegado la observación los riesgos asociados a la clase.

Ejemplo 3.9 Volvemos a usar el árbol de decisión del ejemplo 3.7 para predecir la clase de las siguientes observaciones.

Sea $b-$=(soleado, 70, ?, no) entonces $X_1(b-)$=sol y $X_3(b-)$= ?. Por tanto:

- Si humedad $\leq$75 entonces $Y(b-)$="jugar" con probabilidad 1 y riesgo 0 e $Y(b-)$="no jugar" con probabilidad 0 y riesgo 0.
- Si humedad>75, entonces $Y(b-)$="no jugar" con probabilidad 3/3,25 y riesgo 0,26 e $Y(b-)$="jugar" con probabilidad 0,25/3,25 y riesgo 0,375.

El conjunto de experiencias se divide en 2 y 3,25 experiencias para los valores de X_3=humedad. Por lo tanto

- $Y(b-)$="jugar" con probabilidad (2/5,25).1+(3,25/5,25).(0,25/3,25)= =0,43 y riesgo 0,375
- $Y(b-)$="no jugar" con probabilidad (3,25/5,25).(3/3,25)=0,57 y riesgo 0,26

Para la observación $b-$=(?, 72, 90, sí) la predicción es:

- Si es soleado entonces $Y(b-)$="jugar" con probabilidad 0,25/3,25 y riesgo 0,375 e $Y(b-)$ = "no jugar" con probabilidad 3/3,25 y riesgo 0,26.
- Si es cubierto entonces $Y(b-)$="jugar" con probabilidad 1 y riesgo 0,2 e $Y(b-)$="no jugar" con probabilidad 0 y riesgo 0,33.
- Si es lluvioso, $Y(b-)$="jugar" con probabilidad 0,375/2,375 y riesgo 0,29 e $Y(b-)$="no jugar" con probabilidad 2/2,375 y riesgo 0,4.

La división de las 14 experiencias fue: 5,25, 3,375 y 5,375 para los valores de X_1=cielo. Entonces:

- $Y(b-)$="jugar" con probabilidad
  (5,25/14).(0,25/3,25)+(3,375/14).1+(5,375/14).(0,375/2,375)=0,33
  y riesgo 0,375+0,2+0,29=0,865

- $Y(b-)$="no jugar" con probabilidad
  (5,25/14).(3/3,25)+(3,375/14).0+(5,375/14).(2/2,375)=0,67
  y riesgo 0,26+0,33+0,4=0,99.

■

Capítulo 4

Inferencia de Gramáticas

Supongamos que tenemos una muestra de palabras que provienen de un determinado lenguaje junto a otras palabras que no pertenecen a ese lenguaje. El problema en el que se centra este capítulo consiste en inferir una gramática que represente el lenguaje del que provienen todas las palabras de la muestra.

Este problema se encuadra dentro del aprendizaje supervisado, sin conocimiento adicional, con modelo y aproximado. Es supervisado porque conocemos qué palabras de la muestra pertenecen al lenguaje y cuáles no. Sólo tenemos como información esa muestra. El modelo que supondremos será el de una gramática regular probabilística o difusa y desarrollamos los algoritmos de inferencia para obtener las reglas de la gramática. Lo importante es que el método produzca un modelo que refleje la estructura inherente de la muestra. A menudo la muestra no refleja todos los posibles elementos que pueden ser generados, debido a que pueda resultar difícil y costoso recoger los datos de la muestra, además de que los datos con los que se cuente sean limitados.

Se presentan dos versiones del algoritmo de inferencia: una probabilística y otra difusa. La versión probabilística se basa en la frecuencia de

aparición de los elementos en las palabras suministradas. Esas frecuencias se analizan en el algoritmo de inferencia y se asignan unas probabilidades a las reglas de la gramática regular inferida. Además, el estudio se completa obteniendo la cadena de Markov asociada a la gramática inferida, para estudiar propiedades estadísticas relativas al lenguaje.

La versión difusa se basa en unos grados de pertenencia que asociamos, de una forma subjetiva, a los elementos del alfabeto que aparecen en las palabras de la muestra. Estos grados se aplicarán mediante relaciones difusas.

Definición 4.1 *Sea* $\Sigma = \{e_1, \ldots, e_n\}$ *el alfabeto. Sea* $\mathcal{L}$ *un lenguaje en* Σ^*. *Sea* $\mathcal{G} = (V, T, S, r)$ *la gramática que genera el lenguaje* $\mathcal{L}$. *Definimos la función* $f : \Sigma^* \to \{0, 1\}$ *de la siguiente manera:*

Si $s \in \mathcal{L} \subseteq \Sigma^*$ *entonces* $f(s) = 1$ *y lo representamos por el par* $(s, 1)$.

Si $s \in \Sigma^* - \mathcal{L}$, *entonces* $f(s) = 0$ *y lo representamos por el par* $(s, 0)$.

Definición 4.2 *Definimos el conjunto formado por todas las palabras que pertenecen a* $\mathcal{L}$ *y lo denotamos por* $S_{\mathcal{L}}^+$ *donde*

$$S_{\mathcal{L}}^+ = \{(s, 1),\ s \in \mathcal{L}\}$$

Definimos el conjunto formado por todas las cadenas que no pertenecen a $\mathcal{L}$ *y lo denotamos por* $S_{\mathcal{L}}^-$ *donde*

$$S_{\mathcal{L}}^- = \{(s, 0),\ s \in \Sigma^* - \mathcal{L}\}$$

Se define $S_{\mathcal{L}} = S_{\mathcal{L}}^+ \cup S_{\mathcal{L}}^-$.

El problema de la inferencia de gramáticas se enuncia así:

> *"Dado algún subconjunto de* $S_{\mathcal{L}}$ *o todo* $S_{\mathcal{L}}$, *¿se puede inferir una buena gramática* $\mathcal{G}$ *que describa a* $\mathcal{L}$*?"*

Nosotros estudiamos el problema de la inferencia de gramáticas en el caso en que tengamos un subconjunto finito $M = \{(s_k, 1) : 1 \leq k \leq m\}$ de $S_{\mathcal{L}}^{+}$, donde sus elementos pueden ser considerados palabras pertenecientes a un lenguaje regular.

4.1 Inferencia de gramáticas regulares probabilísticas

Desarrollamos ahora la versión probabilística del algoritmo de inferencia, con la que obtendremos una gramática regular probabilística capaz de generar nuevas palabras en el lenguaje. A cada una de estas palabras generadas por la gramática inferida se les asocia una probabilidad de pertenencia al lenguaje. Esta probabilidad se calcula como el producto de las probabilidades de las reglas involucradas en la derivación de la palabra.

El algoritmo se centra, como característica principal, en que la conexión relativa a dos subcadenas de longitud 2 sólo es válida con relación a los ejemplos suministrados; esto es, si en las palabras de la muestra aparecen las subcadenas xy e yz, entonces se permitirá, en principio, la existencia de la subcadena de longitud 3, xyz. Si esta subcadena no aparece en las palabras, entonces hay que eliminar esa sobreconectividad.

El algoritmo genera un autómata que posteriormente es transformado en una gramática regular. A partir de las frecuencias de aparición de las subcadenas en la muestra, se asignarán probabilidades a las reglas de la gramática regular obtenida.

El algoritmo consta de 6 pasos que son descritos a continuación:

1. Construcción de dos tablas de probabilidades de transición.
2. Construcción del grafo de sucesos G.
3. Destrucción de la sobreconectividad.

4. Construcción del dual G'.

5. Transformación del autómata G' en la gramática regular.

6. Asignación de probabilidades a las reglas.

1. Construcción de dos tablas de probabilidades de transición

En la primera tabla aparecen las probabilidades de transición entre los elementos del alfabeto y en la segunda tabla aparecen las probabilidades de transición desde las cadenas de longitud 2 a los elementos del alfabeto.

1. Para la primera tabla, calculamos a partir de $M \subset S_{\mathcal{L}}^{+}$, la frecuencia de que $e_i \in \Sigma$ venga seguido de $e_j \in \Sigma$ con $i = 1, \ldots, n$ y $j = 1, \ldots, n$.

   En el caso de que e_i sea el último elemento en alguna palabra de M, esas apariciones no se tienen en cuenta en el cómputo de casos posibles.

2. Para hallar la segunda tabla de probabilidades de transición, se toman las componentes distintas de cero de la tabla anterior y se calculan a partir de $M \subset S_{\mathcal{L}}^{+}$, la frecuencia de que esas cadenas de longitud 2 vengan seguidas de un elemento del alfabeto e_i, con $i = 1, \ldots, n$.

   Como en el caso anterior, si la cadena de sucesos de longitud 2 considerada se encuentra al final en alguna palabra de M, esas apariciones no se tienen en cuenta en el cómputo de casos posibles.

Ejemplo 4.1 Damos un ejemplo sencillo donde la muestra está formada por una sola palabra

$$ABDBCABDBCACBD$$

El alfabeto en este caso está formado por los elementos que aparecen en la palabra, es decir $\Sigma = \{A, B, C, D\}$. Las tablas de probabilidad de

transición son:

Tabla 1	A	B	C	D
A	0	2/3	1/3	0
B	0	0	2/5	3/5
C	2/3	1/3	0	0
D	0	1	0	0

Tabla 2	A	B	C	D
AB	0	0	0	1
AC	0	1	0	0
BC	1	0	0	0
BD	0	1	0	0
CA	0	1/2	1/2	0
CB	0	0	0	1
DB	0	0	1	0

■

2. Construcción del grafo de sucesos G

Fijado el nivel de confianza, construimos el grafo G a partir de la tabla 1 de la siguiente forma:

1. Creamos un vértice por cada símbolo del alfabeto y lo etiquetamos con ese símbolo.

2. Si la probabilidad para la cadena $e_i e_j$ es mayor que el nivel de confianza fijado creamos un arco que va desde el símbolo e_i al e_j. Los arcos se etiquetan con números naturales consecutivos.

Al construir el grafo G, los circuitos que aparecen debido a las cadenas de la muestra deben mantenerse, permaneciendo los arcos de transición relativos a ese circuito en la misma cara del grafo.

Ejemplo 4.2 Si consideramos un nivel de confianza de 0,25, la figura 4.1 muestra el grafo G relativo al ejemplo 4.1. ■

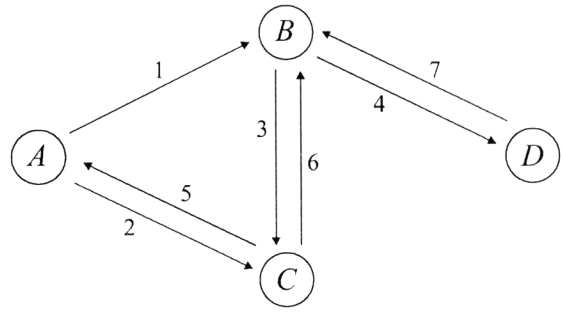


Figura 4.1: Grafo de sucesos G

3. Destrucción de la sobreconectividad

El paso anterior puede dar lugar a vértices sobreconectados en el grafo G. Los vértices e_i, e_j y e_p están sobreconectados si existe un camino $e_i e_j e_p$ en G con probabilidad igual a cero en la segunda tabla de probabilidades de transición. Por lo tanto el grafo G produce secuencias ilegales.

Destruimos la sobreconectividad como sigue: El vértice e_j es duplicado en dos vértices, uno de ellos teniendo un arco desde él a un vértice del extremo de la secuencia ilegal y el otro e_j con un arco desde el otro vértice del extremo a él. Para cada vértice duplicado e_j se consideran de nuevo los mismos arcos que existían con el resto de los vértices, excepto con el e_p.

Todos los arcos son etiquetados con el mismo número que tenían. Se repite este proceso hasta obtener en G todas las secuencias legales y ninguna secuencia ilegal.

Ejemplo 4.3 Supongamos que tenemos un grafo de sucesos G con secuencias legales: AB, BC, BD y ABD y con la secuencia ilegal: ABC. Entonces para destruir esa secuencia ilegal tenemos que duplicar el nodo B. Por lo tanto en todos los sitios de G donde aparezca la secuencia ilegal consideramos el subgrafo correspondiente a los vértices A, B, C y D y actuamos de la siguiente forma. Eliminamos el arco AB, duplicamos B (lo hemos denotado por B' para mayor claridad) y creamos el arco nuevo

$A\,B'$ que aparece representada con línea discontinua en la figura 4.2. A continuación, creamos el arco $B'\,D$ (representada con línea discontinua en la figura 4.2). Estas dos nuevos arcos se crean para seguir representando en G las secuencias $A\,B$, $B\,D$ y $A\,B\,D$ que sí son legales. El último paso sería considerar para el vértice duplicado B' los mismos arcos que existían para B con el resto de los vértices, excepto con el vértice C. ■

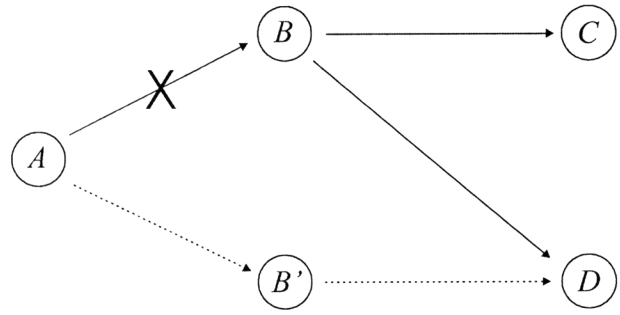


Figura 4.2: Destrucción de sobreconectividad

4. Construcción del dual G'

Cada arco en G es un vértice en G' que está etiquetado con la misma etiqueta del arco de G. Para cada par de arcos de entrada b y de salida f de un vértice e_j en G se crean en G' dos vértices etiquetados con los números de los arcos de entrada y salida de G (b y f) y un arco e_j que va de b a f.

Si un vértice e_i en G sólo tiene arcos de salida b, entonces se crea un arco en G' que va desde un vértice etiquetado con A_0 (estado inicial) al vértice b de G'. El arco en G' se etiqueta con e_i.

Si un vértice e_p en G sólo tiene arcos de entrada f se crea en G' un vértice f y un arco que va desde f hasta un vértice etiquetado con F (estado final). Este arco en G' se etiqueta con e_p.

Con este proceso hemos obtenido un autómata finito determinista.

Ejemplo 4.4 En la figura 4.3 damos un breve esquema de la construcción del grafo dual para los tres casos que se pueden presentar. ■

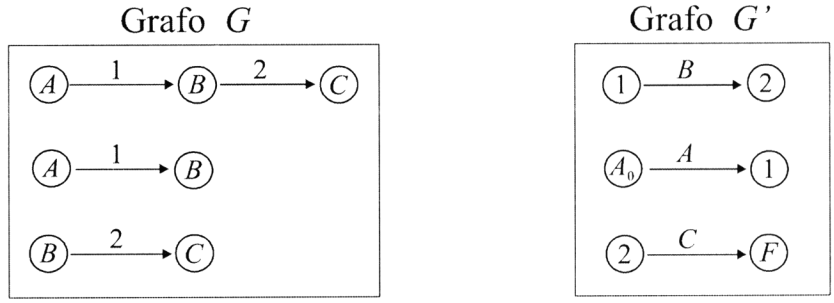


Figura 4.3: Transformación de G a su dual G'

5. Transformación del autómata G' en la gramática regular

Transformamos el autómata G' en una gramática regular de la forma usual. Los no terminales de la gramática los representaremos con la letra A y un subíndice que indica el número que etiqueta el vértice en G'. Por ejemplo A_b o A_f. Además tendremos en cuenta las siguientes consideraciones:

El estado inicial A_0 podrá tener más transiciones que partan desde él como consecuencia de los estados (que no sean estados sin retorno) con arco de entrada u, por donde hay que comenzar en G para tener la consistencia con todos los comienzos de las cadenas de sucesos de M. En este caso añadiremos las reglas $A_0 \to \varepsilon A_u$. Si todos coinciden con los estados sin retorno, desde A_0 sólo tendremos las reglas $A_0 \to e_i A_b$ que obteníamos por la propia construcción de G'.

Todos aquellos estados en G' que tengan un arco de entrada etiquetado con un suceso elemental que sea consistente con los finales de las cadenas de M, serán estados finales para ese arco. El nodo (vértice) F de G' también será estado final. Al pasar a la gramática, para los estados f en G' con

transiciones a F mediante e_p, consideraremos sólo las reglas de la forma $A_f \rightarrow e_p$.

6. Asignación de probabilidades a las reglas

Una vez obtenida la gramática regular, el último paso es la asignación de probabilidades a las reglas para obtener una gramática regular probabilística. Para ello existen dos fases: en la primera se asocian las probabilidades a las reglas usando las dos tablas de probabilidades de transición y en la segunda se relativizan las probabilidades para obtener una gramática regular propia.

A cada regla de la forma $A_b \rightarrow e_j A_f$ que correspondía en G con el camino de e_i a e_j mediante b y de e_j a e_p mediante f, le asociamos como probabilidad, la que le asigna la segunda tabla a la cadena $e_i e_j e_p$.

A las reglas de la forma $A_f \rightarrow e_p$ donde f son los estados en G' con transiciones a F y f era el arco en G de e_j a e_p, se les asigna como probabilidad la que la cadena $e_j e_p$ tenga asignada en la primera tabla. Si f es un estado en G' consistente con todos los finales de las cadenas de M, en la gramática regular existen las reglas $A_b \rightarrow e_j A_f$ y $A_b \rightarrow e_j$. A la regla $A_b \rightarrow e_j$ también se le asigna como probabilidad la que la segunda tabla asigne a la cadena $e_i e_j e_p$.

A las reglas de la forma $A_0 \rightarrow e_i A_b$ donde b era el único arco en G desde e_i a e_j, se le asigna la probabilidad correspondiente a la cadena $e_i e_j$ en la primera tabla.

La probabilidad se distribuye de manera uniforme entre las reglas de la forma $A_0 \rightarrow \varepsilon A_u$. En el caso particular en que sólo exista una regla que tenga como antecedente a A_0 se le asigna probabilidad 1.

Ahora distribuimos las probabilidades para obtener una gramática regular probabilística propia. Para cada A_i, con $i \geq 0$, se toman todas las reglas donde A_i aparezca como antecedente y se calculan los valores relativos de

sus probabilidades para que la suma de las probabilidades de todas las reglas con A_i como antecedente sea 1.

Ejemplo 4.5 Para el grafo G' del ejemplo 4.4 obtenemos la regla $A_1 \rightarrow B\,A_2$ a la que le asignamos la probabilidad que la tabla 2 le asignara a la cadena $A\,B\,C$, la regla $A_0 \rightarrow A\,A_1$ a la que le asignamos la probabilidad que la tabla 1 le asignara a la cadena $A\,B$ y la regla $A_2 \rightarrow C$ a la que le asignamos la probabilidad que la tabla 1 le asignara a la cadena $B\,C$. También tendríamos que asignar las probabilidades de la forma descrita aquellas reglas que surgen para tener la consistencia con los comienzos y finales de cadenas de M y por último relativizar las probabilidades necesarias para que la gramática regular probabilística sea propia. ■

Con este último paso obtenemos una gramática regular probabilística. En la siguiente sección obtenemos la cadena de Markov asociada.

4.2 Cadena de Markov asociada

Bajo determinadas condiciones, dado un sistema y una sucesión de observaciones de la evolución del sistema, se puede considerar una cadena de Markov asociada al sistema. Las cadenas de Markov se aplican aquí al campo del reconocimiento sintáctico de patrones, más concretamente, lo aplicamos a nuestro caso, donde los sistemas son las gramáticas regulares probabilísticas.

Las características relacionadas con el proceso de derivación de una cadena juegan un papel importante en el diseño de una gramática que exprese con suficiente claridad el lenguaje con el que se está trabajando. Por eso, es interesante conocer por ejemplo, cuál es el número medio de reglas que es necesario aplicar para generar una cadena de caracteres o el número medio de caracteres en una cadena, o saber si un carácter aparece con más frecuencia que otro en las cadenas.

Sea $\mathcal{G}_P =< \mathcal{G}, P >$ una gramática regular probabilística propia donde $\mathcal{G} = (V, T, S, r)$. Entonces existe un autómata finito probabilista N que es equivalente a la gramática $\mathcal{G}_P$. El conjunto de estados K de N está formado por el conjunto V de no terminales de $\mathcal{G}_P$ y por el estado F que se crea en N debido a las reglas terminales de la gramática. La llegada al estado F indica que se ha terminado de generar la cadena en la gramática.

En el autómata existe un único estado inicial S que es el estado inicial de la gramática. La función de transición γ del autómata se define a partir de las reglas de derivación de la gramática de la forma habitual. La función de probabilidad π también se define a partir de las probabilidades de las reglas de la gramática. Por ser una gramática regular probabilística propia se cumplen las condiciones con respecto a las funciones de probabilidad: para todo $q \in K - F$,

$$\sum_{\sigma \in \Sigma} \gamma(q, \sigma) = 1 \qquad \text{y} \qquad \sum_{q \in K} \pi(q) = 1.$$

Definición 4.3 *Dado el autómata finito probabilista N que es equivalente a una gramática regular probabilista $\mathcal{G}_P$, se define el proceso estocástico asociado a N como la familia de variables aleatorias $\{X_t : t \in \mathcal{T}\}$ que toman los valores en el conjunto $K = V \cup \{F\}$ siendo $\mathcal{T}$ un espacio paramétrico continuo correspondiente al tiempo.*

Veamos que este proceso estocástico induce una cadena de Markov. El diagrama de transición entre los estados del autómata finito probabilista coincide con el diagrama de transición entre los estados de esta cadena de Markov donde los estados de la cadena van a ser $V \cup \{F\}$.

Proposición 4.1 *Sea N el autómata finito probabilista equivalente a la gramática regular probabilística $\mathcal{G}_P$. Entonces el proceso estocástico $\{X_t : t \in \mathcal{T}\}$ asociado a N es un proceso de Markov.*

Demostración: El proceso estocástico $\{X_t : t \in \mathcal{T}\}$ cumple la propiedad de Markov porque la probabilidad de transición entre estados, es decir, de

aplicar otra regla de la gramática, sólo depende del estado actual, o sea del no terminal que aparece como consecuente en la regla actual, y no de las reglas aplicadas hasta ese momento. □

El proceso de derivación, que se realiza en instantes discretos, es equivalente al paso de un no terminal (antecedente de la regla) a otro no terminal (en el consecuente).

Definición 4.4 *Sea N el autómata finito probabilista equivalente a la gramática regular probabilística $\mathcal{G}_P$ y sea el proceso estocástico $\{X_t : t \in \mathcal{T}\}$ asociado a N. Llamamos* cadena de Markov *asociada a N a la cadena de Markov $\{X_n : n = 0, 1, \dots\}$ inducida del proceso de Marvov $\{X_t : t \in \mathcal{T}\}$ por los instantes $\tau_n \in \mathcal{T}$ de aplicación de una regla en la gramática.*

El espacio de estados de esta cadena de Markov es $K = V \cup \{F\}$, el vector de probabilidades iniciales $\mathbf{P}(0)$ asigna probabilidad 1 al estado inicial S y 0 al resto y la matriz de transición $\mathbf{P} = (p_{I,J})$ se define de la siguiente manera:

$$p_{I,J} = \begin{cases} \displaystyle\sum_{\{p_{i,h}:A_i \to xA_j \ / \ x \in T\}} p_{i,h} & si \ \ I = A_i \in V, J = A_j \in V \\ 1 & si \ \ I = J = F \\ 0 & en \ el \ resto \end{cases}$$

La cadena de Markov obtenida es homogénea porque las probabilidades asociadas a las reglas permanecen constantes en el tiempo y sólo dependen del número de etapas que se saltan y no de la etapa en la que estemos en ese momento. Por lo tanto, existe una cadena de Markov homogénea asociada al proceso de derivación de una gramática regular probabilística.

El algoritmo siguiente permite obtener la cadena de Markov asociada a la gramática regular probabilística y la creación de la matriz de transición de la cadena de Markov a partir de las probabilidades de las reglas de la gramática. Vamos a denotar los estados de la cadena por los subíndices asociados a los no terminales de la gramática y por el estado F.

Algoritmo A.1. (Cadena de Markov asociada)
`Entrada`: Gramática $\mathcal{G}_P$.
`Salida`: Cadena de Markov representada por $\{X_n\}$, $\mathbf{P}$ y $\mathbf{P}(0)$.
`Método`:

1. $\{X_n\} = (V - T) \cup \{F\}$.

2. Inicializar $\mathbf{P}(0)$ con ceros. Asignar distribución de probabilidad a los estados iniciales.

3. Inicializar $\mathbf{P}$ con ceros

4. **for** cada regla de derivación de la gramática **do**
   if $(p_{i,h} : A_i \to a_k A_j)$ **then**
   $P_{i,j} := P_{i,j} + p_{i,h}$
   fi
   if $p_{i,h} : A_i \to a_k$ **then**
   $P_{i,F} := P_{i,F} + p_{i,h}$
   fi
   $P_{F,F} := 1$
   od

Proposición 4.2 *Sea $\mathcal{G}_P =< (V, T, S, r), P >$ una gramática regular probabilística propia y sea $\{X_n\}$ la cadena de Markov asociada a $\mathcal{G}_P$. En el espacio de estados existen clases comunicantes transitorias abiertas formadas por los no terminales V de la gramática y existe una única clase recurrente cerrada formada por el estado absorbente F.*

Demostración: Por el teorema de descomposición de una cadena de Markov finita, se tiene que el espacio de estados se descompone de manera única como unión disjunta de clases comunicantes donde cada clase puede ser de uno de estos dos tipos: recurrente cerrada o transitoria abierta. Y, además, las clases del tipo recurrente tienen que existir necesariamente.

Veamos que en nuestro caso existe al menos una clase transitoria y una única clase recurrente.

El estado F es absorbente y determina una clase recurrente cerrada formada únicamente por ese estado porque si existieran más estados en esa clase, el estado F obligaría a que fueran transitorios y a descomponer la clase en una transitoria y otra recurrente formada por F.

Además esta clase recurrente es única porque al obtener el autómata equivalente a la gramática todas las reglas terminales se han asociado al estado F.

Al ser única la clase recurrente el resto de las clases son transitorias abiertas y sabemos que existe una que corresponde a la clase donde se encuentre el estado inicial S. □

Modelizar nuestra gramática regular probabilística mediante una cadena de Markov es interesante porque podemos estudiar entre otros los siguientes parámetros:

- Ocurrencia media de terminales y no terminales en la generación de una cadena.
- Probabilidad de aparición de terminales y no terminales.
- Probabilidad de terminar la cadena que se está generando.

Ese será ahora nuestro objetivo. Las propiedades relativas a los no terminales (los estados de la cadena de Markov) se pueden obtener de una forma directa. Sin embargo, para poder estudiar las propiedades relativas a los terminales es necesario que las reglas de la gramática cumplan una condición de unicidad. Si no es así, antes de realizar el estudio estadístico tenemos que incluir nuevas reglas en la gramática. Todo este proceso se explica de forma más detallada en la siguiente sección.

4.2.1 Ampliación de la gramática regular probabilística

Para estudiar las propiedades de la gramática desde un punto de vista estadístico no habrá problemas de ambigüedad en el estudio de los parámetros relativos a los terminales de la gramática si se cumple la siguiente condición de unicidad:

> *Cada no terminal en el consecuente está asociado con un único terminal aunque cada terminal puede tener en el consecuente más de un no terminal* asociado.

Veamos un ejemplo que pone de manifiesto por qué es necesario que se cumpla esa condición para hacer un estudio exhaustivo de las propiedades estadísticas de los terminales de la gramática.

Ejemplo 4.6 Supongamos que se tienen las reglas $C \to aS$ y $B \to bS$. La información de la ocurrencia media del estado S de la cadena de Markov no será suficiente para determinar la ocurrencia media de a y de b. ∎

Así, antes de pasar a la cadena de Markov, hay que eliminar esa ambigüedad ampliando la gramática con nuevas reglas y después, convertirla en cadena de Markov. El proceso que luego describimos de forma algorítmica se muestra mediante un ejemplo.

Ejemplo 4.7 En la gramática que tiene las reglas del ejemplo anterior $C \to aS$ y $B \to bS$, se añaden las nuevas reglas $C \to aS_1$ y $B \to bS_2$, eliminando las antiguas. Todas las reglas que tienen a S en el antecedente, por ejemplo $S \to aD$, se duplican de la forma $S_1 \to aD$ y $S_2 \to aD$. Para las reglas de derivación terminales $B \to d$ y $B \to e$ se actúa sustituyéndolas por $B \to dT_1$ y $B \to eT_2$ y $T_k \to \varepsilon$, $\forall\, k$. ∎

Con este ejemplo se pone de manifiesto que si en la ampliación de la gramática surge más de un estado inicial, entonces el vector de probabilidad inicial $\mathbf{P}(0)$ se modifica y se reparte de forma uniforme la probabilidad

entre los estados iniciales. Por eso en el algoritmo A.1. aparece que se distribuye la probabilidad entre los estados iniciales.

Definimos además unos conjuntos para almacenar toda la información sobre la relación entre los no terminales y terminales de gramática original y la ampliada. El conjunto C_{A_j} contendrá a todos los nuevos estados que surgen de A_j y el conjunto C_{a_k} es el conjunto de pares (estados de la gramática ampliada que aparecen en el consecuente junto a a_k, número de veces que aparece a_k junto a esos estados). C_{Term} contiene los estados T_k asociados a las reglas de derivación terminales.

El algoritmo que realiza todo el proceso de la ampliación de la gramática es el siguiente.

Algoritmo A.2. (Ampliación de la gramática)
`Entrada`: Gramática $\mathcal{G}_P$.
`Salida`: Cadena de Markov asociada a la gramática ampliada que cumple la propiedad de unicidad.
`Método`:

1. **for** cada no terminal A_j **do**
   Hallar $DiffA_j = \{a_k \in T / \exists(p_{i,h} : A_i \rightarrow a_k A_j))\ \}$
   if $card(DiffA_j) \geq 1$ **then**
   for cada regla que aparece en la definición de $DiffA_j$ **do**
   Sustituir A_j que está asociado al terminal a_k
   por un nuevo y único no terminal $A_{j,k}$
   $V - T := \{(V - T) - A_j\} \cup \{A_{j,k}\}$
   $C_{A_j} := C_{A_j} \cup \{A_{j,k}\}$
   $C_{a_k} := C_{a_k} \cup \{(A_{j,k}; \sharp a_k)\}$
   od
   fi
   od

2. **for** cada regla r con A_j como antecedente **do**
   for cada regla r' con $A_{j,k}$ en el consecuente **do**

Generar copia de r sustituyendo el antecedente A_j por $A_{j,k}$
od
Eliminar r
od

3. **for** cada regla terminal **do**
   $Diff_{Term} := \{a_k \in T / \exists p_{i,h} : A_i \rightarrow a_k$ en las reglas de la gramática$\}$
   for cada terminal diferente **do**
   Sustituir $p_{i,h} : A_i \rightarrow a_k$ por $p_{i,h} : A_i \rightarrow a_k T_k$
   $V - T := (V - T) \cup \{T_k\}$
   $C_{Term} := C_{Term} \cup \{T_k\}$
   $C_{a_k} := C_{a_k} \cup \{(T_k; \sharp a_k)\}$
   od
   od

4. Aplicar el algoritmo A.1. a la gramática ampliada

5. Añadir $P_{T_k,F} = 1, \forall k$

Las probabilidades que tenían las reglas donde aparecían los no terminales A_j como antecedente las heredan todas las reglas correspondientes donde aparecen los nuevos no terminales $A_{j,k}$ como antecedente.

La matriz de transición **P** de salida se puede dividir por cajas. En la primera caja de la diagonal se encuentra la clase recurrente de F y la segunda caja corresponde con la clase transitoria que se ordena de la siguiente forma. Primero aparece la fila de todos los no terminales correspondientes a C_{A_1}, o sea los $A_{1,k}$, que denotaremos por S_s (ahora todos estos estados son iniciales), después los de C_{A_2} , y así sucesivamente hasta los T_k. Una vez conseguida la matriz **P** se pasa a estudiar las características de la cadena.

4.2.2 Estudio estadístico de la cadena de Markov

Una vez hechas las transformaciones necesarias en la gramática, se estudian las características de la cadena de Markov asociada. Se ha visto que

la cadena tiene una única clase recurrente formada por el estado absorbente F y una o varias clases transitorias formadas por los no terminales.

Un punto importante de estudio es la probabilidad de absorción, es decir, la probabilidad de generar una cadena. No olvidemos que el lenguaje generado por una gramática está formado por cadenas que se generan mediante la derivación de reglas donde aparecen los no terminales. Por ello, se calculará la probabilidad de acabar de generar una cadena si en un instante de la derivación se está en un determinado no terminal (esto nos lo dará la matriz $\mathbf{F}$).

En el lenguaje hay inducida una distribución de probabilidad, por tanto, es interesante calcular también la probabilidad que tiene una cadena en el lenguaje, para ello se calculará la probabilidad de ocurrencia de cada terminal. Además se obtiene la probabilidad que tiene cada no terminal de ser visitado partiendo del estado inicial.

Otras características a estudiar en las cadenas del lenguaje serán la ocurrencia media de cada terminal y la longitud media de las cadenas, que es equivalente al número medio de reglas que se aplican para generar una cadena, (ya que cada regla después de la derivación tiene en el consecuente un sólo terminal). Además se hallará la ocurrencia media de cada no terminal.

También se calculará el número de veces que se visita un no terminal saliendo de cualquier no terminal hasta la absorción y el número de veces que aparece un terminal después de aparecer otro, hasta la absorción.

El algoritmo que se presenta ahora halla todos estos parámetros a partir de la matriz fundamental de una cadena de Markov, que es la matriz que recoge las probabilidades de transición entre estados transitorios de la cadena. Esto es $N = (I - Q)^{-1}$ $(N = (n_{i,j}))$ donde Q es la matriz de las probabilidades de transición entre los estados transitorios. Los elementos de N representan el número medio de ocurrencias de un estado j dado que comenzamos en el estado i. Por lo tanto, en el caso que se estudia interesa

la primera fila de N. Se utilizarán estos elementos de la matriz para calcular la ocurrencia media de terminales (mean-occ(terminales)) y la ocurrencia media de no terminales excepto F (mean-occ(no terminales-$\{F\}$)). La suma de la primera fila de N es el tiempo medio hasta la absorción, dado que la cadena comienza en el estado inicial. En el algoritmo se representa mean-occ(all states).

Algoritmo A.3 (Estudio estadístico)

`Entrada`: Cadena de Markov asociada.

`Salida`: Número medio de visitas, longitud media hasta la absorción, probabilidades de visita, probabilidad de absorción.

`Método`:

1. Aplicar el Algoritmo A.2.
2. Calcular la matriz fundamental $N = (I - Q)^{-1}$ donde Q es la matriz correspondiente a las transiciones entre los estados transitorios.
3. Calcular $\mathbf{F} = NA$ donde A es la matriz de transición desde los estados transitorios al recurrente F.
4. Calcular el número de etapas en las que se visita A_j saliendo de A_p hasta la absorción . Es decir, $\forall j$ y $\forall p$ hallar

$$\sum_{\sharp A_j \times \sharp A_p} n_{A_{p,i},A_{j,k}} \qquad \text{donde} \qquad \begin{cases} A_j = \{A_{j,k},\ \forall k\} \\ A_p = \{A_{p,i},\ \forall k\} \end{cases}$$

5. Calcular el número de veces que aparece a_i después de aparecer a_q hasta la absorción. Es decir, $\forall j$ hallar

$$\sum_{\sharp C_{a_i} \times \sharp C_{a_q}} n_{A_{j,q},A_{j,i}} \times \sharp a_i, \qquad \text{donde} \qquad \begin{cases} C_{a_i} = \{(A_{j,i}, \sharp a_i)\},\ \forall a_i \\ C_{a_q} = \{(A_{j,q}, \sharp a_q)\},\ \forall a_q \end{cases}$$

6. Calcular la ocurrencia media de los terminales. Si v el número de no terminales pertenecientes a la clase C_{A_1} entonces

$$\text{Mean occ}(a_k) = \sum_{s=1}^{v} \left[p(S_s) \left(\sum_{A_{j,k} \in C_{a_k}} n_{s,A_{j,k}} \times \sharp a_k + \sum_{T_k \in C_{a_k}} n_{s,T_k} \times \sharp a_k \right) \right]$$

donde $\sharp$ está asociado al $A_{j,k}$ o T_k del momento.

7. Calcular la ocurrencia media de los no terminales

$$\begin{aligned} \text{Mean occ}(A_j) &= \sum_{s=1}^{v} \left[p(S_s) \left(\sum_{A_{j,k} \in C_{A_j}} n_{s,A_{j,k}} \right) \right] \\ \text{Mean occ}(T_k) &= \sum_{s=1}^{v} \left[p(S_s) \left(\sum_{T_k \in C_{Term}} n_{s,T_k} \right) \right] \\ \text{Mean occ}(S) &= p(S_s) \left(\sum_{s=1}^{v} n_{s,S_s} \right) \end{aligned}$$

donde $S_s \in C_{A_1}$.

8. Calcular la longitud media de la cadena generada

$$\text{Mean String Length} = \sum_{a_k} \text{Mean occ}(a_k)$$

9. Calcular la ocurrencia media de todos los estados

$$\text{Mean occ (all States)} = \sum_{s=1}^{v} \left[p(S_s) \left(\sum_{j,k} n_{s,A_{j,k}} + \sum_{k} n_{s,T_k} \right) \right]$$

10. Calcular la duración media (número medio de etapas) de la cadena desde cualquier estado transitorio E hasta el estado absorbente. Supongamos que $N \in M_{d \times c}$ entonces

$$\text{Mean Time}(E) = \sum_{w=1}^{c} n_{E,w}$$

11. Calcular la probabilidad de absorción

$$P(terminating) = \mathbf{P}(0) \cdot \mathbf{F}$$

12. Calcular la probabilidad de que se visite un estado no terminal dado que se comienza en el estado inicial
for todos los estados $A_{j,k}$ excepto F **do**
Hacer $A_{j,k}$ estado absorbente y calcular la nueva $\mathbf{P}$
Calcular nueva $\mathbf{F}$

$$\mathbf{F} = (f'_{s,A_{j,k}})$$

La probabilidad es

$$p(A_{j,k}) = p(S_s)\left(\sum_{s=1}^{v} f'_{s,A_{j,k}}\right)$$

od

13. Calcular la probabilidad de cada terminal
for todos los terminales **do**

$$p(a_k) = \sum_{A_{j,k} \in C_{a_k}} \frac{\sharp a_k}{\sum\limits_{(A_{j,k}, \sharp a_k)} \sharp a_k} p(A_{j,k})$$

od
donde el $\sharp$ del numerador está asociado al $A_{j,k} \in C_{a_k}$ correspondiente en ese momento en el sumatorio.

Ejemplo 4.8 Consideremos la siguiente gramática probabilística

$$0,4 : S \to a \quad \text{y} \quad 0,6 : S \to a\ S$$

Esta gramática cumple la condición de unicidad por lo tanto podemos considerar directamente su cadena de Markov asociada. Al obtener la cadena de Markov asociada aparecerán los estados F y T_1. El estado T_1 surge

debido a la regla terminal y el estado F para crear el estado absorbente que indica que la gramática ha terminado de generar la cadena.

En este caso, el vector de probabilidades de inicio para los estados F, S y T_1 es $(0, 1, 0)$ (porque S no se ha duplicado), la matriz de transición entre estados (con las filas etiquetadas siguiendo el orden F, S, y T_1) es

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 0.6 & 0.4 \\ 1.0 & 0.0 & 0.0 \end{pmatrix}$$

y el estudio estadístico de la cadena determina los resultados que aparecen en la siguiente tabla:

Parámetro	Valor
Ocurr. Media (a)	3.5
Ocurr. Media (S)	2.5
Ocurr. Media (T_1)	1.0
Ocurr. Media de todos los estados	1.0
Duración media, desde (S) hasta el estado absorbente F	3.5
Duración media, desde (T_1) hasta el estado absorbente F	1.0
Nº de veces que aparece (a) después de (a)	4.5
Nº etapas que se visita (S) saliendo de (S)	2.5
Nº etapas que se visita (T_1) saliendo de (S)	1.0
Long. Media Cadenas	3.5
Prob. de Terminación	1.0
Probabilidad (S)	1.0
Probabilidad (T_1)	1.0
Probabilidad (a)	1.0

■

Se puede recuperar la gramática regular probabilística a partir del diagrama de transición asociado a la cadena. El diagrama de transición puede

verse como un autómata etiquetando los arcos de transición con el único terminal que aparece en el consecuente de la regla.

4.3 Inferencia de gramáticas regulares difusas

Ahora presentamos el estudio desde el punto de vista difuso para inferir una gramática regular difusa. Este es el algoritmo de inferencia de este caso:

1. Definición de dos relaciones binarias difusas

1. Sea E el conjunto de todos los elementos del alfabeto e_i que aparecen en $M \subset S_{\mathcal{L}}^{+}$. Se define la relación binaria difusa $P(E,E)$ como "-cercano a-" y con función de pertenencia $\mu_P(e_i, e_j)$. Esta función nos indica el grado de creencia que tenemos de que $e_i \in \Sigma$ con $i = 1, \ldots, n$ venga seguido en la cadena de $e_j \in \Sigma$ con $j = 1, \ldots, n$.

2. Sea D el conjunto de todas las cadenas de sucesos de longitud dos que tienen asociado grado distinto de cero en R. Se define la relación difusa $Q(D,E)$ como "-cercano a-" con función de pertenencia μ_Q que indicará el grado de creencia de que esas cadenas de longitud 2 vengan seguidas del elemento e_i $i = 1, \ldots, n$. Para calcular el grado de cada cadena de longitud tres se usa una t-norma y para reflejar el número de existencias de esa cadena se utiliza la co-norma correspondiente para acumular los grados.

2. Construcción del grafo de sucesos G

Creamos el grafo G a partir de la matriz asociada a P de la siguiente forma:

- Creamos un vértice por cada símbolo del alfabeto y lo etiquetamos con ese símbolo.

- Si la probabilidad para la cadena $e_i e_j$ es mayor que el nivel de confianza fijado creamos un arco que va desde el símbolo e_i al e_j. Los arcos se etiquetan con números naturales consecutivos.

Al construir el grafo G, los circuitos que aparecen en las cadenas de la muestra deben mantenerse, permaneciendo los arcos de transición relativos a ese circuito en la misma cara del grafo.

3. Destrucción de la sobreconectividad

El paso anterior puede dar lugar a vértices en el grafo G sobreconectados produciendo así secuencias ilegales (caminos $e_i e_j e_p$ en G con grado igual a cero en la matriz asociada a Q). Entonces destruimos la sobreconectividad de forma análoga al paso 3 del algoritmo de inferencia de gramática regular probabilística.

4. Construcción del dual G'

Análogo al paso 4 del algoritmo de inferencia de gramáticas regulares probabilísticas.

5. Transformación del autómata G' en la gramática regular

Análogo al paso 5 del algoritmo de inferencia de gramáticas regulares probabilísticas.

6. Gramática regular difusa: asignación de grados a las reglas

A cada regla de la gramática regular obtenida le asignamos un grado de la siguiente forma:

A cada regla de la forma $A_b \rightarrow e_j A_f$ que correspondía en G con el camino de e_i a e_j mediante el arco b y de e_j a e_p mediante el arco f, le asignamos el grado asociado a la cadena $e_i e_j e_p$ mediante Q.

A las reglas $A_f \rightarrow e_p$ donde f son los estados que en G' tienen transiciones a F y además f era el arco en G que iba desde e_j a e_p, le asignamos como grado el que P le asignaba a la cadena $e_j e_p$. Si f es un estado en G' consistente con todos los finales de las cadenas de sucesos de S, en la gramática regular existen las reglas $A_b \rightarrow e_j A_f$ y $A_b \rightarrow e_j$. A la regla $A_b \rightarrow e_j$ se le asigna como grado el que asigne la relación Q a la cadena $e_i e_j e_p$.

A las reglas de la forma $A_0 \rightarrow e_i A_b$, donde A_b era un arco en G que iba desde e_i a otro suceso elemental e_j y además era el único arco con esa propiedad, le asignaremos como grado el que tenga asociado la cadena $e_i e_j$ mediante P. A las reglas de la forma $A_0 \rightarrow \varepsilon A_u$ le asignamos grado 1. En el caso particular en el que A_0 sólo tenga un arco de salida le asignaremos grado 1 a esa regla.

4.4 Complejidad

Se realiza ahora un estudio para mostrar que el algoritmos de inferencia de gramáticas regulares probabilísticas y el algoritmo de inferencia de gramáticas regulares difusas tienen complejidad polinómica.

Sea $|\Sigma| = n$ el número de símbolos del alfabeto. Sea $|M| = m$ el número de ejemplos en la muestra y sea $t + 1$ la longitud del ejemplo más largo en la muestra. En estas condiciones sobre el total de ejemplos hay a lo sumo $t \times m$ subcadenas de longitud 2 y $(t - 1) \times m$ subcadenas de longitud 3

(distintas o no).

La primera tabla en los dos casos (probabilístico y difuso) tiene n^2 elementos cuyo cálculo supone $t \times m$ cómputos. Si aparece la subcadena $e_i e_j$ en el caso probabilístico incrementamos en 1 el elemento (i, j) de la tabla; posteriormente todos esos números se dividen por el total de subcadenas consideradas por cada fila y en el caso difuso si aparece la subcadena $e_i e_j$ entonces calculamos el grado del elemento (i, j). La segunda tabla tiene n^3 elementos cuyo cálculo requiere $(t-1) \times m$ cómputos (en los dos casos se hace de manera análoga a lo anterior).

Considerando el proceso en forma directa: Para la construcción del grafo G, la primera tabla de cada caso se convierte en una matriz N que representa al grafo G con n vértices. Para destruir la sobreconectividad, pasamos a lo sumo a $n+n$ vértices, lo que supone a lo sumo $(2 \cdot n)^2$ arcos (y una nueva matriz N'). La construcción del dual es un proceso cuadrático en el número de vértices de G.

Por último la transformación en una gramática y la asignación de probabilidades en la versión probabilística y la asignación de grados en el caso difuso, son ambos procesos de orden lineal con respecto al tamaño del grafo. En resumen, los algoritmos son polinómicos de grado 4 en el número de símbolos del alfabeto.

Como dijimos, los algoritmos controlan la sobreconectividad de longitud 3. Este proceso se podría continuar para longitudes 4, 5, etc, con un incremento multiplicativo en el tiempo consumido, en un factor igual al tamaño del alfabeto. El nivel de robustez del modelo probabilístico es controlado por el nivel de confianza que fijemos para las probabilidades. Este parámetro puede usarse también para controlar la complejidad en el modelo que se infiere a partir de grandes muestras de datos. Si conocemos que los datos son correctos, es decir no contienen ruido, no se ignorará ninguna cadena de sucesos con probabilidad distinta de cero. En el caso difuso el nivel de confianza lo controla la función de asignación de grados.

4.5 Aplicación de los algoritmos de inferencia

En esta sección aplicamos los algoritmos de inferencia descritos a un caso práctico. El problema que se presenta intenta capturar la estructura intrínseca de las pinturas de Piet Mondrian (1872-1944), para poder generar automáticamente, otras pinturas consideradas dentro del estilo de Mondrian. Consideramos las pinturas como una muestra de palabras generadas por unas reglas que pertenecen a una gramática regular probabilística o difusa. La aplicación consecutiva de las reglas de la gramática obtiene una nueva pintura donde la confianza de que la pintura sea del estilo del pintor, la da la probabilidad o el grado que tenga asociada esa palabra (pintura) en el lenguaje. Un estudio más profundo de las actuaciones de la gramática, lleva a considerar la cadena de Markov asociada y a estudiar las propiedades relativas a esta cadena de Markov.

Los pasos a seguir para modelar este problema mediante la inferencia de gramáticas regulares probabilísticas o difusas son:

1. Algoritmo de conversión de cuadros a cadenas.
2. Ejecución del algoritmo de inferencia de gramáticas regulares probabilísticas o difusas.
3. Algoritmo de conversión de cadenas a cuadros.

4.5.1 Algoritmo de conversión de cuadros a cadenas

Describimos cómo convertir los cuadros de Mondrian en cadenas unidimensionales. Los cuadros escogidos corresponden a la etapa abstracta del pintor. Los cuadros están formados por rectángulos blancos, azules, rojos y amarilos. Se ha considerado un sistema de referencia basado en los puntos cardinales. Los puntos cardinales usados han sido estos: $N, NE, NW, N \wedge NE, N \wedge NW, S, SE, SW, S \wedge SE, S \wedge SW, E, E \wedge NE, E \wedge SE, W, W \wedge NW, W \wedge SW$. Cada punto cardinal puede venir acompañado del color R

(red), B (blue) o Y (yellow) que tenga asignado el rectángulo en ese punto cardinal. El alfabeto terminal tiene 39 símbolos porque no todos los puntos cardinales tienen asociado un color en las pinturas de la muestra.

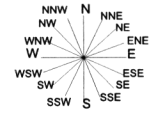


Figura 4.4: Puntos cardinales

Para transformar los cuadros a cadenas se considera en cada rectángulo del cuadro su esquina superior izquierda para fijar su orientación y a cada rectángulo le corresponde una estructura de la forma: (coordenada x, coordenada y, anchura, altura, color). Una vez el cuadro en dicho formato, se procede a pasar a cadenas. Para ello, de forma recursiva, a cada coordenada se le hace corresponder cada una de las posibles cadenas, y se mira si se cumplen los requisitos para que dicho rectángulo del cuadro pueda ser sustituido por la cadena en curso.

Ejemplo 4.9 Consideremos el siguiente cuadro (Composition in Red, Blue and Yellow (1928), 122×79cm, La Haya Gemeentemuseum), ver figura 4.5. Una posible cadena de generación podría ser

$$(EB)(N \wedge NE)(E \wedge SE)(NR)(SW)(S \wedge SEY)(rec)$$

El proceso que se describe ahora se puede ver en la figura 4.6. El recuadro azul está al este del cuadro, luego se escribirá EB (B=Blue). Para codificar el recuadro blanco que existe por encima del azul se le asigna $(N \wedge NE)$ que es el punto cardinal correspondiente. Al recuadro que queda por debajo del azul se le asigna el punto cardinal $(E \wedge SE)$. Ahora el recuadro rojo se encuentra al norte y este punto cardinal todavía no se ha utilizado así que se le asocia NR. El recuadro blanco de la esquina inferior izquierda corresponde con el punto cardinal SW. El recuadro que tiene al

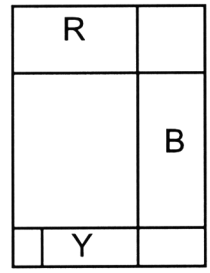


Figura 4.5: Composition in Red, Blue and Yellow (1928)

lado amarillo con $(S \wedge SEY)$. Y por último el recuadro blanco que queda se le asigna *rec*. ■

4.5.2 Algoritmos de inferencia

El algoritmo de inferencia de gramáticas regulares probabilísticas se puede aplicar directamente porque tiene todos sus parámetros definidos. Cuando se obtenga una gramática regular probabilística se puede obtener la cadena de Markov asociada y realizar el estudio estadístico de la gramática.

Nos centramos ahora en explicar cómo se han definido las relaciones difusas P y Q necesarias para la ejecución del algoritmo. Consideramos el conjunto E como el de todos los terminales existentes en las cadenas de la muestra M. Más concretamente los elementos de E serán los puntos cardinales con sus respectivos colores asociados que aparecen en M.

La función de pertenencia μ_P se ha definido como sigue: a las cadenas de longitud dos que no aparecen en ninguna cadena de M se le asocia grado cero; a las cadenas de sucesos de longitud dos que sí aparecen se le asocia un grado de creencia teniendo en cuenta la distancia en grados que separa un punto cardinal del otro. En todos los casos que se han estudiado los ángulos se reducen al primer cuadrante. A estos ángulos se les halla el seno y mediante un proceso de interpolación y de redondeo tomamos los valores:

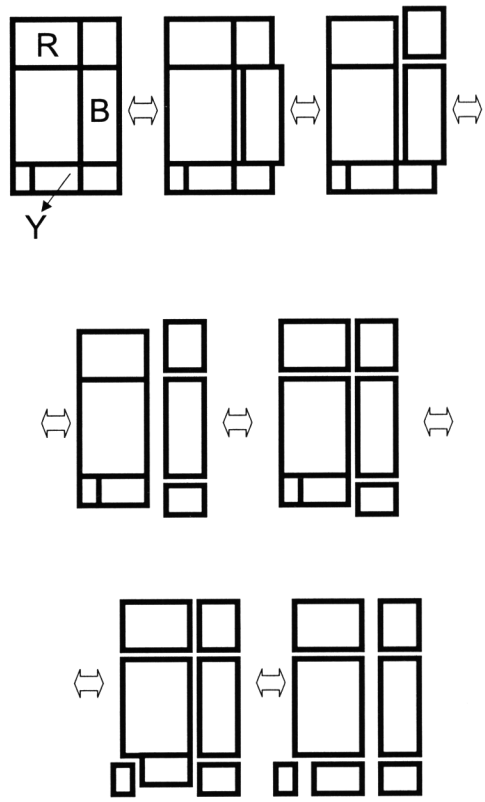


Figura 4.6: $(EB)(N \wedge NE)(E \wedge SE)(NR)(SW)(S \wedge SEY)(rec)$

1; 0,97; 0,95; 0,92; 0,70. Estos son los grados que asigna μ_P.

Definimos la función de pertenencia μ_Q teniendo en cuenta la existencia de las cadenas de longitud tres en M. Si no existen, el grado es cero y si existen se le asocia como grado la intersección de los grados mediante la medida de la intersección de Dombi [Klir y Folger, 1988] considerando así una composición de una relación binaria en sí misma. El número de existencias en la cadena de longitud tres se tiene en cuenta haciendo una acumulación de grados mediante la unión difusa de Dombi.

$$\frac{1}{1+\left[\left(\frac{1}{a}-1\right)^{-\lambda}+\left(\frac{1}{b}-1\right)^{-\lambda}\right]^{-\frac{1}{\lambda}}} \quad \lambda \in (0,\infty)$$

Unión difusa de Dombi (1982)

$$\frac{1}{1+\left[\left(\frac{1}{a}-1\right)^{\lambda}+\left(\frac{1}{b}-1\right)^{\lambda}\right]^{\frac{1}{\lambda}}} \quad \lambda \in (0,\infty)$$

Intersección difusa de Dombi (1982)

A partir de aquí se ejecuta el algoritmo de inferencia de gramáticas regularas difusas descrito anteriormente.

4.5.3 Algoritmo de conversión de cadenas a cuadros

Para convertir las cadenas en cuadros, se comienza con un rectángulo blanco donde están disponibles todos los puntos cardinales. Al recorrer la cadena, se van insertando los rectángulos en el punto cardinal que indique la cadena con sus respectivos colores. Al insertar estos rectángulos vamos teniendo disponibles menos puntos cardinales, que no podremos volver a utilizar hasta que se produzca otro rectángulo blanco dentro del cuadro que vamos creando. En ese momento, trasladamos el sistema de referencia a

ese rectángulo blanco y volvemos a tener disponibles todos los puntos cardinales, (en el ejemplo de antes, lo que se haría sería el proceso inverso y se irían colocando los rectángulos en el punto cardinal que indique la cadena). Nótese que existen varias cadenas que representan la misma pintura.

Una vez acabado este proceso, se trabaja en la asignación de las medidas exteriores del cuadro y en la posición de líneas horizontales y verticales dentro de él.

Veamos primero cómo asignar estas medidas a las pinturas generadas, Las pinturas de Mondrian que se han estudiado se pueden dividir en dos etapas, en la primera, las pinturas tienen pocos rectángulos mientras que en la segunda etapa tienen un número mayor de rectángulos. Por lo tanto se ha hecho una clasificación en el estudio dependiendo del número de rectángulos de la pintura generada.

Clasificada ya la pintura generada en función del número de sus rectángulos, se escogen sus medidas exteriores de una tabla de probabilidades que representan la frecuencia con que el pintor usó esas dimensiones.

Teniendo las medidas exteriores se estudia la posición de las líneas horizontales y verticales de la pintura. Si la pintura generada tiene menos de 10 rectángulos se descompensan las proporciones en la misma medida que lo estén las de la pintura original que le corresponda. Esta correspondencia se hace buscando la diferencia mínima (entre rectángulos y orientación de los rectángulos) de las pinturas originales y la generada.

Si la pintura generada tiene 10 o más de 10 rectángulos, se trazan las líneas horizontales y verticales de forma similar a la pintura original que le corresponda. Esta correspondencia está basada en el criterio descrito previamente.

Asignar las medidas exteriores y el trazado de líneas horizontales y verticales a las pinturas originales es fácil, pues están todas almacenadas y sólo hay que reproducirlas.

4.5.4 Resultados obtenidos

Partiendo de la muestra de todas las cadenas que generan los 29 cuadros de Mondrian se ha obtenido mediante la aplicación de los algoritmos de inferencia una gramática regular que consta de 129 reglas y 93 no terminales.

La obtención de nuevos cuadros mediante la gramática regular probabilística y mediante la gramática regular difusa se hace del siguiente modo. Para generar un nuevo cuadro en el caso probabilístico se aplican consecutivamente las reglas y se obtiene una cadena con su respectiva probabilidad (el producto de las probabilidades de las reglas que se han aplicado) que se transformará en el cuadro mediante el algoritmo de conversión. Si una misma pintura es obtenida a través de varias cadenas, se toma como mejor cadena para generar esa pintura, la que tenga mayor probabilidad asociada.

La probabilidad asociada a cada cuadro es el nivel de credibilidad de que la pintura generada por la gramática se pueda considerar dentro del estilo de Mondrian. Para generar un nuevo cuadro en el caso difuso se aplican consecutivamente las reglas y se obtiene una cadena con su respectivo grado que se calcula de la forma usual en una gramática regular difusa, usando la t-norma de Dombi nuevamente. Esta cadena se transforma en el cuadro mediante el algoritmo de conversión.

Si una misma pintura es obtenida a través de varias cadenas se toma como mejor cadena para generar esa pintura la que tenga mayor grado asociado.

El grado η, asociado a cada cuadro es el nivel de credibilidad de que la pintura generada por la gramática se pueda considerar dentro del estilo de Mondrian.

En las figuras 4.7, 4.8, 4.9, 4.10, se muestran ejemplos de nuevos cuadros generados usando la gramática regular difusa. Predomina la verticalidad frente al aspecto horizontal como se presenta en los cuadros de Mondrian. Además las líneas verticales son cortadas sólo por algunas líneas horizon-

tales. Como en los cuadros de Mondrian los colores se sitúan en la frontera de los cuadros y aislados unos de otros.

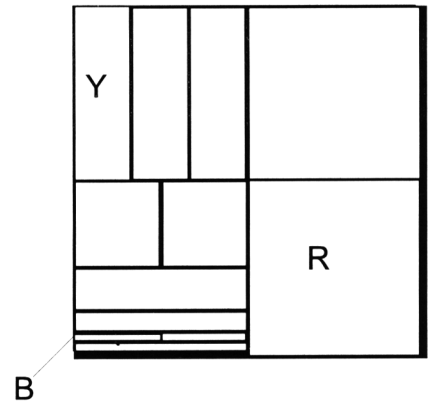


Figura 4.7: $\eta = 0.3090$

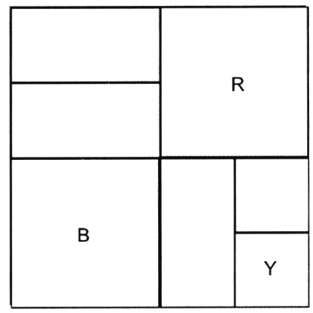


Figura 4.8: $\eta = 0.4535$

Capítulo 5

Conclusiones y trabajo futuro

En este trabajo de investigación se han estudiado tres problemas que se enmarcan dentro de las tareas de asociación, clasificación e inferencia de gramáticas. Se han usado técnicas de prospección de datos y de aprendizaje computacional.

Los objetivos marcados eran el estudio de la búsqueda de conjuntos frecuentes con información negativa, la adaptación de los algoritmos de aprendizaje inductivo de árboles de decisión para incluir los valores desconocidos y la inferencia de gramáticas regulares probabilísticas y gramáticas regulares difusas. Para cada uno de estos puntos se han desarrollado algoritmos que obtienen esos resultados.

Para la búsqueda de información negativa se ha desarrollado un algoritmo polinómico en el número de pasadas por la base de datos y se han presentado una serie de resultados para evitar la evaluación explícita de determinadas frecuencias en la base de datos reduciendo así el coste computacional. Además los algoritmos organizan la búsqueda añadiendo atributos negativos secuencialmente. De esa forma se puede controlar la

aparición de información negativa irrelevante.

Nuestro método se puede combinar con otras ideas que se han usado junto con Apriori como puede ser muestreo para evaluar las frecuencias [Domingo et al., 1999], o también se pueden usar junto con nuestra estrategia algoritmos alternativos como DIC [Brin et al., 1997] o Ready&Go [Baixeries et al., 2000] en la búsqueda de la información frecuente. Nuestra actual línea de trabajo consiste en implementar nuestros algoritmos para su evaluación en distintas bases de datos. Además trabajaremos en la búsqueda de reglas muy fuertes introducidas en [Berzal et al. 2001].

Otra línea de trabajo a corto plazo es la de extender nuestro estudio de búsqueda de información negativa frecuente a la búsqueda de episodios frecuentes con información negativa en secuencias.

A medio plazo, usaremos nuestros algoritmos de inferencia (en las dos versiones probabilística y difusa) para extraer patrones de conducta en los ficheros logs. Creemos que debido al carácter generador de la gramática, se pueden predecir patrones con cierto nivel de confianza. Utilizaremos otro de los resultados desarrollados: la obtención de la cadena de Markov asociada a la gramática regular probabilística para hacer un estudio estadístico del lenguaje y por tanto de los patrones de comportamiento del usuario.

Por último, se ha presentando un nuevo enfoque para trabajar con valores desconocidos en los atributos que integra conceptos de probabilidad y algoritmos de aprendizaje inductivos. Se han adaptado todas las etapas en los algoritmos de aprendizaje inductivo de árboles de decisión donde el valor del atributo es necesario.

Se ha definido formalmente el concepto de criterio de división y se ha adaptado para trabajar con valores desconocidos en los atributos. En nuestro método de asignación de valores se tiene en cuenta la información que dan los valores del atributo y los valores de la clase. También se ha estudiado cómo hacer la predicción de observaciones con valores desconocidos en todas las variantes de los árboles de decisión. Se ha desarrollado un

nuevo enfoque para asignar valores usando la teoría de la decisión. Cada experiencia con un valor desconocido en un atributo tiene dos parámetros asociados: confianza y error.

Como trabajo futuro tendremos en cuenta al menos tres aspectos: riesgo, poda y boosting. Con el riesgo intentaremos mejorar la predicción (como un parámetro independiente o como función de otros parámetros). Proponemos la poda para simplificar los árboles. En particular usaremos el método CIDIM [Ramos y Morales, 1999], [Ramos et al., 2000]. Finalmente, proponemos adaptar otros métodos para obtener árboles de decisión (mediante votación [Ramos, 2001] y muestreo [Ramos y Morales, 2000]) para nuestro caso: valores desconocidos en los atributos. Otra línea de trabajo consistirá en generalizar nuestro enfoque para asignar valores teniendo en cuenta dos o más atributos que tengan una correlación alta.

Apéndice A

Algoritmos de búsqueda acotada en los negativos

Los siguientes algoritmos calculan todos los itemsets frecuentes con un número acotado de atributos negativos. Hemos incluido una condición en los algoritmos, para que no generen candidatos que tengan más atributos negativos que los permitidos. Denotamos por MaxNeg el número máximo de atributos negativos permitidos en los itemsets. Suponemos la condición MaxNeg ≥ 1 para que el estudio de los atributos negativos tenga sentido.

Para que los algoritmos no consideren las plantas superiores a la cota MaxNeg impuesta, hacemos una modificación de los valores que puede tomar la variable correspondiente a la planta, en cada bucle donde intervenga la variable correspondiente a la planta. En los algoritmos desarrollados en el capítulo de información negativa frecuente, los valores de la variable ligada a la planta que aparecía en los bucles siempre estaban ligados a los valores del nivel ℓ.

Entonces, si MaxNeg es mayor que ℓ, la planta como máximo puede tomar el valor ℓ ya que ℓ es el cardinal del itemset y no puede haber más atributos negativos que atributos en el itemset. Si MaxNeg es menor que

ℓ entonces la planta sólo puede llegar a tomar el valor MaxNeg porque si dejamos que planta sea mayor que MaxNeg podríamos obtener itemsets con más atributos negativos que MaxNeg. Resumiendo, la condición consiste en que la variable de la planta siempre tiene que ser menor o igual que $\min(\ell, \text{MaxNeg})$

Algoritmo Acotado-Neg-Apriori

1. $L_{0,1} := \{A^{\{i\},\emptyset}, \forall i \in I / \, fr(A^{\{i\},\emptyset}) > \sigma\}$;
   $L_{1,1} := \{A^{\emptyset,\{i\}}, \forall i \in I / \, fr(A^{\emptyset,\{i\}}) = 1 - fr(A^{\{i\},\emptyset}) > \sigma\}$;
   $\overline{I} := \{i \in I / A^{\emptyset,\{i\}} \in L_{1,1}\}$;
   if $\overline{I} = \emptyset$ **then** Apriori **else**

2. $C_{0,2} := \{A^{p,\emptyset} \in I_{0,2} / \, \forall i \in p, \, A^{p-\{i\},\emptyset} \in L_{0,1}\}$;
   $L_{0,2} := \{A^{p,\emptyset} \in C_{0,2} / \, fr(A^{p,\emptyset}) > \sigma\}$;

3. $f := 0$; $\ell := 2$; $J := \emptyset$;

4. **while** $f \leq \min(\ell, \text{MaxNeg})$ and $\ell \leq n$ **do**
   while $L_{f,\ell} \neq \emptyset$ and $\ell \leq n$ **do**
   $k := f + 1$;
   $L_{\ell,\ell-1} := \emptyset$;
   Acot-Cand-Freq
   if $\ell = n$ **then** output;
   $\ell := \ell + 1$;
   $J := J \cup \{k + 1 / \, k \in J, k < n\}$;
   if $f = 0$ **then**
   $C_{0,\ell} := \{A^{p,\emptyset} \in I_{0,\ell} / \, \forall i \in p, \, A^{p-\{i\},\emptyset} \in L_{0,\ell-1}\}$;
   $L_{0,\ell} := \{A^{p,\emptyset} \in C_{0,\ell} / \, fr(A^{p,\emptyset}) > \sigma\}$
   else
   $L_{f,\ell} := \emptyset$
   fi
   od (while ℓ)
   if $f = \min(\ell, \text{MaxNeg})$ **then** output;

$f := f + 1$;
$C_{f,\ell} := \{A^{p,s} \in I_{f,\ell} / \forall i \in p, A^{p-\{i\},s} \in L_{f,\ell-1},$
$\forall j \in s,\ A^{p,s-\{j\}} \in L_{f-1,\ell-1}\}$;
$L_{f,\ell} := \{A^{p,s} \in C_{f,\ell} / \exists j \in s, fr(A^{p,s-\{j\}}) > \sigma + fr(A^{p\cup\{j\},s-\{j\}})\}$;
$J := J \cup \{f-1\}$
od (while f)

5. output $\bigcup_{k\leq\ell\leq n} L_{k,\ell}$

fi

Acot-Cand-Freq
while $k \leq$min(ℓ,MaxNeg) **do**
if $k \notin J$ **then**
$L_{k,\ell} := \emptyset$;
if $\sigma \leq 0,5$ **then**
$C_{k,\ell} := \{A^{p,s} \in I_{k,\ell} / \forall i \in p, A^{p-\{i\},s} \in L_{k,\ell-1},$
$\forall j \in s, A^{p,s-\{j\}} \in L_{k-1,\ell-1}\}$
else
$C_{k,\ell} := \{A^{p,s} \in I_{k,\ell} / \forall i \in p, A^{p-\{i\},s} \in L_{k,\ell-1},$
$\forall j \in s, A^{p,s-\{j\}} \in L_{k-1,\ell-1},$
$\forall j \in s, fr(A^{p\cup\{j\},s-\{j\}}) < \sigma\}$
fi
$L_{k,\ell} := \{A^{p,s} \in C_{k,\ell} / \exists j \in s,\ fr(A^{p,s-\{j\}}) > \sigma + fr(A^{p\cup\{j\},s-\{j\}})\}$;
if $L_{k,\ell} = \emptyset$ **then** $J := J \cup \{k\}$ **fi**
fi
$k := k + 1$;
od (while k)

Algoritmo Acotado-Neg-Apriori-1 ($\sigma \leq 0,5$)

1. $L_{0,1} := \{A^{\{i\},\emptyset},\ \forall i \in I/\ fr(A^{\{i\},\emptyset}) > \sigma\}$;
   $L_{1,1} := \{A^{\emptyset,\{i\}},\ \forall i \in I/\ fr(A^{\emptyset,\{i\}}) = 1 - fr(A^{\{i\},\emptyset}) > \sigma\}$;
   $\overline{I} := \{i \in I\ / A^{\emptyset,\{i\}} \in L_{1,1}\}$;
   if $\overline{I} = \emptyset$ **then** Apriori **else**

2. $f := 0$; $\ell := 1$; $J := \emptyset$;

3. **while** $f \leq$min(ℓ,MaxNeg) and $\ell < n$ **do**
   while $L_{f,\ell} \neq \emptyset$ and $\ell < n$ **do**
   $\ell := \ell + 1$;
   $L_{-1,\ell-1} := \emptyset$; $L_{\ell,\ell-1} := \emptyset$;
   if $f \notin J$ **then**
   $C_{f,\ell} := \{A^{p,s} \in I_{f,\ell}/\ \forall i \in p, A^{p-\{i\},s} \in L_{f,\ell-1},$
   $\forall j \in s, A^{p,s-\{j\}} \in L_{f-1,\ell-1}\}$
   else
   $L_{f,\ell} := \emptyset$
   fi
   Acot-Cand-Freq-level
   $J := J \cup \{k+1/\ k \in J, k < n\}$;
   od (while ℓ)
   while $L_{f,\ell} = \emptyset$ and $f \leq$min(ℓ,MaxNeg) and $\ell < n$ **do**
   $J := J \cup \{f\}$;
   $f := f + 1$
   od(while f)
   od(while f and ℓ)

4. output $\bigcup_{k \leq \ell \leq n} L_{k,\ell}$
   fi

Acot-Cand-Freq-level

$k := f + 1$;
$L_{\ell,\ell-1} := \emptyset$;
while $k \leq$min(ℓ,MaxNeg) **do**
if $k \notin J$ **then**
$C_{k,\ell} := \{A^{p,s} \in I_{k,\ell} / \forall i \in p, A^{p-\{i\},s} \in L_{k,\ell-1},$
$\forall j \in s, A^{p,s-\{j\}} \in L_{k-1,\ell-1}\}$;
$\tilde{C}_{k,\ell} := \{A^{p,s} \in C_{k,\ell} / \forall j \in s, A^{p\cup\{j\},s-\{j\}} \notin C_{k-1,\ell}\}$
fi
$k := k + 1$
od (while k)
$\tilde{C}_\ell := C_{f,\ell} \cup \left(\bigcup_{k:=f+1,k\notin J}^{\min(\ell,\text{MaxNeg})} \tilde{C}_{k,\ell} \right)$;
$\tilde{L}_\ell := \{A^{p,s} \in \tilde{C}_\ell \,/\, fr(A^{p,s}) > \sigma\}$;
$L_{f,\ell} := \tilde{L}_\ell \cap C_{f,\ell}$;
$k := f + 1$;
while $k \leq$min(ℓ,MaxNeg) **do**
if $k \notin J$ **then**
$L_{k,\ell} := \{A^{p,s} \in C_{k,\ell} - \tilde{C}_{k,\ell} / \exists j \in s,\ fr(A^{p,s-\{j\}}) > \sigma + fr(A^{p\cup\{j\},s-\{j\}})\}$;
$L_{k,\ell} := L_{k,\ell} \cup (\tilde{L}_\ell \cap \tilde{C}_{k,\ell})$;
if $L_{k,\ell} = \emptyset$ **then** $J := J \cup \{k\}$ **fi**
else $L_{k,\ell} := \emptyset$
fi
$k := k + 1$
od(while k)

Algoritmo Acotado-Neg-Apriori-2 ($\sigma > 0,5$)

1. $L_{0,1} := \{A^{\{i\},\emptyset},\ \forall i \in I / fr(A^{\{i\},\emptyset}) > \sigma\}$;
   $L_{1,1} := \{A^{\emptyset,\{i\}},\ \forall i \in I / fr(A^{\emptyset,\{i\}}) = 1 - fr(A^{\{i\},\emptyset}) > \sigma\}$;
   $\overline{I} := \{i \in I / A^{\emptyset,\{i\}} \in L_{1,1}\}$;
   if $\overline{I} = \emptyset$ **then** Apriori **else**

2. $C_{0,2} := \{A^{p,\emptyset} \in I_{0,2} / \forall i \in p,\ A^{p-\{i\},\emptyset} \in L_{0,1}\}$;
   $L_{0,2} := \{A^{p,\emptyset} \in C_{0,2} / fr(A^{p,\emptyset}) > \sigma\}$;

3. $f := 0$; $\ell := 2$; $J := \emptyset$;

4. **while** $L_{0,\ell} \neq \emptyset$ and $\ell < n$ **do**
   Acot-Cand-level
   $C_{0,\ell+1} := \{A^{p,\emptyset} \in I_{0,\ell+1} / \forall i \in p,\ A^{p-\{i\},\emptyset} \in L_{0,\ell}\}$;
   $\tilde{C}_\ell := C_{0,\ell+1} \cup \left(\bigcup_{k:=1,k\notin J}^{\min(\ell,\text{MaxNeg})} \tilde{C}_{k,\ell} \right)$;
   $\tilde{L}_\ell := \{A^{p,s} \in \tilde{C}_\ell / \ fr(A^{p,s}) > \sigma\}$;
   $L_{0,\ell+1} := \tilde{L}_\ell \cap C_{0,\ell+1}$;
   Acot-Freq-level
   $J := J \cup \{k+1 / k \in J, k < n\}$;
   $\ell := \ell + 1$
   od (while ℓ)

5. **if** $L_{0,\ell} \neq \emptyset$ and $\ell = n$ **then** output;
   Acot-Cand-level
   $\tilde{C}_\ell := \bigcup_{k:=1,k\notin J}^{\min(\ell,\text{MaxNeg})} \tilde{C}_{k,\ell}$;
   $\tilde{L}_\ell := \{A^{p,s} \in \tilde{C}_\ell / \ fr(A^{p,s}) > \sigma\}$;
   Acot-Freq-level
   if $\ell = n$ then output;
   while $L_{f,\ell} = \emptyset$ and $f \leq \min(\ell,\text{MaxNeg})$ **do**
   $J := J \cup \{f\}$;
   $f := f + 1$
   od(while f)

6. **Punto 3 del algoritmo de** $\sigma \leq 0,5$

7. output $\bigcup_{k \leq \ell \leq n} L_{k,\ell}$

fi

Acot-Cand-level

$k := 1$; $L_{\ell,\ell-1} := \emptyset$; $\tilde{C}_{0,\ell} = \emptyset$;

while $k \leq$min(ℓ,MaxNeg) **do**

if $k \notin J$ **then**

$C_{k,\ell} := \{A^{p,s} \in I_{k,\ell} \,/\, \forall i \in p, A^{p-\{i\},s} \in L_{k,\ell-1},$
$\forall j \in s, A^{p,s-\{j\}} \in L_{k-1,\ell-1},$
$\exists j \in s, A^{p\cup\{j\},s-\{j\}} \in C_{k-1,\ell} - \tilde{C}_{k-1,\ell},$
$fr(A^{p\cup\{j\},s-\{j\}}) < 1 - \sigma\}$;

$\tilde{C}_{k,\ell} := \{A^{p,s} \in C_{k,\ell} / \forall j \in s, A^{p\cup\{j\},s-\{j\}} \notin C_{k-1,\ell}\}$;

$L_{k,\ell} := \{A^{p,s} \in C_{k,\ell} - \tilde{C}_{k,\ell} / \exists j \in s,\ fr(A^{p,s-\{j\}}) > \sigma + fr(A^{p\cup\{j\},s-\{j\}})\}$;

$k := k + 1$

fi

od (while k)

Acot-Freq-level

$k := 1$;

while $k \leq$min(ℓ,MaxNeg) **do**

if $k \notin J$ **then**

$L_{k,\ell} := L_{k,\ell} \cup (\tilde{L}_\ell \cap \tilde{C}_{k,\ell})$;

if $L_{k,\ell} = \emptyset$ **then** $J := J \cup \{k\}$ **fi**

else $L_{k,\ell} := \emptyset$

fi

$k := k + 1$

od(while k)

Apéndice B

Algoritmo teórico por plantas

A continuación, se presenta el algoritmo que realiza un recorrido en la estructura de itemsets por plantas para encontrar los itemsets frecuentes.

Algorithm planta-neg-Apriori

1. $L_{0,1} := \{A^{\{i\},\emptyset}, \forall i \in I / \, fr(A^{\{i\},\emptyset}) > \sigma\}$

2. set current floor $f := 0$
   set current level $\ell := 1$

3. **while** $L_{0,\ell} :\neq \emptyset$ **do**
   $\ell := \ell + 1$
   $C_{0,\ell} := \{\ell - \text{itemset}, \, A^{p,\emptyset} / \forall i \in p, \, A^{p-\{i\},\emptyset} \in L_{0,\ell-1}\}$
   $L_{0,\ell} := \{A^{p,\emptyset} \in C_{0,\ell} / \, fr(A^{p,\emptyset}) > \sigma\}$
   od (while)

4. **repeat**
   $f := f + 1$
   $C_{f,\ell} := \{\ell - \text{itemset}, \, A^{p,s} / \forall i \in p, A^{p-\{i\},s} \in L_{f,\ell-1},$

$\forall j \in s,\ A^{p,s-\{j\}} \in L_{f-1,\ell-1}\}$
$L_{f,\ell} := \{A^{p,s} \in C_{f,\ell} / \ fr(A^{p,s}) > \sigma\}$
if ($L_{f,f} := \emptyset$ **or** $\ell = n$) **then**
output $\bigcup_{f \leq \ell \leq n} L_{f,\ell}$
else
while $L_{f,\ell} :\neq \emptyset$ **do**
set current level $\ell := \ell + 1$
if $\sigma \leq 0,5$ **then**
for each $A^{p,s'} \in L_{f-1,\ell-1}$ **do**
for all $\{m\} \in I - \{p \cup s'\}$ **do**
$C_{f,\ell} := \{A^{p,s} / s = s' \cup \{m\},\ \forall i \in p, A^{p-\{i\},s} \in L_{f,\ell-1},$
$\forall j \in s, A^{p,s-\{j\}} \in L_{f-1,\ell-1}$
$L_{f,\ell} := L_{f,\ell} \cup \{A^{p,s} \in C_{f,\ell},\ / \forall j \in s,$
$fr(A^{p,s-\{j\}}) > \sigma + fr(A^{p\cup\{j\},s-\{j\}})\}$
od
od
else
for each $A^{p,s'} \in L_{f-1,\ell}$ **do**
for all $\{m\} \in I - \{p \cup s'\}$ **do**
$C_{f,\ell} := \{A^{p,s} / s = s' \cup \{m\},\ \forall i \in p, A^{p-\{i\},s} \in L_{f,\ell-1}, \forall j \in s,$
$A^{p,s-\{j\}} \in L_{f-1,\ell-1},\ \forall j \in s, fr(A^{p\cup\{j\},s-\{j\}}) < \sigma\}$
$L_{f,\ell} := L_{f,\ell} \cup \{A^{p,s} \in C_{f,\ell},\ / \forall j \in s,$
$fr(A^{p,s-\{j\}}) > \sigma + fr(A^{p\cup\{j\},s-\{j\}})\}$
od
od
fi
od (while)
fi
until $f < n$

5. output $\bigcup_{f \leq \ell \leq n} L_{f,\ell}$

Damos ahora unas consideraciones sobre la decisión de hacer el recorrido por niveles o por plantas.

Tanto en el algoritmo que recorre la estructura de itemsets por niveles y en el algoritmo que la recorre por plantas hemos incluido una poda para conocer a priori que itemsets no son candidatos. Esta poda se efectua en los algoritmos en distintos momentos, pero no se puede decir que un algoritmo sea mejor que otro en este sentido.

Supongamos que por razones de tiempo o por limitaciones en los costes se para la ejecución del algoritmo, la información que da cada algoritmo es diferente:

- En el algoritmo de búsqueda por plantas se obtienen itemsets con mayor número de atributos pero con un número pequeño o acotado de atributos negativos y
- en el algoritmo de búsqueda por niveles se obtienen itemsets con un menor número de atributos pero con mayor diversidad en los atributos negativos que en el caso por plantas.

El uso de un algoritmo u otro dependerá de si el usuario necesita itemsets con pocos atributos negativos o si por el contrario necesita itemsets con pocos atributos.

La diversidad de atributos negativos da más información en el siguiente sentido. En el algoritmo de búsqueda por niveles, en el nivel ℓ, tenemos toda la información sobre la frecuencia de todos los ℓ-itemsets. Sin embargo, en el algoritmo de búsqueda por plantas esta información con respecto a todos los itemsets de un mismo cardinal es limitada. Para conseguir esa información en el algoritmo de búsqueda por plantas tendremos que pasar por $\ell + 1$ plantas calculando frecuencias innecesarias si no queremos tener muchos atributos negativos.

Por ejemplo, supongamos que $fr(A\overline{B}) < \sigma$. Si $\sigma \leq 0,5$, no conocemos nada acerca de las frecuencias de $AB, \overline{A}B, \overline{A}\,\overline{B}$. Si $\sigma > 0,5$, sólo sabemos

que $1 - \sigma$ se reparte entre todos esos itemsets pero no sabemos en que forma.

Por estas razones y por seguir la filosofía de búsqueda del algoritmo Apriori, decidimos realizar todo el estudio para el caso de búsqueda por niveles. De todas formas, esta decisión no excluye a que se pueda usar si es conveniente el algoritmo de búsqueda por plantas que puede ser más intuitivo además de más simple en su estructura.

Bibliografía

[Adamo, 2000] ADAMO J.M. *Data mining and association rules and sequential patterns. Sequential and parallel algorithms.* New York: Springer-Verlag.

[Adrians y Zantinge, 1996] ADRIANS P., ZANTINGE D. *Data mining.* Addison-Wesley, Reading, MA.

[Agrawal et al., 1993] AGRAWAL R., IMIELINSKI T., SWAMI A.N. Mining association rules between sets of items in large databases. *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'93)*, Washington D.C., May 26-28. pp. 207-216. ACM Press.

[Agrawal y Srikant, 1994] AGRAWAL R., SRIKANT R. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 2th International Conference on Very Large Data Bases, Sept. 12-15, 1994, Santiago de Chile, Chile*, pp. 487-499. Morgan Kaufmann.

[Agrawal y Srikant, 1995] AGRAWAL R., SRIKANT R. Mining sequential patterns. *ICDE, Proc. of the Int'l Conf. on Data Engineering*, Taipei, Taiwan.

[Agrawal et al., 1996] AGRAWAL R., MANNILA H., SRIKANT R., TOIVONEN H., VERKAMO A.I. Fast discovery of association rules, in Fayyad U.M., Piatetsky-Shapiro G., Smyth P., Uthurusamy R. Eds, *Advances*

in Knowledge Discovery and Data Mining, pp. 307-328. AAAI Press, Menlo Park, CA.

[Agrawal y Yu, 1998] AGRAWAL C.C., YU P.S. A new framework for item set generation. *PODS'98, Seattle, WA*, pp. 18-24.

[Angluin, 1982] ANGLUIN D. Inference of reversible languages. *Journal of the ACM*, 29(3), pp. 741-765.

[Angluin, 1987] ANGLUIN D. Learning regular sets from queries and counterexamples. *Information and Computation*, 75 , pp. 87-106.

[Angluin, 1988] ANGLUIN D. Queries and concept learning. *Machine Learning*. 2, pp. 319-342.

[Baixeries et al., 2000] BAIXERIES J., CASAS-GARRIGA G., BALCÁZAR J.L. Frequent sets, sequences, and taxonomies: new, efficient algorithmic proposals. Tech. Rep. LSI-00-78-R. UPC.

[Balcázar et al., 1997] BALCÁZAR J.L., DÍAZ J., GAVALDÀ R., WATANABE O. Algorithms for learning finite automata from queries : A unified view, in *Advances in Algorithms, Languages and Complexity*. D. -Z. Du and K. -I. Ko (Eds.), Kluwer Academic Publishers, pp. 73-91.

[Bayardo et al., 1999] BAYARDO R.J., AGRAWAL R. Pruning and summarizing the discovered associations. *Proc. ACM-SIGKDD Conference*, pp. 125-134.

[Berger,1988] BERGER J. O. *Statistical decision theory and bayesian analysis*. New York: Springer Verlag.

[Berzal et al. 2001] BERZAL F., BLANCO I., SÁNCHEZ I., VILA M.A. A new framework to asses association rules. *IDA 2001, LNCS 2189*, pp. 95-104.

[Biermann y Feldman, 1972] BIERMANN A.W., FELDMAN J.A. On the synthesis of finite state machines from samples of their behaviour. *IEEE Transactions on Computers*, 21, pp. 592-597.

[Blake et al., 1998] BLAKE C., KEOGH E., 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.

[Borges y Levere, 1999] BORGES J., LEVERE M. Data Mining of user navigation patterns. *Proc. of the workshop in web usage Analysis and user Profiling*, pp. 92-111. Published by Springer-Verlag as LNCS vol 1836.

[Breiman et al., 1984] BREIMAN L., FRIEDMAN J. H., OLSHEN R. A., STONE C. J. *Classification and regression trees.* Belmont, CA: Wadsworth.

[Brin et al., 1997] BRIN S., MOTWANI R., ULLMAN J.D., TSUR S. Dynamic itemset counting and implication rules for market basket data. *Int. Conf. Management of Data*, pp. 255-264, ACM Press.

[Brin et al., 1997a] BRIN S., MOTWANI R., SILVERSTEIN C. Beyond market baskets: generalizing association rules to correlations. *SIGMOD'97*, AZ, pp. 265-276.

[Carrasco y Oncina, 1994] CARRASCO R.C., ONCINA J. *Grammatical inference and applications, vol 862 of Lecture Notes in Artificial Intelligence (subseries of LNCS)*, pp. 139-152. New York: Springer-Verlag.

[Castellanos et al., 1994] CASTELLANOS A. , GALIANO I., VIDAL E. *Grammatical inference and applications, vol. 862 of Lecture Notes in Artificial Intelligence (subseries of LNCS)*, pp. 93-105. New York: Springer-Verlag.

[Cestnik y Kononenko, 1987] CESTNIK B., KONONENKO I. ASSISTANT 86: A knowledge-elicitation tool for sophisticated users. *In I. Bratko*

and N. Lavrac (eds.), Progress in Machine Learning. Wilmslow, UK: Sigma Press.

[Chen et al., 1998] CHEN M. S., PARK J. S., YU P.S. Efficient data mining for traversal patterns. *IEEE Trans. on Knowledge and Data Eng. 10(2)*, pp. 209-221.

[Cook y Woolf, 1996] COOK J.E., WOLF A.L. Discovering models of software processes from event-based data. Technical Report CU-CS-820-96, Department of Computer Science, University of Colorado, Boulder, CO 803009 USA.

[Domingo et al., 1999] DOMINGO C., GAVALDÀ R., WATANABE O. Adaptative Sampling Methods for Scaling Up Knowledge Discovery Algorithms To appear in *Data Mining and Knowledge Discovery.* Preliminary version in *Proc. Second International Conference on Discovery Science* (DS'99), pp. 172-183.

[Fayyad et al., 1996] FAYYAD U.M., PIATETSKY-SHAPIRO G., SMYTH P. From data mining to knowledge discovery: An overview. In Fayyad U.M., Piatetsky-Shapiro G., Smyth P. and Uthurusamy R., eds, *Advances in Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA, pp. 1-34.

[Feller, 1971] FELLER W. *An Introduction to Probality Theory and Its Applications.* John Wiley & Sons, second edition.

[Fortes et al., 1998] FORTES I., MORALES BUENO R., PÉREZ DE LA CRUZ J.L., TRIGUERO F., COMINO M.A. Obtaining Mondrian-Style Paintings through Probabilistic Regular Grammars, *Proceedings of World Multiconference on Systemics, Cybernetics and Informatics (SCI´98), Vol. 2*, Orlando (U.S.A.), Julio, pp. 292-296.

[Fortes et al., 1998a] FORTES I., MORALES BUENO R., PÉREZ DE LA CRUZ J.L., TRIGUERO F., COMINO M.A. Inferencia de Gramáticas Regulares Probabilísticas a partir de Ejemplos, en *Actas de las IV*

Jornadas de Informática, Las Palmas de Gran Canaria (España), pp. 47-56.

[Fortes et al., 1998b] FORTES I., MORALES BUENO R., PÉREZ DE LA CRUZ J.L., TRIGUERO F., COMINO M.A. Inferencia de Gramáticas Regulares Difusas a partir de Ejemplos, *Actas del VIII Congreso Español sobre Tecnologías y Lógica Fuzzy* , Pamplona (España), pp. 91-96.

[Fortes et al., 1999] FORTES I., MORALES BUENO R., PÉREZ DE LA CRUZ J.L., TRIGUERO F., COMINO M.A. Fuzzy Regular Grammars from Examples. *Mathware Soft & Computing. Vol 6, 2-3*, pp. 277-291.

[Fortes et al,. 1999a] FORTES I., MORALES BUENO R., TRIGUERO F. Un Método Probabilístico para tratar en los Algoritmos de Aprendizaje Inductivos los Valores Desconocidos, en *Actas de la VII Conferencia de la Asociación Española para la Inteligencia Artificial CAEPIA '99, Vol. 1*, pp. 1-8.

[Fortes et al., 2000] FORTES I., MORALES BUENO R., PÉREZ DE LA CRUZ J.L., TRIGUERO F., GONZÁLEZ F.J. GRAMMARKOV: A Markov Chain Application to Probabilistic Linear Grammars, en *3ECM 2000 (Third European Congress of Mathematics)* Barcelona.

[Fortes et al., 2000a] FORTES I., MORALES BUENO R., MORA LL., TRIGUERO F. A decision theory approach to work with missing attribute values in inductive learning algorithms. *Proc. of COMPSTAT2000 (14TH Conference of the International Association for Statistical Computing)*, Utrecht.

[Fortes et al., 2001] FORTES I., BALCÁZAR J.L., MORALES BUENO R. Bounding Negative Information in Frequent Sets Algorithms, *LNAI 2226: 4th International Conference Discovery Science, DS 2001, Washington, DC, USA.*, pp. 50–58.

[Friedman, 1977] FRIEDMAN J. H. A recursive partitioning decision rule for non-parametric classification. *IEEE Transactions on Computers*, pp. 404-408.

[Fukuda et al., 1996] FUKUDA T., MORIMOTO Y., MORISHITA S., TOKUYAMA T. Mining optimized association rules for numeric attributes. *PODS'96, Proc. of the 15th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, Montreal, Canada.

[González y Thomason, 1978] GONZÁLEZ R.C., THOMASON M.G. *Syntactic patter recognition. An introduction.* Reading: Addison-Wesley Publishing Company.

[Gras, 1996] GRAS R. *L'Implication Statistique*, Nouvelle méthode exploratoire de données, Applications à la didactique, La Pensée Sauvage, Editions.

[Grimmet y Stirzaker, 1995] GRIMMET G.R., STIRZAKER D.R. *Probability and random processes.* New York: Oxford University Press Inc.

[Gunopulos et al., 1997] GUNOPULOS D., KHARDON R., MANNILA H., TOIVONEN H. Data mining, hypergraph transversals, and machine learning. *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 209-216, ACM Press, Tucson, Arizona, May, 12-14.

[Han et al., 2000] HAN J., PEI J., YIN Y. Mining frequent patterns without candidate generation. *Proc. of the 2000 ACM-SIGMOD Int'l Conf. on Management of Data*, Mayo, Dallas, Texas, USA.

[Heckerman, 1997] HECKERMAN D. Bayesian networks for data mining. *Data Mining and Knowledge Discovery.* Vol. 1 num. 1, pp. 79-120.

[Hipp et al., 1998] HIPP J., MYKA A., WIRTH R., GÜNTZER U. A new algorithm for faster mining of generalized association rules. *PKDD'98,*

Proc. of the 2nd European Symposium on Principles of Data Mining and Knowwledge Discovery, Septiembre, Nantes, Francia.

[Hipp et al., 2000] HIPP J., GÜNTZER U., NAKHAEIZADEH G. Algorithms for association rule mining-A general survey and comparison. *SIGKDD Explorations* Vol. 2, num. 1, pp. 58-64.

[Holte, 1993] HOLTE R. C. Very simple classification rules perform well on most commonly used datasets. *Machine Learning* 11, pp. 63-91.

[Honda et al., 1977] HONDA N. ET AL. F-Recognition of fuzzy languages en Gupta M.M. Saridis G. Gaines B.R (eds): *Fuzzy Automata and Decision Process,* Cap. 9. Amsterdam: North-Holland.

[Hopcroft y Ullman, 1979] HOPCROFT J.E., ULLMAN J.D. *Introduction to automata theory languajes and computation.* Reading, Massachusetts: Addison Wesley.

[Hunt et al., 1966] HUNT E. B.,MARIN J., STONE P.J. *Experiments in induction* New York: Academic Press.

[Hyalf y Rivest, 1976] HYALF L., RIVEST R. L. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters 5* 1, pp. 15-17.

[Kearns y Vazirani, 1994] KEARNS M.J., VAZIRANI U.V. *An introduction to computational learning theory.* Cambridge: MIT Press.

[Klir y Folger, 1988] KLIR G.J., FOLGER T.A. *Fuzzy sets, uncertainty and information.* London: Prentice Hall.

[Little y Rubin, 1987] LITTLE R. J. A., RUBIN D.B. *Statistical analysis with missing data.* John Wiley & Sons Inc.

[Mannila y Toivonen, 1997] MANNILA H., TOIVONEN H. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery.* **1(3)**, pp. 241-258.

[Mannila et al., 1997] MANNILA H., TOIVONEN H. Y VERKAMO I. Discovery of frequent episodes in event sequences. *Data Mining and Knowwledge Discovery*, **1(3)**, Noviembre.

[Ng et al., 1998] NG R., LAKSHMANAN L.S., HAN J., PANG A. Exploratory mining and pruning optimization of constrained associations rules. *Proc. of 1998 ACM SIGMOD Int'l Conf on Management of Data*, Junio, Seattlethe, Washington, USA.

[Quinlan, 1979] QUINLAN J. Discovering rules by induction from large collections of examples. *In D. Michie (ed.), Expert Systems in the Micro Electronic Age.*

[Quinlan, 1983] QUINLAN J. Learning efficient classification procedures *In R. S. Michlaski, J. G. Carbonell, T. M. Mitchell (eds.), Machine Learning: An Artificial Intelligence Approach.* Palo Alto, CA: Tioga Press.

[Quinlan, 1986] QUINLAN J. The effect of noise on concept learning. *In R. S. Michlaski, J. G. Carbonell, T. M. Mitchell (eds.), Machine Learning: An Artificial Intelligence Approach* (Vol. 2). San Mateo, CA: Morgan Kaufmann.

[Quinlan, 1986a] QUINLAN J. Induction of decision trees. *Machine Learning 1*, pp. 81-106.

[Quinlan, 1989] QUINLAN J. Unknown attribute values in induction. *Proceedings of the Sixth International Machine Learning Workshop*, pp. 164-168. San Mateo, CA: Morgan Kaufmann.

[Quinlan, 1992] QUINLAN J. *C4.5: Programs for machine learning.* Morgan Kaufmann, San Mateo, CA.

[Ramos y Morales, 1999] RAMOS G., MORALES R. Formalizacion de los algoritmos TDIDT y CIDIM. Techn. Report LCC-ITI 99/01. Dept. Computer Science. Malaga University.

[Ramos et al., 2000] RAMOS G., MORALES R., VILLALBA A. CIDIM. Control of induction of sample division method. Una mejora de los algoritmos TDIDT. Techn. Report LCC-ITI 97/08. Dept. Computer Science. Malaga University.

[Ramos, 2001] RAMOS G. Nuevos desarrollos en aprendizaje inductivo. Tesis Doctoral. Departamento de Lenguajes y Ciencias de la Computación. Universidad de Málaga.

[Ramos y Morales, 2000] RAMOS G., MORALES R. A new method for induction decision trees by sampling. *Neurocolt Workshop on Applications of Learning Theory* Bellaterra, Barcelona.

[Rastogi y Shim, 1999] RASTOGI R., SHIM K. Mining optimized support rules for numeric attributes. *Proc. of the 15th Int'l Conf on Data Engineering*, IEEE Computer Society Press, Marzo.

[Santos, 1977] SANTOS E.S. Regular fuzzy expression en Gupta M.M., Saridis G., Gaines B.R (eds): *Fuzzy Automata and Decision Process*, Cap. 10, pp. 169-175. Amsterdam: North-Holland.

[Savsere et al., 1995] SAVSERE A., OMIECINSKI E., NAVATHE S. An efficient algorithm for mining association rules in large databases. In *Proc. of the 21st International Conference on Very Large Databases*, pp. 432-444.

[Srikant y Agrawal, 1995] SRIKANT R., AGRAWAL R. Mining generalized association rules. *VLDB'95, Proc. of the 21st Conf on Very Large Databases*, Septiembre, Zürish, Switzerland.

[Srikant y Agrawal, 1996] SRIKANT R., AGRAWAL R. Mining quantitative association rules in large relational tables *Proceedings of the 1996 ACM SIGMOD Conf. on Management of Data,* Montreal, Canada, Junio .

[Srikant et al., 1997] SRIKANT R., VU Q. Y AGRAWAL R. Mining association rules with item constraints. *KDD'97, Proc. of the 3rd Int'l Conf on KDD and Data Mining*, Agosto, Newport Beach, California.

[Toivonen, 1996] TOIVONEN H. Sampling large databases for association rules. *Proceedings of the 22nd Conference on Very Large Databases*, pp. 134-145.

[Valiant, 1984] VALIANT L. G. A theory of the learnable. *Communications of the ACM.* 27(11), pp. 1134-1142.

[Wilson y Martinez, 1997] WILSON D. R., MARTINEZ T. R. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research* 6(1), pp. 1-34.

[Zaki et al., 1997] ZAKI M.J., PARTHASARATHY S., OGIHARA M., LI W. New algorithms for fast discovery of association rules. *KDD'97, Proc. of the 3rd Int'l Conf on KDD and Data Mining*, Agosto, Newport Beach, California.

Índice de Materias