Minería de Datos

Tema 6 - Reglas de asociación

Nombre: Alejandro pérez Belando

---

1. Introducción

Las reglas de asociación identifican patrones de comportamiento entre los datos en función de la aparición conjunta de valores de dos o más atributos. Estas reglas de asociación nos permiten establecer relaciones entre variables cualitativas.

2. Conceptos previos

Sea $I = \{i_1, \ldots, i_d\}$ el **conjunto de todos los items** que pueden aparecer en una transacción, un **itemset** es un conjunto de items. Un **k-itemset** es un itemset con k elementos.

Sea $T = \{t_1, \ldots, t_N\}$ el **conjunto de todas las transacciones**, una **transacción** está definida por el itemset que indica los items involucrados en la misma.

Una **base de datos transaccional** hace referencia a una colección de transacciones. Las bases de datos transaccionales se pueden representar de tres formas:

- **Representación horizontal (figura izquierda)**: muestra el ID de la transacción (TID) y la lista de items dentro de esa transacción.
- **Representación vertical (figura central)**: muestra (en orden) los TID en los que están cada item. Por ejemplo, el pan está en los itemsets 1, 2, 4, 5
- **Representación binaria (figura derecha)**: muestra los TID y de manera binaria si un itemestá en la transacción (1 si lo está, 0 si no).

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}

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				

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 itemset: sea $X \equiv$ un itemset ($X \in T$) y $\sigma(X) \equiv$ número de transacciones en las que aparece X ($\sigma(X) = \left|\{ t \in T \mid X \subseteq t\}\right|$), el **soporte de** X es la fracción de transacciones que lo incluyen:

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

Un itemset es frecuente si $\text{supp}(X) \geq \sigma_{min}$ (un valor mínimo establecido)

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: es una implicación $X \to Y$, donde $X, Y \in T$ sin itemsets disjuntos[1].

Por ejemplo: la regla $\{Detergente\} \to \{Cerveza\}$ indica que hay una fuerte relación entre la venta de detergente y cerveza.

Soporte de una regla de asociación: fracción de transacciones en las que están incluidos los items tanto del antecedente (X) como del consecuente (Y):

$$\text{supp}(X \to Y) = \text{supp}(X \cup Y)$$

Mide la frecuencia con la que aparecen juntos el antecedente (X) y el consecuente (Y) en todas las transacciones.

Confianza de una regla de asociación: fracción de transacciones en las que aparece el itemset X y que también incluyen al itemset Y:

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

Cuantifica la frecuencia en la que, habiendo ocurrido el antecedente (X), también ocurra el consecuente (Y).

[1] Que dos itemsets sean disjunts significa que no comparten ningún elemento en común. Es decir, su intersección es el conjunto vacío: $X \cap Y = \emptyset$

Por ejemplo: Dada la siguiente representación de bases de datos transaccionales con la regla de asciación $\{Leche,\ Detergente\} \rightarrow \{Cerveza\}$:

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}

El soporte del itemset X es:

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

El soporte de la regla de asociación es:

$$\text{supp}(X \rightarrow Y) = \text{supp}(Leche,\ Detergente,\ Cerveza) = 3/5$$

La confianza de la regla de asociación es:

$$\text{conf}(X \rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)} = \frac{3/5}{4/5} = \frac{3}{4}$$

3. Descubrimiento de las reglas de asociación

Definición del problema: hay que encontrar todas las reglas de asociación que tengan:

- Un soporte $\text{supp}(X \rightarrow Y) \geq \sigma_{min}$
- Una confianza $\text{conf}(X \rightarrow Y) \geq \tau_{min}$

Entonces, debemos definir un soporte y una confianza mínimos.

3.1. Generación de itemsets frecuentes

A partir de un conjunto de datos con k items, se podrían generar $2^k - 1$ itemsets frecuentes. Si K es grande, esto no es viable.

Para reducir la carga computacional se puede:

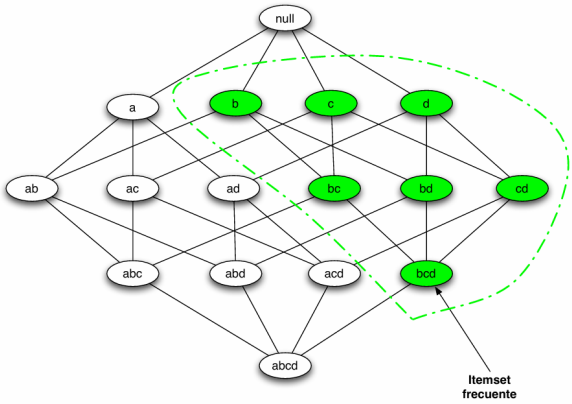
- Reducir el número de itemsets candidatos (algoritmo *Apriori*).
- Reducir el número de comparaciones necesarias para determinar los itemsets frecuentes.

Principio Apriori

Sea un conjunto de items (I) y $T \in 2^I \setminus \{\emptyset\}$ (conjunto no vacío de I) un itemset frecuente ($\text{supp}(T) \geq \sigma_{min}$), entonces:

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

Todos los subconjuntos no vacíos de un itemset frecuente también serán itemsets frecuentes (si un itemset T es frecuente, cualquier adición de items al mismo también será frecuente, aparecerá en todas las transacciones en las que lo haga T).

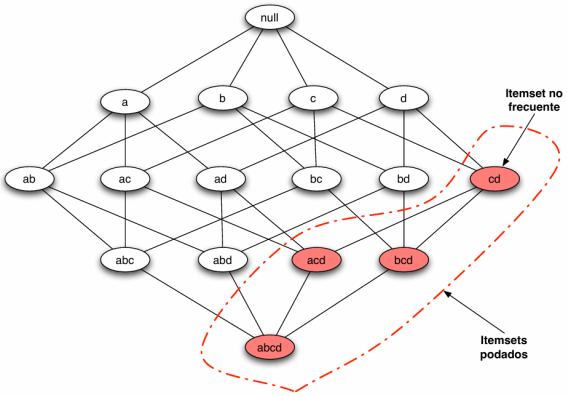


Principio Antimonotonicidad

Sea un conjunto de items (I), una medida f es antimonótina si:

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

Si un itemset I no es frecuente, cualquier adición de items al mismo no será frecuente.Como el **soporte** es una medida **antimonótona**, esta propiedad se puede usar para **podar** el espacio de búsqueda usandolo.



3.2. Algoritmo Apriori

Algoritmo Algoritmo Apriori: Generación de itemsets frecuente

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

Explicación del algoritmo. Sea $C_k \equiv$ conjunto de k itemsets candidatos y $F_k \equiv$ conjunto de k itemsets frecuentes, el algoritmo comienza de la siguiente forma:

1. Barrido de los datos para determinar los F_1 (1-itemsets frecuetes, los itemsets frecuentes formados por 1 item) [2].
2. Se van generando iterativamente los C_k (k-itemsets candidatos) a partir de los F_{k-1} ((k-1)-itemsets frecuentes) [5].
3. Se recorren de nuevo los datos para calcuar el supp(C_k) (el soporte de cada k-itemset candidato) [8-10].
4. Del conjunto C_k se seleccionan los F_k (k-itemsets que son frecuentes) [12].
5. El proceso acaba cuando no se generan más F_k y se devuelven todos los itemsets frecuentes de tamalos 1 hasta k [13]

3.3. Generación de itemsets candidatos

Este proceso de realiza en dos fases:

- Generación de los C_k a partir de los F_{k-1} obtenidos antes.
- Poda de candidatos: elimiar los C_k cuyo soporte sea $< \sigma_{min}$

3.3.1. Estrategia de fuerza bruta para generar candidatos

El número de posibles candidatos es $\binom{d}{k}$[2].

El proceso de poda eliminará los candidatos no frecuentes. En este caso el proceso se poda es muy costoso, la complejidad total sería $O\left(\sum_{k=1}^{d} \times \binom{d}{k}\right) = O(d2^{d-1})$

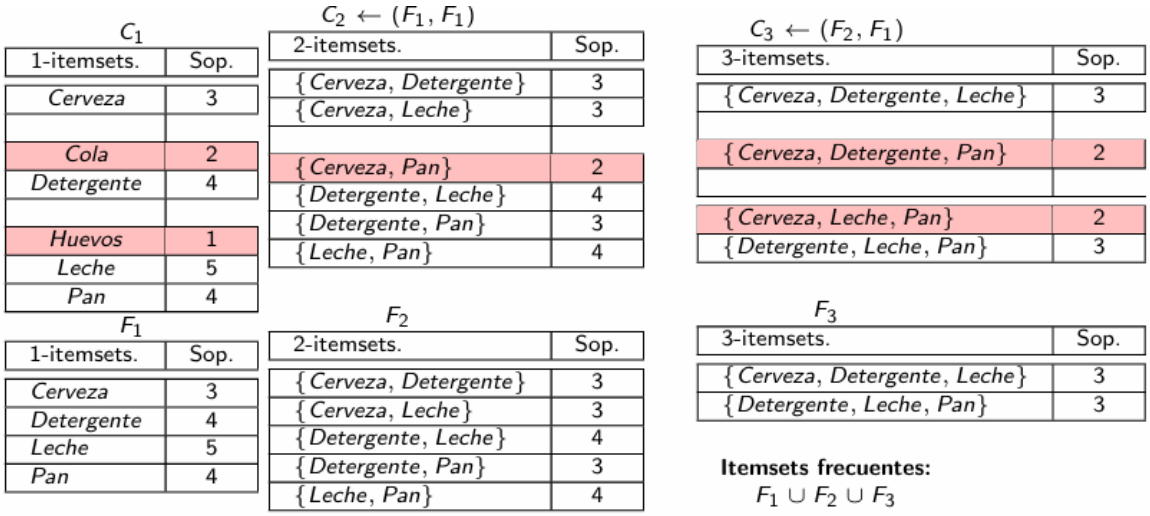
3.3.2. Estrategia de $F_{k-1} \times F_k$ para generar candidatos

La idea es extender cada (k-1)-itemset frecuente con un 1-itemset frecuente.

El coste total sería $O\left(\sum_k k|F_{k-1}||F_1|\right)$

El método es completo pero se pueden generar itemsets duplicados y/o innecesarios.

Ejemplo para entender este concepto: en la siguiente imagen se muestran los k-itemsets candidatos (C_k) y frecuentes (F_k). Establecemos el soporte mínimo en 3 (es decir, cualquier itemset cuyo soporte sea menor que 3 se poda):



Fijándonos en la figura podemos ver lo siguiente:

1. Temenos los 1-itemsets candidatos (C_1), donde hemos dicho que los 1-itemsets frecuentes (F_1) son los que tienen soporte mayor o igual a 3 (supp ≥ 3), por lo que los itemsets marcados en rojo son aquellos que no cumplen esta condición y son podados.
2. Para construir los 2-itemsets candidatos se usan distintas combinaciones de los 1-itemsets frecuentes (F_1) consigo mismos, es decir, (F_1, F_1). De esos 2-itemsets candidatos se podan los 2-itemsets que no cumplen supp ≥ 3 para generar los 2-itemsets frecuentes (F_2).
3. El proceso se repite pero en este caso se generan los 3-itemsets candidatos usando combinaciones de los 2-itemsets frecuentes (F_2) y los 1-itemsets frecuentes (F_1), es decir, (F_2, F_1). Finalmente, en este ejemplo, los dos 3-itemsets frecuentes (F_3) que quedan son {Cerveza, Detergente, Leche} y {Detergente, Leche, Pan}.

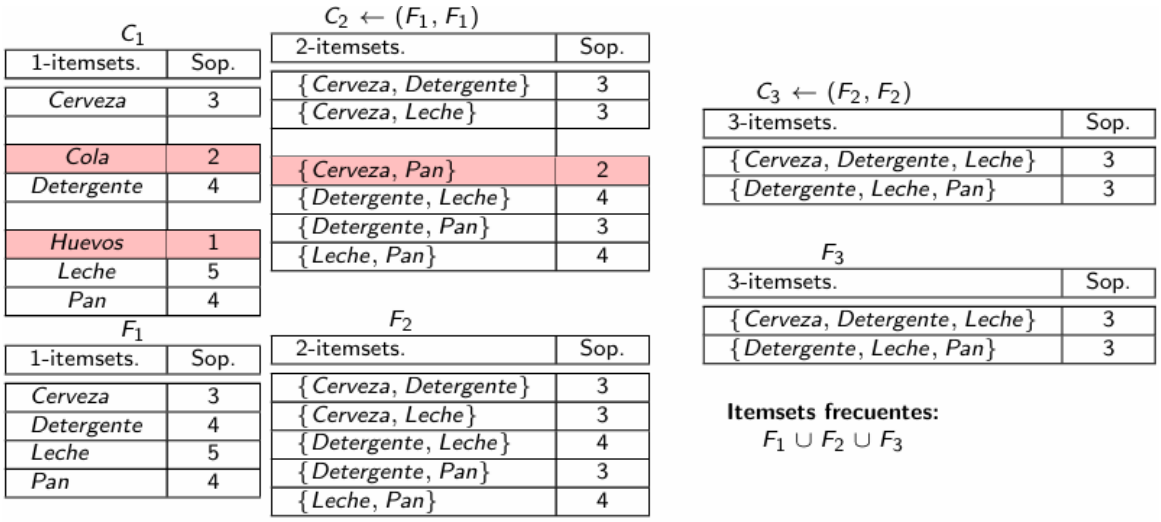
[2]Recordemos que esto se calcula de la forma: $\binom{d}{k} = \frac{d!}{k!(d-k)!}$

3.3.3. Estrategia de $F_{k-1} \times F_{k-1}$ para generar candidatos

Se fusionan dos (k-1)-itemset frecuentes cuyos primeros (k-2)-items son idénticos (suuponiendo un orden alfabético en los items)

El método es completo y no se pueden generar itemsets duplicados al estar en orden alfabético.

Ejemplo para entender este concepto: en la siguiente imagen se muestran los k-itemsets candidatos (C_k) y frecuentes (F_k). Establecemos de nuevo el soporte mínimo en 3:



El proceso es similar al anterior, pero con un nuevo matiz.

1. Este paso es igual que en el proceso anterior.
2. Este paso es también igual que en el proceso anterior.
3. Aquí es donde cambia la cosa, ya que como hemos dicho, en este caso lo 3-itemsets candidatos (C_3) se construyen fusionando los F_2 coyos primeros items sean idénticos. de esta forma, hay solo 2 C_3 y ninguno se poda ya que cumplen la restricción del soporte: {Cerveza, Detergente, Leche} y {Detergente, Leche, Pan}.

3.4. Algoritmo Apriori: cálculo del soporte

Una manera burda de proceder puede ser comparar cada transacción con cada itemset candidato, aunque esto es computacionalmente muy costoso.

Existen **otras técnicas** basadas 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, detectando solo los drecuentes para determinadas longitudes.
- **Fase Backward**: Calcular el soporte para el resto de itemsets no considerados (Eliminando previamente los itemsets que son subconjuntos de algún itemset frecuente). Este proceso encuentra los itemsets maximales.

Itemsets maximales: un itemset frecuente es maximal si ninguno de los superconjuntos[3] propios es frecuente.

Los itemsets maximales constituyen el conjunto más pequeño de itemsets frecuentes a partir del que se pueden generar el resto de itemsets frecuentes.

Itemsets cerrados frecuentes: un itemset es **cerrado** si ninguno de sus superconjuntos tiene exactamente el mismo soporte.

Un itemset X es **cerrado frecuente** si X es un itemset cerrado y frecuente. Es una forma compacta de representar los itemsets frecuentes y su soporte.

Relaciones entre itemsets maximales y cerrados frecuentes:

- **Todo itemset maximal es cerrado**, pero no todo itemset cerrado es maximal.
- Los conjuntos cerrados representan todos los conjuntos de items frecuentes, pero los maximales pueden perder parte de esta información.

3.5. Algoritmo Frequent-Pattern Growth (FP-Growth)

nace para evitar dos problemas del algoritmo Apriori: la generación de muchos itemsets candidatos y la necesidad de realizar varias pasadas a la base de datos.

Estrategia básica: divide y vencerás

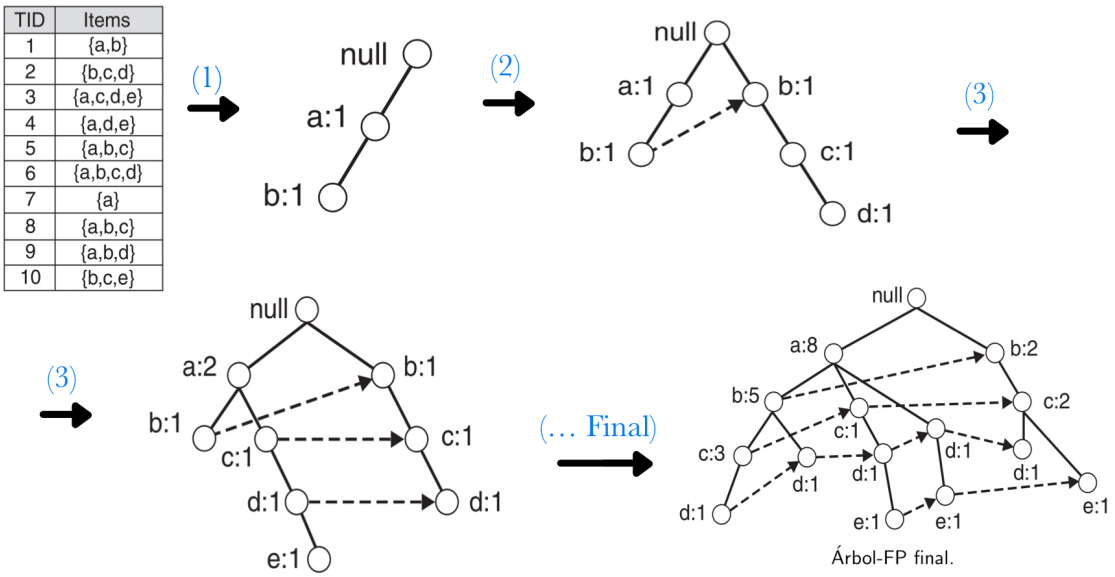
- Compactar la base de datos usando un Frequent Pattern tree (FP-tree). Algunas nociones de estos árboles:
  - Cada nodo del árbol representa un item y tiene un contador con el número de transacciones que comparten el camino (la secuencia de ítems) desde la raíz hasta ese nodo.
  - El nodo raíz es el nodo *null*.
  - Hay enlaces entre los nodos que representan items idénticos (flechas discontinuas).
- Dividir la base compactada en bases de datos condicionales y extraen los itemsets frecuentes de ellas.

[3]Un superconjunto propio de un conjunto X es cualquier conjunto que contiene todos los elementos de X y al menos uno más).

3.5.1. Algoritmo FP_{Growth}: construcción de un FP_{tree}

Este proceso se va a explicar ejemplificando el proceso con la tabla y la figura adjunta. Antes que nada, hay que calcular el soporte de cada itemset y eliminar los no frecuentes, ordenar los frecuentes en orden descendente según su soporte en cada intersección (el orden descendente del soporte hace que se obtengan árboles más compactos). El proceso de construcción es el siguiente:

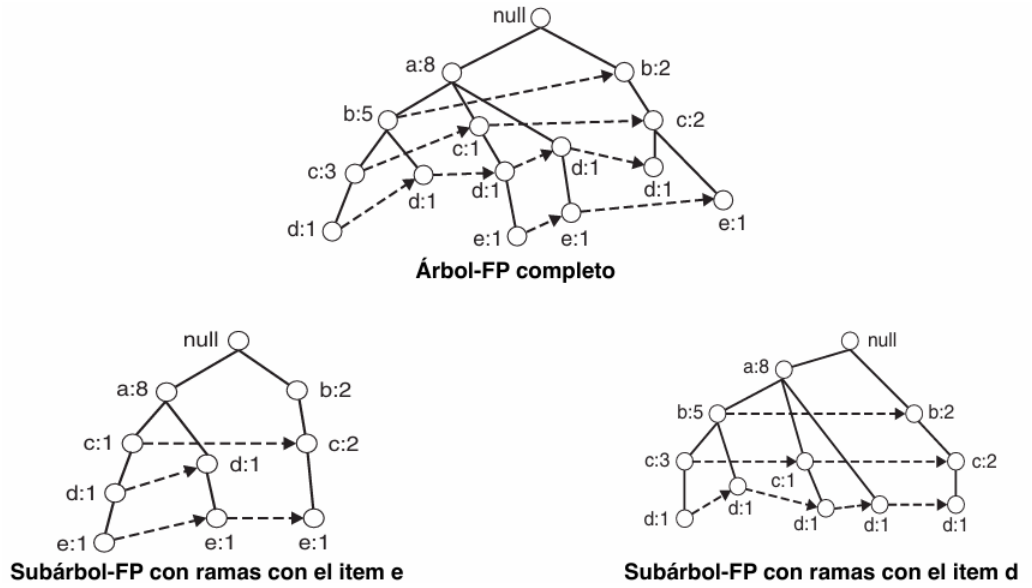
1. La primera transacción, {a, b} da lugar al camino $null \rightarrow a \rightarrow b$. A cada una de las etiquetas se les inicia el contador en 1.
2. La segunda transacción, {b, c, d} da lugar al camino $null \rightarrow b \rightarrow c \rightarrow d$. los contadores nuevos se inicializan también en 1. Aunque comparten el item b, como no es el primer iem, se crea una rama nueva y se crea un enlace (flecha discontinua) entre los nodos que representan el item b.
3. La tercera transacción, {a, c, d, e} da lugar al camino $null \rightarrow a \rightarrow c \rightarrow d \rightarrow e$. Como en este caso se comparte el primer item con el primer item de una rama ya existente, comparten nodo con ese item ya etiquetado con a (el nodo a incrementa su contador en 1). El resto de items de la transacción se crean nuevos porque no comparten más ítems ordenados con la rama ya creada.
4. Ese proceso se repite hasta el final.



3.5.2. Algoritmo FP_{Growth}: generación de itemsets frecuentes

Esto se realiza recorriendo el FP_{tree} en orden ascendente.

Primero que nada, vamos a dividir el árbol completo obtenido anteriormente en dos subárboles (con las ramas que tienen el item e y con las ramas que tienen el item d):

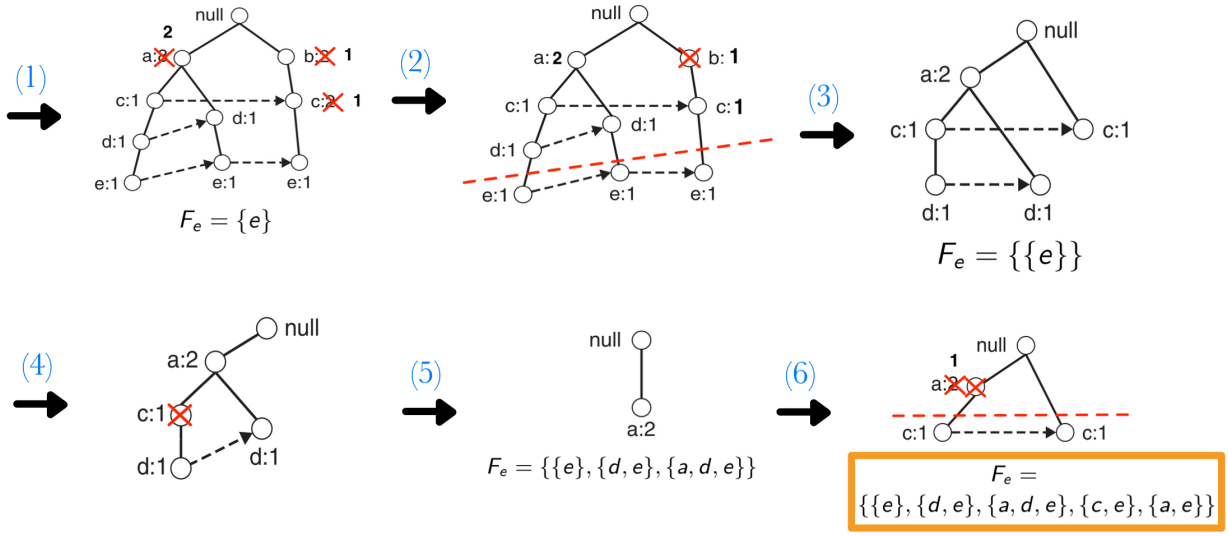


Vamos a calcular únicamente, a modo de ejemplo, todos los itemsets frecuentes que acaban e (en las ramas e)

1. Actualizar contadores del nuevo árbol (lo consideramos como independiente) Sumando los dos contadores, vemos que el soporte de los nodos e es 3, por lo que {e} es frecuente.
2. Eliminar los nodos que contienen e (ya no son necesarios) y los nodos no frecuentes (en este caso b, porque su contador es 1 y no tiene enlaces).
3. Se usa ahora este árbol conidcional para e para encontrar los itemsets frecuentes acabados en de, ce y ae.
4. Para encontrar los itemsets frecuentes acabados en de, generamos un subárbol acabado en d a partir del árbol que teníamos en el paso 3. Eliminamos c ya que su soporte es 1. Como d tiene un enlace, sumamos los dos contadores y vemos que el doporte de los nodos d es 2, por lo que {d, e} también es frecuente.
5. Se construye un subárbol acabado en a a partir del árbol que teníamos en el paso 4 (Ya no usamos los nodos d). Como este árbol condicional solo tiene el nodo a con soporte 2, el itemset {a, d, e} es frecuente también.
6. Para encontrar los itemsets frecuentes acabados en ce, generamos un subárbol acabado en c a partir del árbol que teníamos en el paso 3. Reiniciamos el contador de a. Como c

tiene un enlace, sumamos los dos contadores y vemos que el doporte de los nodos c es 2, por lo que {c, e} también es frecuente.

7. Se construye un subárbol a partir del árbol que teníamos en el paso 5. Como ya no usamos los nodos c y eliminamos el nodo a que no es frecuente, el árbol restante es vacío el itemset frecuente es {a, e}.



Realizando el proceso análogo a todos los itemsets acabados en d, c, b, y a, los itemsets frecuentes serían:

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	$\{a\}$

Ventajas del algoritmo FP-Growth:

- Es mas rápido que el Apriori: solo da una pasada a la base de datos. No se generan itemsets frecuentes duplicados.
- Comprime los datos.

Desventajas del algoritmo FP-Growth:

- La construcción del árbol es muy costosa Una vez construido encontrar los itemsets frecuentes es muy facil.
- Solo se poda itmsets individuales y no-itemsets.

4. Generación de reglas de asociación

A partir de cada itemset frecuente, se pueden generar $2^k - 2$ reglas de asociación.

Para crear una regla e asociación hay que:

1. Dividir el itemset frecuente (X) en dos conjuntos disjuntos: Y; $X \setminus Y$
2. Generar la regla $Y \rightarrow X \setminus Y$, si se supera la confianza mínima τ_{min}
3. No se tienen en cuenta las reglas $Y \rightarrow \emptyset$[4] y $\emptyset \rightarrow Y$[5]

Ejemplo: sea el itemset frecuente {a, b, c}, a partir del mimso se pueden generar dos reglas de asociación:

- $\{a, b\} \rightarrow \{c\}, \{a, c\} \rightarrow \{b\}, \{b, c\} \rightarrow \{a\}$
- $\{a\} \rightarrow \{b, c\}, \{b\} \rightarrow \{a, c\} \rightarrow \{b\}, \{c\} \rightarrow \{a, b\}$

Notas sobre la generación de reglas de asociación:

- El soporte de cada regla supera el mínimo al generarse de un itemset frecuente.
- El cálculo de la confianza de las reglas no requiere volvr a recorrer la base de datos:

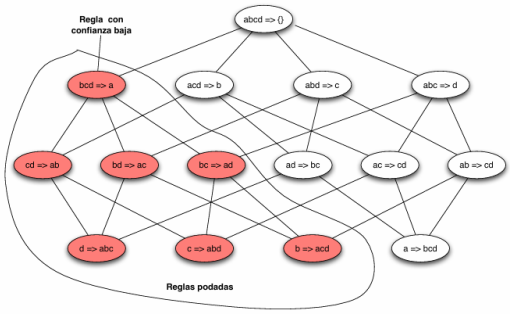
$$conf(X \rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)}$$

4.1. Generación de reglas de asociación: poda basada en confianza

Si una regla de asociación ($X \rightarrow Y$) no supera el umbral de confianza, entonces la regla de confianza creada por el superconjunto Y' ($Y' \subseteq Y$) tampoco va a superar ese umbral. De esta forma se puede podar un conjunto de reglas de asociación.

De forma matemática:

$$\text{conf}\big(Y \rightarrow X \setminus Y\big) \leq \tau_{\text{mín}} \implies \forall Y' \in 2^X \setminus \{\emptyset\},\ Y' \subseteq Y :\ \text{conf}\big(Y' \rightarrow X \setminus Y'\big) \leq \tau_{\text{mín}}.$$



[4] $Y \rightarrow \emptyset$, quiere decir algo como 'sin ninguna condición, sucede Y'

[5] $\emptyset \rightarrow Y$, por el otro lado, significa 'si ocurre Y, entonces no ocurre nada'

4.2. Generación de reglas de asociación: algoritmo Apriori

En este caso sigue un proceso por niveles, donde cada nivel se corresponde con el número de items que tiene el consecuente. Comenzamos con las reglas que tienen solo un item en el consecuente y nos quedamos con las que tienen una confianza más alta.

Estas reglas se usan para generar las reglas del siguiente nivel:

$$\left.\begin{array}{l} \{a,c,d\} \rightarrow \{b\} \\ \{a,b,d\} \rightarrow \{c\} \end{array}\right\} \Longrightarrow \{a,d\} \rightarrow \{b,c\}.$$

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)

1: $k = |f_k|$; ▷ Tamaño del itemset frecuente
2: $m = |H_m|$ ▷ Tamaño del consecuente
3: **si** $k > m + 1$ **entonces**
4: H_{m+1} =apriori-gen(H_m)
5: **para cada** $h_{m+1} \in H_{m+1}$ **hacer**
6: $conf = \sigma(f_k)/\sigma(f_k \setminus h_{m+1})$
7: **si** $conf \geq \tau_{min}$ **entonces**
8: **generar la regla** $(f_k \setminus h_{m+1}) \rightarrow h_{m+1}$
9: **sino**
10: $H_{m+1} = H_{m+1} \setminus h_{m+1}$
11: **fin si**
12: **fin para**
13: ap-genrules(f_k, H_{m+1})
14: **fin si**

5. Evaluación de reglas de asociación

Los métodos analizados pueden generar miles o millones de reglas.

5.1. Medidas objetivas

Muchas de las medidas se calculan a partir 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

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

5.1.1. Medidas objetivas: soporte y confianza

El soporte puede presentar problemas en casos de 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.

Un soporte demasiado bajo aumenta los requisitos de computación, da muchas reglas y se pueden tener reglas espúreas[6].

Reglas con una confianza alta se pueden perder (la confianza no iene en cuenta el soporte del consecuente). Esto se puede evitar usando **lift**.

5.1.2. Medidas objetivas: Lift y factor de interés

Expresión general de la medida **lift**:

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

Para variables binarias, el lift equivale al **factor de interés**:

$$I(p,q) = \frac{\text{supp}(p,q)}{\text{supp}(p)\text{supp}(q)} = \frac{N f_{11}}{f_{+1} f_{+1}}$$

[6]Regla espúrea: regla que, a pesar de cumplir los umbrales de soporte y confianza, no refleja una verdadera dependencia entre antecedente (X) y consecuente (Y)

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

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

Puede dase que asociaciones muy frecuentes tengan un factor de interés cercano a 1 por las distribuciones de las ocurrencias (especialmente si el peso de la matriz de contingencia se concentra en f_{11}

5.1.3. Medidas objetivas: análisis de correlación

Basada en técnicas estadísticas para medir la relación entre dos variables. Para variables contínuas, se usa el coeficiente de correlación de Pearson. Para variables binarias:

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

Donde:

- $\phi = -1 \equiv$ correlación negativa perfecta.
- $\phi = 0 \equiv$ variables estadístiicamente independientes.
- $\phi = 1 \equiv$ correlación positiva perfecta.

Esta medida es muy útil cuando se están analizando variables binarias simétricas, pero se ve afectada por cambios proporcionales en el tamaño de las muestras.

5.1.4. Medidas objetivas: medida IS

Propuesta para tratar variables binarias asimétricas

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

Para variables binarias, esta medida equivale a la medida basada en el coseno

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

También es la **media geométrica de la confianza** de la regla de asociación

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

Cuando las **variables** p y q son **independientes**:

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

Esta medida tiene los mismos problemas que la confianza: su valor puede ser alto incluso para variables no correlacionadas o negativamente correlacionadas.

6. Otros tipos de reglas de asociación

6.1. Reglas de asociación multinivel

Son reglas de asociación representadas en distintos niveles de abstracción y para extraerlas se requiere de una jerarquía conceptual (que agrupe los distintos conceptos en diferentes niveles de abstracción).

Se procede recorriendo esta jerarquía en sentido descendente: se localizan los itemsets frecuentes nivel por nivel.

Tres aproximaciones al problema:

- **Soporte uniforme**: en cada nivel de abstracción se usa el mismo umbral para el soporte. Si el umbral se coloca alto, se puede perder relaciones interesantes a nivel bajo de abstracción. Si se coloca muy bajo es malo también (el pdf está cortao xd)
- **Soporte reducido**: en cada nivel, un umbral de soporte distinto.Conforme descendemos en la jerarquía, el umbral se hace más pequeño.
- **Soporte basado en los grupos**: conociendo información sobre qué itemsets son más importantes, se puede ajustar el soporte por items o grupos.

6.2. reglas de asociación multidimensionales

Hasta ahora, solo hemos tenido en cuenta la presencia o no de los items en cada transacción. Esto Solo nos permite tener reglas que implican un único predicado (unidimensionales o intradimensionales).

En la actualidad se dispone de bases de datos relacionales (o datawarehouses) más que transaccionales, y podemos generar reglas multidimensionales (que no repiten los predicados) o de dimensionalidad híbrida (reglas multidimensionales que repiten algún predicado).

En este tipo de reglas podemos tener atributos cuantitativos y categóricos.

Reglas según tratan a los atributos cuantitativos:

- **Reglas de discretización estática**: la discretización se hace al principio del proceso y no se modifica.

- **Reglas de discretización dinámica**: la discretización puede variar a medida que se avanza en el proceso de descubrimiento.

6.3. Reglas de asociación basadas en restricciones

Permite de el proceso de búsqueda sea más eficiente (reduce los datos sobre los que hacer la búsqueda) y efectivo (centra el proceso en los tipos de reglas que son interesantes).

Metarreglas: permiten definir en qué tipos de reglas estamosinteresados a través de plantillas. Pueden ser interpretadas como hipótesis sobre determinadas relaciones que se está intentando comprobar.

Restricciones: permiten definir relaciones que tienen que cumplir las variables incluidas en una regla de asociación. Pueden ser aplicadas al final del prodeso para filtrar los datos pero es más eficiente incluirlas durante el proceso de descubrimiento. Las reglas pueden clasificarse en:

- **Antimonotónicas**: si no se cumplen en un itemset, ninguno de sus superconjuntos las cumplirán.
- **Monotónicas**: si un itemset satisface esta restricción todos sus superconjuntos la cumplen.
- **Concisas**: Permiten enumerar todos los itemsets que satisfacen la restricción. Permiten podar el conjunto de itemsets antes del proceso de cálculo de soporte.
- **Transformables**: no pertenecen a ninguna de las categorías anteriores, pero reordenando los items de un itemset se pueden convertir en antimonotónocas o monotónicas.