Reglas de Asociación

Minería de Datos

José T. Palma

Departamento de Ingeniería de la Información y las Comunicaciones
Universidad de Murcia

DIIC, UMU, 2016

Contenidos de la presentación

Introducción

- Las reglas de asociación expresan patrones de comportamiento entre los datos en función de la aparición conjunta de valores de dos o más atributos.
  - A diferencia de los métodos de correlación, nos permiten establecer relaciones entre variables cualitativas.
- Aplicaciones:
  - Análisis de la cesta de la compra.
  - Estudio de textos.
  - Búsqueda de patrones en páginas web.
  - Diagnóstico médico y bioinformática.
- El concepto de regla de asociación fue introducido [Agrawal *et al.*, 1993] donde se aplica al análisis de la cesta de la compra.

Transacciones e itemsets

- Una **base de datos transaccional** hace referencia a una colección de transacciones.
- Cada transacción queda representada por el conjunto de **items** (artículos) incluidos en la misma.
- Sea $I = \{i_1, i_2, \ldots, i_d\}$ el conjunto de todos los items que pueden aparecer en una transacción.
  - Un **itemset** es un conjunto de items.
  - Si un itemset tiene k elementos se denomina $k - itemset$.
- Sea $T = \{t_1, t_2, \ldots, t_N\}$ el conjunto de todas las transacciones.
  - Una transacción está definida por el *itemset* que indica los items involucrados en la misma.

Transacciones e itemsets

- Existen distintas formas de representar una base de datos transaccional [Tan *et al.*, 2006]
  - Disposición horizontal de los datos.

TID	items
1	{Pan,Leche}
2	{Pan,Leche,Detergente,Cerveza,Huevos}
3	{Leche,Detergente,Cerveza,Cola}
4	{Pan,Leche,Detergente,Cerveza}
5	{Pan,Leche,Detergente,Cola}

Transacciones e itemsets

- Disposición vertical de los datos.

Pan	Leche	Detergente	Cerveza	Huevos	Cola
1	1	2	2	3	3
2	2	3	3		5
4	3	4	4		
5	4	5			
	5				

- Representación binaria de los datos.

TID	Pan	Leche	Detergente	Cerveza	Huevos	Cola
1	1	1	0	0	0	0
2	1	1	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	0	1	1	0	0	1

Soporte de un itemeset

Soporte

Sea $\mathcal{I} = \{i_1, i_2, \cdots, i_n\}$ el conjunto de todos los items, $\mathcal{T} \subseteq 2^I \setminus \{\emptyset\}$ el conjunto de transacciones. $X \in \mathcal{T}$ un itemset, y $\sigma(X)$ el número de transacciones en las que aparece X,

$$\sigma(X) = |\{t | X \subseteq t \wedge t \in \mathcal{T}\}|$$

El **soporte** de X es la fracción de transacciones que lo incluyen:

$$supp(X) = \frac{\sigma(X)}{N}$$

- Un **itemset frecuente**, es un *itemset* cuyo soporte es superior a un mínimo establecido, σ_{min}.

Soporte de un itemeset

TID	items
1	{Pan,Leche}
2	{Pan,Leche,Detergente,Cerveza,Huevos}
3	{Leche,Detergente,Cerveza,Cola}
4	{Pan,Leche,Detergente,Cerveza}
5	{Pan,Leche,Detergente,Cola}

Itemsets	Soporte
{Leche,Detergente}	4/5
{Leche,Detergente,Cerveza}	3/5
{Cerveza,Detergente}	3/5
{Leche,Cerveza}	3/5
{Leche,Cola}	2/5

Regla de asociación

- Sea $\mathcal{I} = \{i_1, i_2, \cdots, i_n\}$ el conjunto de todos los items, $\mathcal{T} \subseteq 2^I \setminus \{\emptyset\}$ el conjunto de transacciones
- Una **regla de asociación** es una implicación de la forma:

$$X \longrightarrow Y$$

- $X, Y \in \mathcal{T}$ son itemsets disjuntos ($X \cap Y = \emptyset$)
- Una regla de asociación no implica causalidad, sólo coocurrencia.
- Por ejemplo, la regla

$$\{Detergente\} \longrightarrow \{Cerveza\}$$

  indica que existe una fuerte relación entre la venta de detergente y cerveza.

Reglas de asociación: Soporte y confianza

- Sea la regla de asociación $X \longrightarrow Y$:

Soporte

El soporte de una regla de asociación es la fracción de transacciones en las que están incluidas los itemsets tanto del antecedente como del consecuente $(X \cup Y)$.

$$supp(X \longrightarrow Y) = supp(X \cup Y)$$

Confianza

La confianza de una regla de asociación es la fracción de transacciones en las que aparece el itemset X y que también incluyen al itemset Y.

$$conf(X \longrightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} = \frac{supp(X \cup Y)}{supp(X)}$$

Reglas de asociación: Soporte y confianza

- El soporte de una regla nos indica cuántas veces aplicable en el conjunto de transacciones.
- La confianza nos da una medida de lo frecuente que es que Y aparezca en una transacción que contiene X.

TID	items
1	{Pan,Leche}
2	{Pan,Leche,Detergente,Cerveza,Huevos}
3	{Leche,Detergente,Cerveza,Cola}
4	{Pan,Leche,Detergente,Cerveza}
5	{Pan,Leche,Detergente,Cola}

$$\{Leche, Detergente\} \longrightarrow \{Cerveza\}$$

$$\begin{aligned} supp(X \rightarrow Y) &= sup(\{Leche, Detergente, Cerveza\}) \\ &= 3/5 \end{aligned}$$

$$supp(\{Leche, Detergente\}) = 4/5$$

Definición del problema

Descubrimiento de reglas de asociación

Dados un conjunto de transacciones $\mathcal{T}$ encontrar todas las reglas de asociación que tengan :

- un soporte $\geq \sigma_{min}$, y
- una confianza $\geq \tau_{min}$

- Por lo tanto, para encontrar reglas de asociación debemos definir un soporte y una confianza mínimos.

Fuerza bruta

- Una primera aproximación puede consistir calcular el soporte y la confianza para todas las posibles reglas.
- Esto es impracticable, incluso para conjunto pequeños, ya que el número de posibles reglas para d items es:

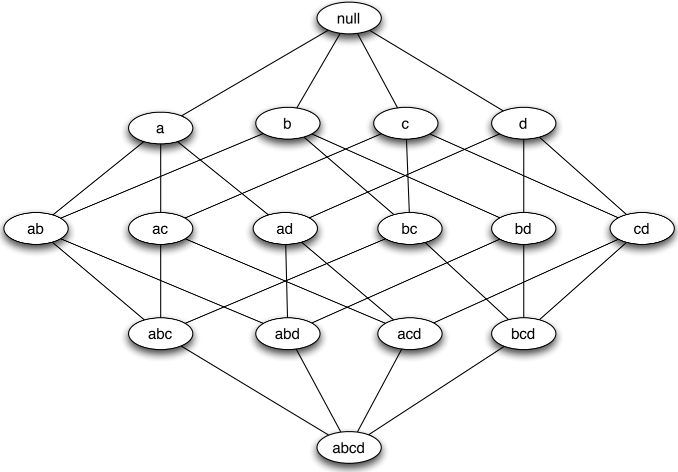
$$R = 3^d - 2^{d+1} + 1$$

- Dado que la confianza es directamente proporcional al soporte, una aproximación puede ser sólo generar reglas para *itemsets* frecuentes.
- Esto nos lleva a un proceso en dos pasos:
  1. Generación de *itemsets* frecuentes.
  2. Generación de reglas.

Generación de Itemsets frecuentes

- Si tenemos un conjunto de datos con k items se podrían generar $2^k - 1$ itemsets frecuentes.
- Evidentemente, con k grande el espacio de búsqueda se hace exponencialmente grande.
- La aproximación por fuerza fruta presenta un coste computacional de $O(NMw)$:
    - N es el número de transacciones, $M = 2^k - 1$ y w es la longitud máxima de todas las transacciones.

Generación de Itemsets frecuentes: Reticulo para los itmes $\{a, b, c\}$



Generación de Itemsets frecuentes

- Existen dos estrategias básicas para reducir la carga computacional:
  - **Reducir el número de itemsets candidatos**. Esta estrategia nos permitirá eliminar en número de candidatos posibles sin necesidad de calcular su soporte.
    - Esto se consigue aplicando el principio *Apriori* [Agrawal *et al.*, 1993, Agrawal and Srikant, 1994]
  - **Reducir el número de comparaciones**. Esta estrategia se basa en la utilización de estructuras de datos avanzadas que permiten reducir el número de comparaciones necesarias para la determinación de los itemsets frecuentes.

Generación de Itemsets frecuentes: Principio apriori

Principio Apriori

Sea $\mathcal{I} = \{i_i, i_2, \cdots, i_n\}$ un conjunto de items y $T \in 2^{\mathcal{I}} \setminus \{\emptyset\}$ un itemset frecuente, es decir, $supp(T) \geq \sigma_{min}$ entonces

$$\forall X \in 2^{T} \setminus \{\emptyset\} : supp(X) \geq supp(T)$$

- Es decir, todos los subconjuntos no vacíos de un itemset frecuentes son también frecuentes.
    - Es decir, si un itemset T es frecuente, cualquier adición de items al mismo también ser frecuente (al menos aparecerá en todas las transacciones en las que aparece T).

Generación de Itemsets frecuentes: Antimonotonicidad

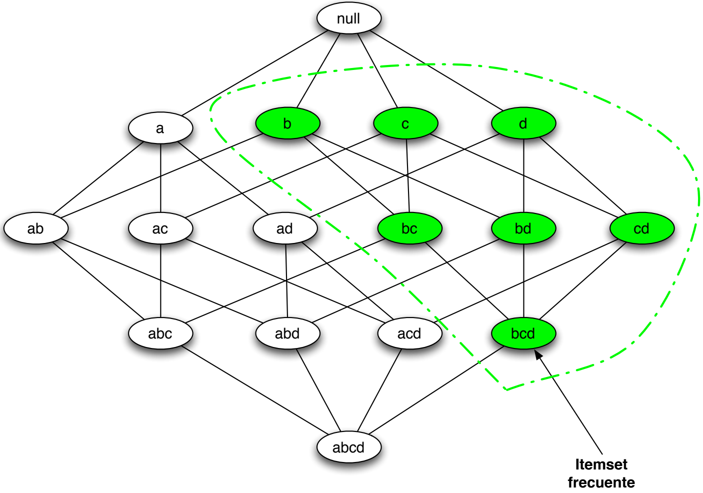
Antimonotonicidad

Sea $\mathcal{I} = \{i_i, i_2, \cdots, i_n\}$ un conjunto de items y una medida f es antimonótona si

$$\forall X, Y \in 2^{\mathcal{I}} \setminus \{\emptyset\} : (X \subseteq Y) \rightarrow f(Y) \leq f(X)$$

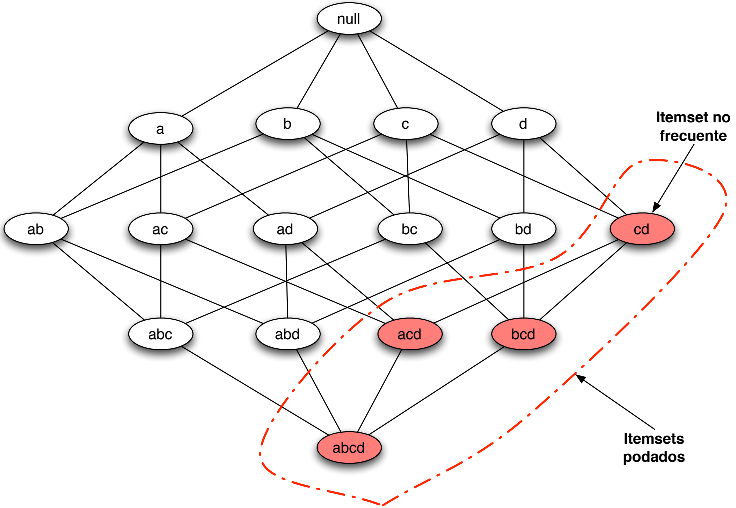
- El **soporte** es una medida **antimonótona**.
- Es decir, si un itemset I no es frecuente, cualquier adición de items al mismo no será frecuente (no puede aparecer en más transacciones en las que aparece I).
  - Esta propiedad puede ser usada para podar el espacio de búsqueda utilizando el soporte.

Generación de Itemsets frecuentes: Principio apriori



Si {b,c,d} es frecuente todos sus subconjuntos son frecuentes

Generación de Itemsets frecuentes: Poda



Algoritmo Algoritmo Apriori: Generación de itemsets frecuente

$k = 1;$
$F_k = \{i | i \in \mathcal{I} \wedge \sigma(\{i\}) \geq N \times \sigma_{min}\}$ ▷ Encontrar itemsets frecuentes
repetir
    $k = k + 1;$
    $C_k = Apriori\_Gen(F_{k-1});$ ▷ Generar itemsets candidatos
    para cada transacción $t \in T$ **hacer**
        $C_t = subset(C_k, t)$ ▷ itemsets candidatos en t
        para cada itemset $c \in C_t$ **hacer**
            $\sigma(c) = \sigma(c) + 1$ ▷ Incrementar el contador de soporte
        fin para
    fin para
    $F_k = \{c | c \in C_k \wedge \sigma(c) \geq N \times \sigma_{min}\}$
hasta que $F_k = \emptyset$
Devolver $\bigcup F_k$

Algoritmo Apriori: Generación de itemsets frecuente

- Sea C_k el conjunto de $k-$itemsets candidatos y F_k el conjunto de $k-$itemsests frecuentes.
- El algoritmo comienza con un barrido sobre los datos para determinar los $1-$itemsets frecuentes, F_1 (línea 2).
- Seguidamente, y de forma iterativa, se van generando k-itemsets candidatos a partir de los $(k-1)$ itemsets frecuentes (linea 5).
- En el siguiente paso (bucle 8-10) se vuelven a recorrer los datos para calcular el soporte de cada $k-$itemset candidato.
- Del conjunto C_k se seleccionan aquellos $k-$itemsets que son frecuentes, conjunto F_k (linea 12).
- El proceso termina cuando yo no se generan más $k-$itemsets frecuentes (linea 13) y se devuelven todos los itemsets frecuentes de tamaños 1 hasta k.

Algoritmo Apriori: Generación de itemsets frecuente

- El algoritmo que acabamos de ver tiene dos propiedades importantes:
  - Recorre el retículo nivel por nivel, desde los $1-$itemsets hasta los itemsets de tamaño máximo.
  - Utiliza una estratégica de generación y prueba.
    - En cada iteración se generan un conjunto de itemsets candidatos a partir de los itemsets frecuentes de iteraciones anteriores.
    - Después se calcula el soporte para todos los itemsets candidatos para buscar los frecuentes.

Algoritmo Apriori: Generación de candidatos y poda

- La generación de itemsets candidatos se realiza en dos fases:
  1. **Generación de candidatos**. En esta fase se generan los k-itemsets candidatos a partir de los $(k-1)$-itemsets frecuentes obtenidos en la iteración previa.
  2. **Poda de candidatos**. Se eliminan los k-itemsets candidatos cuyo soporte sea inferior a σ_{min}.

Algoritmo Apriori: Generación de candidatos y poda

- La generación de candidatos debe cumplir los siguientes requisitos:
  - Se debe evitar la generación de candidatos no necesarios $\longrightarrow$ aplicar la propiedad de **antimonotonicida**.
  - Hay que garantizar que el conjunto de candidatos sea completo $\longrightarrow \forall k : F_k \subseteq C_k$.
  - Se debe de evitar la repetición de candidatos.
- Existen varias estrategias para la generación de candidatos:
  - Fuerza bruta.
  - Método $F_{k-1} \times F_1$.
  - Método $F_{k-1} \times F_{k-1}$.

Generación de candidatos y poda: Fuerza bruta

- En este caso, se consideran que todos los $k-$itemsets posibles como candidatos.
  - El número de posibles candidatos es $\binom{d}{k}$, siendo d el número de items.
- El proceso de poda eliminará los candidatos no frecuentes.
- El cálculo de los $k-$itemsets candidatos es trivial, pero el proceso de poda es muy costoso
  - El cálculo del soporte para cada candidato es $O(k)$.
  - La complejidad total sería $O(\sum_{k=1}^{d} \times \binom{d}{k}) = O(d2^{d-1})$

Generación de candidatos y poda: $F_{k-1} \times F_1$

- La idea básica es la de extender cada $(k-1)-$itemset frecuente con un 1-itemsets frecuente.
- En este caso el número de itemsets candidatos generados será $O(|F_{k-1}||F_1|)$
  - El coste total será $O(\sum_k k|F_{k-1}||F_1|)$
- El método es completo.
- Inconvenientes:
  - Se pueden generar itemsets duplicados $\longrightarrow$ ordenar alfabéticamente los items en los itemsets.
  - Puede producir una gran cantidad de itemsets innecesarios

Generación de candidatos y poda: $F_{k-1} \times F_1$

C_1

1-itemsets.	Sop.
Cerveza	3
Cola	2
Detergente	4
Huevos	1
Leche	5
Pan	4

F_1

1-itemsets.	Sop.
Cerveza	3
Detergente	4
Leche	5
Pan	4

$C_2 \leftarrow (F_1, F_1)$

2-itemsets.	Sop.
$\{Cerveza, Detergente\}$	3
$\{Cerveza, Leche\}$	3
$\{Cerveza, Pan\}$	2
$\{Detergente, Leche\}$	4
$\{Detergente, Pan\}$	3
$\{Leche, Pan\}$	4

F_2

2-itemsets.	Sop.
$\{Cerveza, Detergente\}$	3
$\{Cerveza, Leche\}$	3
$\{Detergente, Leche\}$	4
$\{Detergente, Pan\}$	3
$\{Leche, Pan\}$	4

$C_3 \leftarrow (F_2, F_1)$

3-itemsets.	Sop.
$\{Cerveza, Detergente, Leche\}$	3
$\{Cerveza, Detergente, Pan\}$	2
$\{Cerveza, Leche, Pan\}$	2
$\{Detergente, Leche, Pan\}$	3

F_3

3-itemsets.	Sop.
$\{Cerveza, Detergente, Leche\}$	3
$\{Detergente, Leche, Pan\}$	3

Itemsets frecuentes:
$F_1 \cup F_2 \cup F_3$

Generación de candidatos y poda: $F_{k-1} \times F_{k-1}$

- Se fusionan dos $(k-1)-$itemsets frecuentes cuyos primeros $k-2$ items son idénticos, supuesto un orden alfabético en los items.
- El método es completo.
- El orden alfabético asegura que no se generan itemsets duplicados.
- Se generan menos itemsets candidatos pero sigue siendo necesaria la fase de poda.

Generación de candidatos y poda: $F_{k-1} \times F_{k-1}$

C_1

1-itemsets.	Sop.
Cerveza	3
Cola	2
Detergente	4
Huevos	1
Leche	5
Pan	4

F_1

1-itemsets.	Sop.
Cerveza	3
Detergente	4
Leche	5
Pan	4

$C_2 \leftarrow (F_1, F_1)$

2-itemsets.	Sop.
$\{Cerveza, Detergente\}$	3
$\{Cerveza, Leche\}$	3
$\{Cerveza, Pan\}$	2
$\{Detergente, Leche\}$	4
$\{Detergente, Pan\}$	3
$\{Leche, Pan\}$	4

F_2

2-itemsets.	Sop.
$\{Cerveza, Detergente\}$	3
$\{Cerveza, Leche\}$	3
$\{Detergente, Leche\}$	4
$\{Detergente, Pan\}$	3
$\{Leche, Pan\}$	4

$C_3 \leftarrow (F_2, F_2)$

3-itemsets.	Sop.
$\{Cerveza, Detergente, Leche\}$	3
$\{Detergente, Leche, Pan\}$	3

F_3

3-itemsets.	Sop.
$\{Cerveza, Detergente, Leche\}$	3
$\{Detergente, Leche, Pan\}$	3

Itemsets frecuentes:

$F_1 \cup F_2 \cup F_3$

Algoritmo Apriori: Cálculo del soporte

- Hay que comparar cada transacción con cada itemset candidato
    - Computacionalmente muy costoso.
- Para resolverlo:
    - Calcular todos los itemsets de cada transacción.
    - Buscar en dicha lista los itemsets candidatos.
    - Esto se puede optimizar combinando una estructura de árbol (indexada por los prefijos) y una tabla hash.
- También se pueden eliminar del conjunto de $k-$itemsets candidatos aquellos en los que algunos de sus subconjuntos no esté en F_{k-1}.
    - Existen algoritmos eficientes para detectar este hecho.

Algoritmo Apriori: Cálculo del soporte

- Existen otras técnicas que se basan en no calcular el soporte en todas las fases de generación de candidatos.
  - **Fase Forward**. Se van generando los distintos conjuntos de $k-$itemsets candidatos. Sólo se detectan los frecuentes para determinadas longitudes.
  - **Fase Backward**. Se calcula el soporte para el resto itemsets no considerados, eliminando previamente aquellos itemsets que son subconjuntos de algún itemset frecuente.
- Este último proceso encuentra los itemsets maximales.

Algoritmo Apriori: Itemsets maximales y cerrados frecuentes

Itemsets maximales frecuentes

Un itemset frecuente es **maximal** si ninguno de sus superconjuntos propios es frecuente. Es decir, sea $\mathcal{I}$ el conjunto de todos los items y $\mathcal{F} \subseteq 2^{I} \setminus \{\emptyset\}$ el conjunto de todos los itemsets frecuentes, es decir, $\forall X \in \mathcal{F}\colon supp(X) \geq \sigma_{min}$, un itemset $X \in \mathcal{F}$ es maximal si

$$\nexists Y \in 2^{\mathcal{I}} \setminus \{\emptyset\} \wedge Y \supset X\colon Y \in \mathcal{F}$$

- Constituyen el conjunto más pequeño de itemsets frecuentes a partir de cual se pueden generar el resto de itemsets frecuentes..
- Por lo tanto, proporcionan una representación compacta en el caso de tener muchos itemsets frecuentes.
  - Existen algoritmos eficientes para detectarlos.

Algoritmo Apriori: Itemsets maximales y cerrados frecuentes

Itemsets cerrados frecuentes

Un itemset es **cerrado** si ninguno de sus superconjuntos tiene exactamente su mismo soporte. Es decir, sea $\mathcal{I}$ el conjunto de todos los items, un itemset $X \in 2^{\mathcal{I}} \setminus \{\emptyset\}$ es cerrado si

$$\nexists Y \in 2^{\mathcal{I}} \setminus \{\emptyset\} \wedge Y \supset X : suppY = supp(X)$$

Decimos que $X \in I$ es un **itemset frecuente cerrado** si X es un itemset cerrado y frequente.

$$supp(X) > \sigma_{min}$$

- Dicho de otra forma, un itemset no es cerrado si al menos uno de sus superconjuntos inmediatos tiene su mismo soporte.
- Existen métodos eficientes para calcular, a partir de los itemsets cerrados frecuentes, el soporte de los itemsets no cerrados

Algoritmo Apriori: Itemsets maximales y cerrados frecuentes

- **Diferencias:**
  - Los **temsets frecuentes cerrados** constituyen una forma compacta de representar los itemsets frecuentes y su soporte
  - Los **itemsets maximales** sólo representan los itemsets frecuentes más grandes, descartando sus subconjuntos que son frecuentes.
- **Relaciones:**
  - Todo itemset maximal es también cerrado, pero no todo itemset cerrado es maximal
  - Los conjuntos cerrados proporcionan una representación sin pérdida de todos los conjuntos de ítems frecuentes (es decir, se puede reconstruir su soporte), mientras que los conjuntos maximales pueden perder parte de esta información.

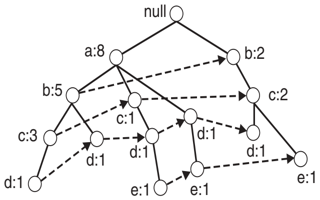
Algoritmo FP-Growth

- Como hemos visto el algoritmo Apriori tiene dos inconvenientes importantes
  - Puede generar un número grande de itemsets candidatos.
  - Puede necesitar realizar varias pasadas a la base de datos.
- El método **frequent-pattern growth** (FT-Growth) [Han *et al.*, 2000, Han *et al.*, 2004] intenta evitar estos inconvenientes.
- Se basa en una estrategia divide y vencerás:
  1. Se compacta la base de datos utilizando un FP-tree (frequent pattern tree).
  2. Se divide la base compactada en bases de datos condicionales y extraen los itemsets frecuentes de ellas.

Algoritmo FP-Growth: Árbol-FP

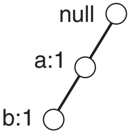
- Cada nodo del árbol-FP representa un item y tiene asociado un contador con el número de transacciones en las que el camino hasta el nodo está presente
- El nodo raíz es el nodo *nulo*.
- También existen enlaces entre los nodos que representan items idénticos.

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}



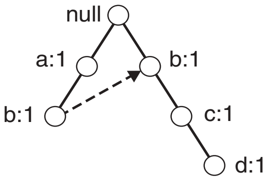
Transacciones y su correspondiente árbol-FP [Tan *et al.*, 2006]

Algoritmo FP-Growth: Construcción del árbol-FP



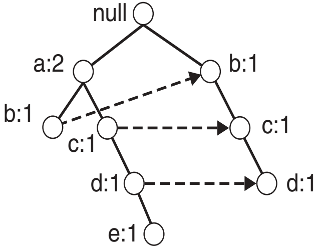
- Primero se determina el soporte de cada item y se eliminan los no frecuentes.
- Se ordenan los items frecuentes en orden descendente según su soporte en cada transacción.
- Ahora se realiza otro barrido de la base de datos para analizar cada transacción.
- La primera transacción $\{a, b\}$ da lugar al camino $null \rightarrow a \rightarrow b$.
- A cada uno de los contadores se les asigna el valor 1.

Algoritmo FP-Growth: Construcción del árbol-FP



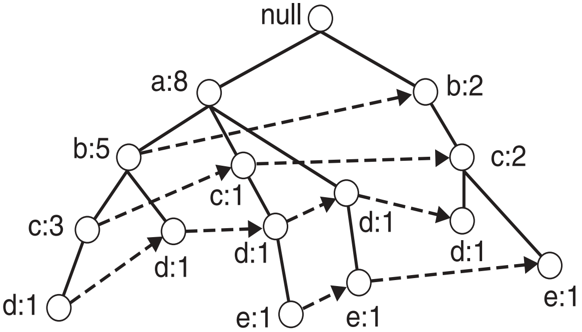
- La sengunda transacción $\{b, c, d\}$ da lugar a una nueva rama $null \rightarrow b \rightarrow c \rightarrow d$.
- Los contadores también se inicializan a 1.
- Aunque comparten el item b, se generan ramas diferentes al no compartir el prefijo.
- También se crea un enlace entre los nodos que representan el item b.

Algoritmo FP-Growth: Construcción del árbol-FP



- La tercera transacción, $\{a, c, d, e\}$, comparte el prefijo a con la primera transacción.
- Por lo tanto, al rama $null \rightarrow a \rightarrow c \rightarrow d \rightarrow e$ compartirán el nodo etiquetado con a.
- Esto hace que el contador del nodo a se incremente en 1.
- También se crearán enlaces entre los nodos c y los nodos d.
- El proceso continuaría hasta que se hallan procesado todas las transacciones.

Algoritmo FP-Growth: Construcción del árbol-FP



Árbol-FP final.

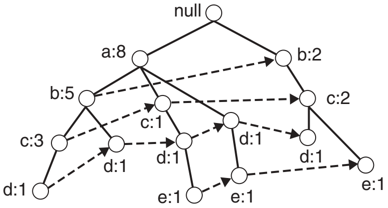
Algoritmo FP-Growth: Construcción del árbol-FP

- El tamaño del árbol-FP es menor que la base de datos original.
- En el caso más favorable el árbol-FP contiene sólo una rama.
    - Todas las transacciones tienen los mismos items.
- En el caso más desfavorable, el árbol-FP tiene una rama por transacción.
- Depende del orden en el que se recorran los items.
    - Con un orden descendente del soporte se obtienen árboles más compactos.

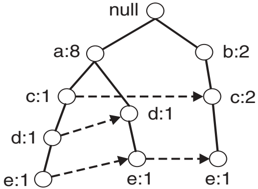
Algoritmo FP-Growth: Generación de itemsets frecuentes

- La generación de itemsets frecuentes se realiza recorriendo el árbol-FP en sentido ascendente.
  - Se empieza buscando los itemsets frecuentes acabados en *e*, después los acabados en *de*, en *ce*, ... en *d*, *cd*,. ..
  - Para localizar todas los itemsets acabados en *e* sólo hace falta recorrer las ramas que acaban en *e*, proceso eficiente gracias a los enlaces entre todos los nodos *e*.

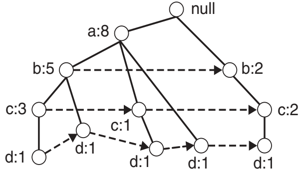
Algoritmo FP-Growth: Generación de itemsets frecuentes



Árbol-FP completo



Subárbol-FP con ramas con el item e



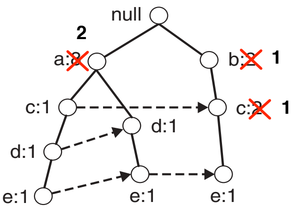
Subárbol-FP con ramas con el item d

null

Algoritmo FP-Growth: Generación de itemsets frecuentes

- Supongamos que queremos calcular todos los itemsets frecuentes que acaban en e.
- Primero debemos determinar si el item e es frecuente.
  - Para ello sólo debemos sumar los contadores asociados a los nodos e.
  - Si suponemos un soporte de 2, el item e es frecuente al ser la suma de sus contadores 3.
  - Por lo tanto, $\{e\}$ es un itemset frecuente.
- Seguidamente buscaríamos los itemsets frecuentes acabados en e, después los acabado en de, seguidos por los acabados en ce, be y ae.

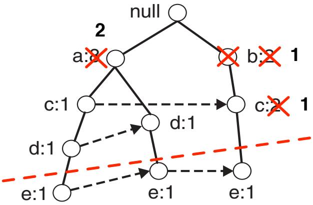
Algoritmo FP-Growth: Generación de itemsets frecuentes

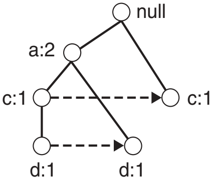


$$F_e = \{e\}$$

- Primero debemos construir un **árbol-FP condicional** a partir del subárbol-FP con las ramas que acaban en e.
  1. Se actualizan los contadores ya que estos incluyen transacciones que no contienen e.
     - La transacción $\{b, c\}$ y algunas que empiezan por a no contienen al item e.
- Es el árbol-FP que se hubiera construido si se eliminan todas las transacciones que no contienen e y, del resto, eliminamos el item e.

Algoritmo FP-Growth: Generación de itemsets frecuentes

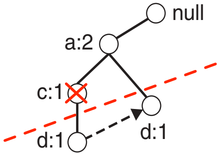


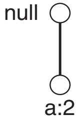


$$F_e = \{\{e\}\}$$

2. Se eliminan los nodos que contienen *e* (ya no son necesarios).
3. Se eliminan los nodos no frecuentes, en este caso *b*.
4. FP-growth usa este árbol-FP condicional para *e* para encontrar los itemsets frecuentes acabados en *de*, *ce* y *ae*.

Algoritmo FP-Growth: Generación de itemsets frecuentes

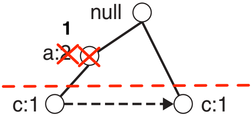




$F_e = \{\{e\}, \{d, e\}, \{a, d, e\}\}$

5. Para encontrar los itemsets frecuentes acabados *de* hay que generar el subárbol-FP acabado en *d* a partir del árbol-FP condicional de *e*.
6. Sumando el soporte de los nodos *d*, que es igual a 2, determinamos que $\{d, e\}$ también es frecuente.
7. Ahora se construye el subárbol-FP condicional para el *de*.
8. Como este árbol condicional sólo tiene un nodo, *a*, con soporte mayor que *minSup*, el itemset $\{a, d, e\}$ también es frecuente.

Algoritmo FP-Growth: Generación de itemsets frecuentes



$F_e = \{\{e\}, \{d, e\}, \{a, d, e\}, \{c, e\}, \{a, e\}\}$

10. Ahora se procede con buscar los acabados en *ce*.
11. Se crea el subárbol-FP acabado en *c* a partir del árbol condicional de *e*.
12. Se determina que el nodo *c* es frecuente, siendo el itemset $\{c, e\}$ frecuente.
13. A partir de este subárbol-FP se genera el árbol-FP condicional para la terminación *ce*.
14. Al ser un árbol vacío se procedería a encontrar los itemsets frecuentes acabados en *ae*, $\{a, e\}$.

Algoritmo FP-Growth: Generación de itemsets frecuentes

- Siguiendo con el proceso para los itemsets acabados en d,c,b y a, tendríamos los siguientes itemsets frecuentes:

Sufijo	Itemsets frecuentes
e	$\{e\}, \{d, e\}, \{a, d, e\}, \{c, e\}, \{a, e\}$
d	$\{d\}, \{c, d\}, \{b, c, d\}, \{a, c, d\}, \{b, d\}, \{a, b, d\}, \{a, d\}$
c	$\{c\}, \{b, c\}, \{a, b, c\}, \{a, c\}$
b	$\{b\}, \{a, b\}$
a	$\{a\}$

Algoritmo FP-Growth: Conclusiones

- **Ventajas**:
  - Sólo se requieren dos pasadas a la base de datos.
  - Comprime los datos.
  - No se generan itemsets frecuentes duplicados al ser los subproblemas disjuntos.
  - El más rápido que Apriori $\rightarrow$ no requiere la generación de candidatos.
  - "Comprime" los datos.

Algoritmo FP-Growth: Conclusiones

- **Desventajas**:
  - El árbol-FP puede que no quepa en memoria.
  - La construcción de una árbol-FP es muy costosa.
    - Sin embargo, una vez construido encontrar los itemsets frecuentes es muy fácil.
    - Sólo se podan items individuales y no itemsets.
    - El soporte sólo se puede calcular una vez construido el árbol-FP.

Generación de reglas de asociación

- Hasta ahora, dado un conjunto de items $\mathcal{I}$ hemos generado todos los los itemsets frecuentes, $\mathcal{F} \subseteq 2^{\mathcal{I}} \setminus \{\emptyset\}$: $(\forall X \in \mathcal{F}: supp(X) \geq \sigma_{min})$
- El siguiente paso consiste en generar las reglas de asociación a partir de los itemsests frecuentes $\mathcal{F}$.
- Cada itemset frecuente, $X \in \mathcal{F}$, pude generar $2^k - 2$ reglas de asociación.
- Para crear una regla de asociación a partir del itemset frecuente $X \in \mathcal{F}$,
  1. Dividir X en dos conjuntos disjuntos Y y $X \setminus Y$.
  2. Generar la regla $Y \rightarrow X \setminus Y$ si estas supera la confianza mínima, τ_{min}.
  3. Las reglas $Y \rightarrow \emptyset$ y $\emptyset \rightarrow Y$ no se tienen en cuenta.

Generación de reglas de asociación

- Ejemplo: Sea el itetset frecuente $\{a, b, c\}$. A partir de él se pueden generar las siguientes reglas de asociación:
  - $\{a, b\} \rightarrow \{c\}, \{a, c\} \rightarrow \{b\}, \{b, c\} \rightarrow \{a\}$
  - $\{a\} \rightarrow \{b, c\}, \{b\} \rightarrow \{a, c\}, \rightarrow \{b\}, \{c\} \rightarrow \{a, b\}$
- Evidentemente, el soporte de cada regla supera el mínimo establecido al ser generada a partir de un itemset frecuente.
- El cálculo de la confianza de las reglas no requiere volver a reocorrer la base de datos.

$$Conf(X \rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$$

Generación de reglas de asociación: Poda basada en la confianza

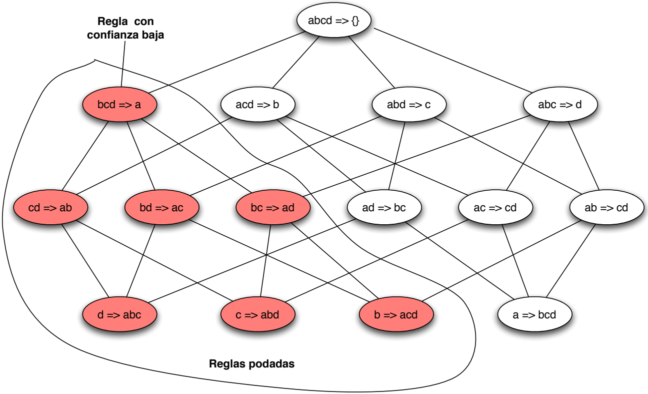
- La medida de confianza no cumple ninguna propiedad de monotonicidad.
- Sin embargo, dado un itemset frecuente $X \in \mathcal{F}$ e $Y \subset X$,

SI la regla la confianza $Y \rightarrow X \setminus Y$ no supera el umbral, entonces cualquier regla $Y' \rightarrow X \setminus Y'$, con $Y' \subseteq Y$, tampoco superará el umbral, es decir,

$$conf(Y \rightarrow X \setminus Y) \leq \tau_{min} \Rightarrow$$
$$\forall Y' \in 2^X \setminus \{\emptyset\},\ Y' \subseteq Y : conf(Y' \rightarrow X \setminus Y') \leq \tau_{min}$$

- Esto nos permite podar un conjunto de reglas.

Generación de reglas de asociación: Poda basada en la confianza



Generación de reglas de asociación: Algoritmo apriori

- El algoritmo Apriori para la generación de reglas sigue un proceso por niveles.
  - Cada nivel se corresponde con el número de items en el consecuente.
- Se comienza con las reglas que tienen un item en el consecuente y nos quedamos con las que tienen una confianza alta.
- Estas reglas se utilizan para generar las reglas del siguiente nivel.

$$\left.\begin{array}{l} \{acd\} \rightarrow \{b\} \\ \\ \{abd\} \rightarrow \{c\} \end{array}\right\} \Longrightarrow \{ad\} \rightarrow \{bc\}$$

Algoritmo Apriori: Generación de reglas en Apriori

1: **para cada** $k-$itemset frecuente f_k, $k \geq 2$ **hacer**
2: $H_1 = \{i | i \in f_k\}$ ▷ consecuentes de un item
3: **ejecutar** ap-genrules(f_k, H_1)
4: **fin para**

Algoritmo ap-genrules(f_k, H_m)

$k = |f_k|$; ▷ Tamaño del itemset frecuente
$m = |H_m|$ ▷ Tamaño del consecuente
si $k > m + 1$ **entonces**
  $H_{m+1} =$apriori-gen(H_m)
  para cada $h_{m+1} \in H_{m+1}$ **hacer**
    $conf = \sigma(f_k)/\sigma(f_k \setminus h_{m+1})$
    si $conf \geq \tau_{min}$ **entonces**
      generar la regla $(f_k \setminus h_{m+1}) \rightarrow h_{m+1}$
    sino
      $H_{m+1} = H_{m+1} \setminus h_{m+1}$
    fin si
  fin para
  ap-genrules(f_k, H_{m+1})
fin si

Evaluación de reglas de asociación

- Los métodos que hemos analizado tienden a generar una gran cantidad de reglas de asociación.
  - En aplicaciones reales podemos estar hablando de miles o millones de reglas.
- Es muy importante tener un criterio que permita evaluar las reglas obtenidas.
- Podemos utilizar criterios objetivos derivados de la estadística.
- También podemos utilizar criterios subjetivos que dependen del dominio.
  - Técnicas de visualización.
  - Extracción basada en plantillas.

Evaluación de reglas de asociación

- Muchas de las medidas se calculan a través de la tabla de contingencia:

	q	$\bar{q}$	
p	f_{11}	f_{10}	f_{1+}
$\bar{p}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

- $\bar{p}$ ($\bar{q}$) indica el que el item p (q) no está en la transacción.
- f_{11} indica el número de transacciones en las que p y q están presentes.
- f_{1+} representa el soporte de p y f_{+1} el soporte de q.

Medidas objetivas: Soporte y confianza

- Son las medidas que hemos analizado anteriormente.
- El soporte puede presentar problemas en los casos en distribuciones asimétricas de itemsets.
  - Supongamos la siguiente distribución de itemsets

Grupo	G_1	G_2	G_3
Soporte	$\leq 1\,\%$	$1\,\% - 90\,\%$	$> 90\,\%$
Nº items	1730	349	22

  - Con un soporte del 20 % podemos perder reglas interesantes (pueden corresponder con productos caros).

Medidas objetivas: Soporte y confianza

- Un soporte demasiado bajo complica el descubrimiento de reglas:
  - Aumentan los requisitos de memoria y computación.
  - Se obtienen muchas reglas.
  - Se pueden obtener reglas espúreas, que relacionen items muy frecuentes con otros muy poco frecuentes.
- Reglas de una confianza alta se pueden perder debido a que la confianza no tiene en cuenta el soporte del consecuente.

Medidas objetivas:Lift

- En algunos casos las reglas con un confianza alta nos pueden llevar a confusión ya que se ignora el soporte del itemset en el consecuente. Una forma de evitar esto consiste en utilizar el **lift**
- La medida **lift** se obtiene de la siguiente manera:

$$Lift = \frac{conf(p \rightarrow q)}{supp(q)}$$

- Para variables binarias el **lift** equivale a **factor de interés**:

$$I(p,q) = \frac{sup(p,q)}{sup(p)sup(q)} = \frac{Nf_{11}}{f_{1+}f_{+1}}$$

- El **factor de interés** compara la frecuencia de la regla con la frecuencia base asumida la independencia lineal entre los items.

Medidas objetivas: Lift y factor de interés

- El factor de interés se puede intrepretar de la siguiente manera:

$$I(p,q) = \begin{cases} < 1 & p \text{ y } q \text{ estan correlacionadas negativamente;} \\ = 1 & p \text{ y } q \text{ son independientes;} \\ > 1 & p \text{ y } q \text{ estan correlacionadas positivamente;} \end{cases}$$

- Puede darse el caso de que asociaciones muy frecuentes tengan un factor de interés cercano a 1 debido a las distribuciones de las ocurrencias.
  - Sobre todo si el peso de la matriz de contingencia se concentra en el término f_{11}

Medidas objetivas: Análisis de la correlación

- Esta medida está basada en técnicas estadísticas para medir la relación entre dos variables.
- Para variables continuas se suele utilizar el coeficiente de correlación de Pearson.
- Para variables binarias se utiliza el coeficiente ϕ:

$$\phi = \frac{f_{11} f_{00} - f_{01} f_{10}}{\sqrt{f_{1+1} f_{+1} f_{0+} f_{+0}}}$$

Medidas objetivas: Análisis de la correlación

- Este valor varía entre -1 (correlación negativa perfecta) y +1 (correlación positiva perfecta).
- Si las variables son estadísticamente independientes su valor es 0.
- El problema de esta medida es que da la misma importancia a la coocurrencia que a la coausencia.
- Por lo tanto, es muy útil cuando se están analizando variables binarias simétricas.
- Otra limitación es que se ve afectada por cambios proporcionales en el tamaño de la muestras.

Medidas objetivas: Medida IS

- La medida IS se propuso para tratar con variables binarias asimétricas:

$$IS(p,q) = \sqrt{I(p,q)sup(p,q)} = \frac{sup(p,q)}{\sqrt{sup(p)sup(q)}}$$

- Este valor es alto cuando el factor de interés y el soporte son altos.
- La medida IS es equivalente a la mediad basada en el coseno para variables binarias.

$$IS(p,q) = \frac{sup(p,q)}{\sqrt{sup(p)sup(q)}} = \frac{\mathbf{p}\cdot\mathbf{q}}{|\mathbf{p}||\mathbf{q}|} = \cos(\mathbf{p},\mathbf{q})$$

- La medida IS también equivale a la media geométrica de la confianza de la regla de asociación:

$$IS(p,q) = \sqrt{\frac{sup(p,q)}{sup(p)}\frac{sup(p,q)}{sup(q)}} = \sqrt{conf(p \rightarrow q)conf(q \rightarrow p)}$$

Medidas objetivas: Medida IS

- La medida IS tenderá a ser baja siempre y cuando cualquiera de las reglas $p \rightarrow q$ y $q \rightarrow p$ tenga una confianza baja.
- Cuando las variables p y q son independientes:

$$IS_{ind}(p,q) = \frac{sup(p)sup(q)}{\sqrt{sup(p)sup(q)}} = \sqrt{sup(p)sup(q)}$$

- Esto implica que esta medida tiene los mismos problemas que la confianza.
  - Su valor puede ser alto incluso para variables no correlacionadas o negativamente correlacionada.

Reglas de asociación multinivel

- Existe dominios de aplicación para los que encontrar asociaciones entre los elementos en niveles de abstracción bajo puede ser complicado.
  - Sin embargo, podemos encontrar conocimiento de interés en niveles de abstracción superiores.
- Este tipo de reglas de asociación representadas en diferentes niveles de abstracción se denominan **reglas de asociación multinivel**.
- Para extraer este tipo de reglas se requiere una jerarquía conceptual que agrupe los diferentes conceptos en distintos niveles de abstracción.

Reglas de asociación multinivel

- Para encontrar este tipo de reglas se procede recorriendo la jerarquía en sentido descendente:
  - Se van localizando los itemsets frecuentes nivel por nivel, hasta que no se puedan encontrar más.
  - En cada nivel se puede utilizar alguno de los algoritmos que hemos visto para encontrar itemsets frecuentes.
- Existen diferentes aproximaciones al problema:
  - **Soporte uniforme**. En cada nivel de abstracción se utiliza el mismo umbral para el soporte.
    - Permite aplicar la regla de la antimonotonicidad en el recorrido descendente de la jerarquía.
    - Sin embargo, es poco probable que los items en niveles bajos de abstracción ocurran tan frecuentemente como los de niveles más altos.
    - Si el umbral se coloca muy alto, se pueden estar perdiendo relaciones interesante en niveles bajos de abstracción
    - Si el umbral se coloca muy bajo, se pueden estar generando

Reglas de asociación multinivel

- **Soporte reducido**. En cada nivel se aplica un umbral de soporte diferente. A medida que vamos descendiendo en la jerarquía, el umbral se va haciendo más pequeño.
- **Soporte basado en los grupos**. Cuando se dispone de información sobre qué grupos de items son más importantes, se puede ajustar el umbral del soporte por items o por grupos.

Reglas de asociación multidimensionales

- Hasta ahora sólo hemos tenido en cuenta la presencia o no de los items en cada transacción.
    - Esto sólo nos permite obtener reglas que implican un único predicado, por ejemplo *compra*:

$$compra(X, "ordenador") \Rightarrow compra(X, "antivirus")$$

    - Este tipo de reglas se denominan **unidimensionales** o **intradimensionales**

Reglas de asociación multidimensionales

- En la actualidad, mas que bases de datos transaccionales se suele disponer de bases de datos relacionales o datawarehouses.
  - Este tipo de almacenamiento nos permite almacenar información adicional como la cantidad de artículos comprados, el precio,...
  - E información demográfica sobre el comprador: edad, ocupación, dirección, ingresos, ..
  - Esto nos permite extraer reglas del tipo:

$$edad(X, "20 \ldots 29") \wedge ocupación(X, "estudiante") \Rightarrow compra(X, "portátil")$$

  - Este tipo de reglas, que no repiten los predicados, se denominan **multidimensionales** o **interdimensioales**

Reglas de asociación multidimensionales

- También se puede obtener reglas multidimensionales que repitan algún predicado:

$$edad(X, "20 \ldots 29") \wedge compra(X, "portátil") \Rightarrow$$
$$compra(X, "memoríaUSB")$$

- A este tipo de reglas se les denomina reglas de **dimensionalidad híbrida**
- En este tipo de reglas podemos tener atributos categóricos y cuantitativos. Las técnicas disponibles se pueden agrupar dependiendo como tratan los atributos cuantitativos:
  - **Reglas con discretización estática**. En este caso la discretización se realiza al principio del proceso y no se modifica.
  - **Reglas con discretización dinámica**. La discretización de las variables puede cambiar a medida que se avanza en el proceso de descubrimiento.

Reglas de asociación basadas en restricciones

- En muchos casos dispone de información adicional que nos permite dirigir el proceso de minería.
    - Esto nos permite reducir el espacio de búsqueda.
    - Muchas de estas restricciones que podemos incluir en el proceso ya las hemos analizado:
        - Restricciones de conocimiento.
        - Restricciones de datos.
        - Restricciones de dimensiones o nivel.
        - Restricciones de medida.
- En el caso de reglas de asociación podemos también especificar restricciones de regla.

Reglas de asociación basadas en restricciones

- La utilización de restricciones de reglas permite que el proceso de descubrimiento sea:
  - Más **eficiente**: se reduce el conjunto de datos sobre los que realizar la búsqueda.
  - Más **efectivo**: centra el proceso sólo en los tipos de reglas que son interesantes.
- Las restricciones de reglas se pueden especificar mediante **metarreglas** o **restricciones**.

Reglas de asociación basadas en restricciones

- Las **metarreglas** nos permiten definir en qué tipos de reglas estamos interesados a través de plantillas.
  - Las metarreglas pueden ser interpretadas como hipótesis sobre determinadas relaciones que se está intentando comprobar.
  - Por ejmplo, la metarregla:

    $$P_1(X, Y) \wedge P_2(X, W) \Longrightarrow compra(X, "suite\ ofimática")$$

    nos puede generar asociaciones como:

    $$edad(X, "30 \ldots 39") \wedge ingresos(X, "40 \ldots 60K") \Longrightarrow compra(X, "suite\ ofimática")$$

Reglas de asociación basadas en restricciones

- La definición de una metarregla puede acelerar el proceso de descubrimiento:
  - Sólo hace falta buscar los itemsets frecuentes del tamaño especificado en la regla.
  - De ese conjunto eliminar los que no cumplan alguna restricción.
  - Calcular el soporte y la confianza.

Reglas de asociación basadas en restricciones

- La utilización de **restricciones** nos permiten definir relaciones que tienen que cumplir las variables incluidas en una regla de asociación.
  - Pueden ser aplicadas al final del proceso para filtrar los resultados.
  - Pero resulta más eficiente incluirlas durante el proceso de descubrimiento.
- Las restricciones se puede clasificar en:
  - **Antimonotónicas**: son restricciones que si no se cumplen en un itemset ninguno de sus superconjunto la cumplirá. Por ejemplo, $sum(I.precio) \leq 100$.
  - **Monotónicas**: si un itemset satisface la restricción todos sus supersets la van a satisfacer: Por ejemplo, $tamaño(I) \geq 5$.

Reglas de asociación basadas en restricciones

- **Concisas**: Son aquellas que nos permiten enumerar todos y cada unos de los itemsets que satisfacen la restricción.
  - Permiten podar el conjunto de itemsets antes del proceso de cálculo del soporte.
  - Por ejemplo, $min(I.precio) \geq 100$.
- **Transformables**: son aquellas que no pertenecen a ninguna de las categorías anteriores, pero con una reordenación de los items en un itemset se pueden convertir en antimonotónicas o monotónicas.
  - Por ejemplo $media(I.precio) \leq 100$ se puede convertir en antimonotónica si los items están en orden ascendente por precios.

- El descubrimiento de patrones frecuentes y asociaciones es muy útil el campos como el marketing selectivo, gestión empresarial y toma de decisiones.
- El proceso de búsqueda de reglas de asociación consiste en dos pasos:
  - Detectar los itemsets frecuentes superen el umbral de soporte.
  - Determinar las reglas que superen el umbral de confianza.
- Para la determinación de los itemset frecuentes hemos visto los algoritmos Apriori y FP-growth.
  - El algoritmo **Apriori** se basa en la propiedad de la antomonotonicidad del soporte para podar el espacio de búsqueda y requiere de la generación de itemsets candidatos.
  - El algoritmo **FP-growth** no necesita generar itemsets candidatos utilizando el árbol-FP.

- Para evaluar la importancia de las reglas de asociación hemos visto medidas como: soporte, confianza, lift, factor de interés, correlación,..
- Podemos extender los algoritmos para tratar con reglas de asociación multinivel, multidimensionales, con atributos no binarios y con restricciones.
- De todas formas, las reglas de asociación no se deben utilizar directamente para predicción sin un análisis posterior o con conocimiento del dominio.
    - No indican ningún tipo de causalidad.
    - Sin embargo, constituyen un interesante punto de partida para realizar un análisis muy profundo.

R. Agrawal and R. Srikant.
Fast algorithms for mining association rules.
In *Proc 20th Int Conf Very Large Data Bases VLDB*, volume 1215, pages 487–499. Citeseer, 1994.

R. Agrawal, T. Imielinski, and A. N. Swami.
Mining Association Rules between Sets of Items in Large Databases.
In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, volume 22, pages 207–216, New York, NY, USA, May 1993. ACM.

Jiawei Han, Jian Pei, and Yiwen Yin.
Mining frequent patterns without candidate generation.
SIGMOD Rec., 29(2):1–12, May 2000.

Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao.
Mining frequent patterns without candidate generation: A frequent-pattern tree approach.
Data Mining and Knowledge Discovery, 8(1):53–87, 2004.

Pang-Ning Tan, Michael Steinbach, Vipin Kumar, et al.
Introduction to data mining, volume 1.
Pearson Addison Wesley Boston, 2006.