Tema 4: Reglas de asociación

Universidad Francisco de Vitoria
UFV Madrid

Alberto Nogales
alberto.nogales@ufv.es
Curso 2021-2022

Índice

- Definición del problema
- Algoritmo Apriori
  - Generación de reglas de asociación
- Generalized Sequential Patterns
  - Extracción de patrones secuenciales

Definición del problema

- **Objetivo:** dado un conjunto de transacciones, encontrar las reglas que predecirán la aparición o no de un elemento (característica) concreto basado en la aparición del resto de los elementos en la transacción.

Id	Items
1	Pan, leche
2	Pan, pañales, cerveza, huevos
3	Leche, pañales, cerveza, cola
4	Pan, leche, pañales, cerveza
5	Pan, leche, pañales, cola

Reglas de asociación:

Pañales → Cerveza

Leche, pan → Huevos, cola

Cerveza, pan → Leche

Aplicaciones

Product placement (donde colocar los productos en el supermercado)

- Objetivo: Identificar los artículos que los clientes compran juntos.

- Solución: Analizar todos los tickets que se producen en el supermercado.

- Ejemplo: Si un cliente compra carbón es muy probable que compre carne (bueno, encontraríamos algo menos obvio).

Aplicaciones

Promociones y ofertas

- Objetivo: Estudiar el impacto de los productos que se venden.

- Solución: Analizar todas las ventas online.

- Ejemplo: Una regla del tipo: impresora → toner. Con el consecuente se puede saber como incrementar sus ventas. Con el antecedente sabemos que productos se pueden ver afectados si se deja de vender este.

Aplicaciones

Gestión de inventarios

- Objetivo: Una empresa de reparación de electrodomésticos quiere anticiparse a las incidencias para mantener los vehículos equipados y no hacer visitas sin solventar los problemas.

- Solución: Procesar los datos de herramientas y piezas usadas en el tipo de reparación.

- Ejemplo: perdida_agua→tubo, goma, ocurre un 40%
  perdida_agua → tubo, ocurre un 60%

Definiciones

- **Conjunto de elementos** (itemset): Una colección de uno o más elementos.
- **K-itemset**: Un conjunto con **k** elementos.

- **Frecuencia de soporte** (σ): Frecuencia de aparición de un conjunto.
- **Soporte**: Frecuencia de un conjunto entre el total de transacciones. La probabilidad condicional de que una transacción que contenga X (antecedente) también contenga Y (consecuente), es decir los elementos de la regla.

- Conjunto de elementos (itemset) **es frecuente**: Un conjunto de elementos cuyo soporte es **>=** que un umbral (**soporte mínimo**) dado o elegido.

Definiciones

Id	Transacciones
1	Pan, leche
2	Pan, pañales, cerveza, huevos
3	Leche, pañales, cerveza, cola
4	Pan, leche, pañales, cerveza
5	Pan, leche, pañales, cola

Conjunto de elementos: {Leche, pan, pañales, cerveza, huevos, cola}

- 3-itemset. {Leche, pan, pañales}
- Frecuencia de soporte: 2.
- Soporte: 2/5 = 0,4.
- Es un conjunto de elementos frecuentes si umbral >= 0,4.

Definiciones

- **Regla de asociación**: una relación X → Y donde X e Y son conjuntos de elementos. Se tiene que cumplir:
  - Todos los elementos de X e Y pertenecen al conjunto inicial de datos.
  - Los elementos de X no se repiten en Y y viceversa.

- Métricas para la evaluación de reglas:
  - **Confianza**: Frecuencia con que los elementos en Y aparecen en transacciones que contienen a X. Dicho de otra manera, la probabilidad de que una transacción contenga X e Y entre todas las que contienen sólo a X. Deben ser >= que un umbral mínimo.

Definiciones

Id	Transacciones
1	Pan, leche
2	Pan, pañales, cerveza, huevos
3	Leche, pañales, cerveza, cola
4	Pan, leche, pañales, cerveza
5	Pan, leche, pañales, cola

Regla de asociación: {Leche, pañales} → Cerveza

Confianza = σ(Leche,pañales,cerveza)/σ(Leche,pañales)

= 2/3 = 0,67

Complejidad

Soluciones → Fuerza bruta:

- Hacer una lista de **todas las posibles asociaciones**.
- **Calcular** el **soporte** y **confianza** de cada una.
- **Eliminar** las reglas que no cumplen los umbrales mínimos.

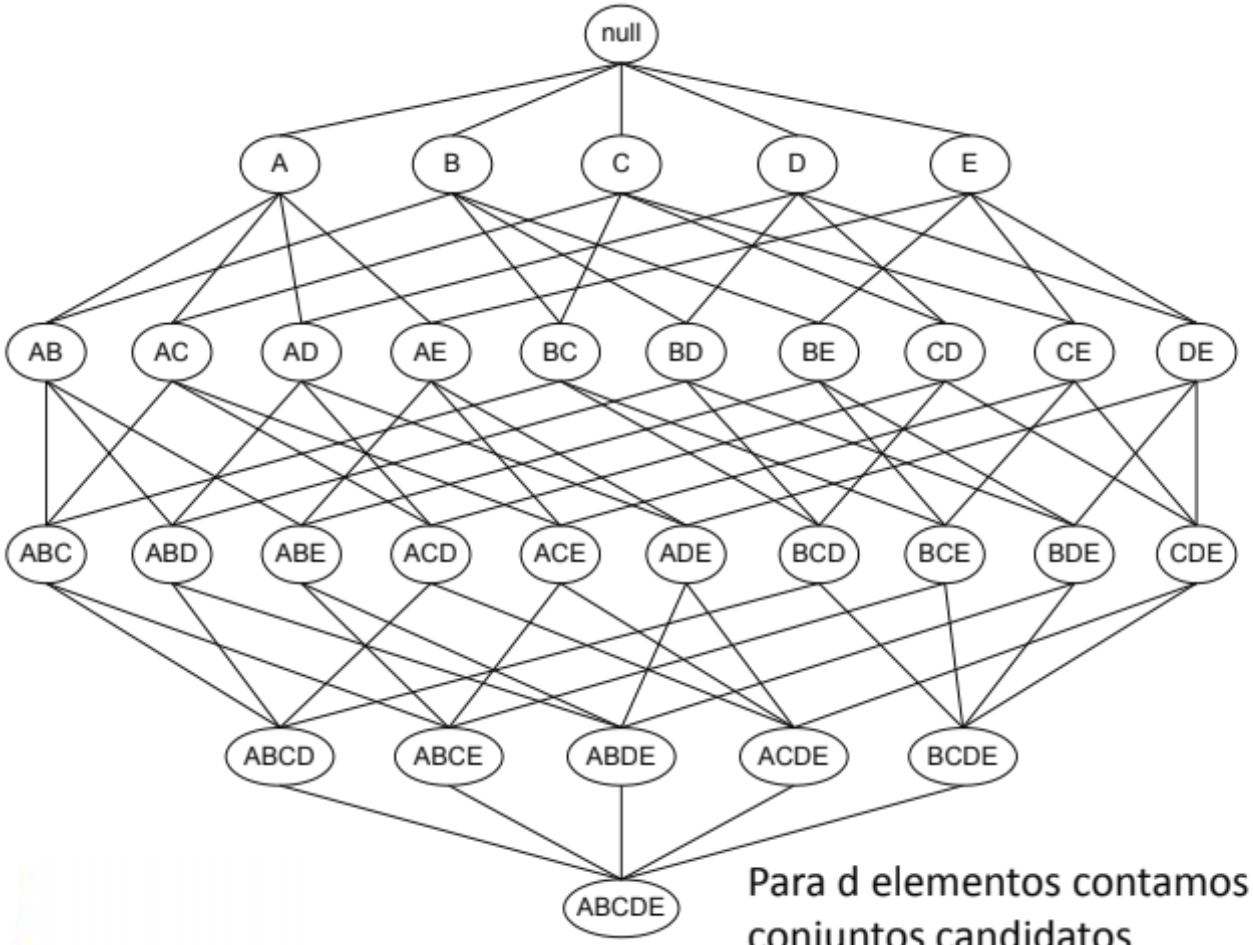
COSTE COMPUTACIONAL MUY ELEVADO!!!!

Complejidad

Generación de los elementos más frecuentes:

- Cada conjunto de elementos del registro de transacciones es un candidato a conjunto más frecuente.
- Seleccionado un candidato se evalúa su soporte examinando todo el dataset.
- Complejidad depende de $M=2^d$ donde d es el número de elementos total.

Complejidad



Para d elementos contamos con 2^d conjuntos candidatos

Complejidad

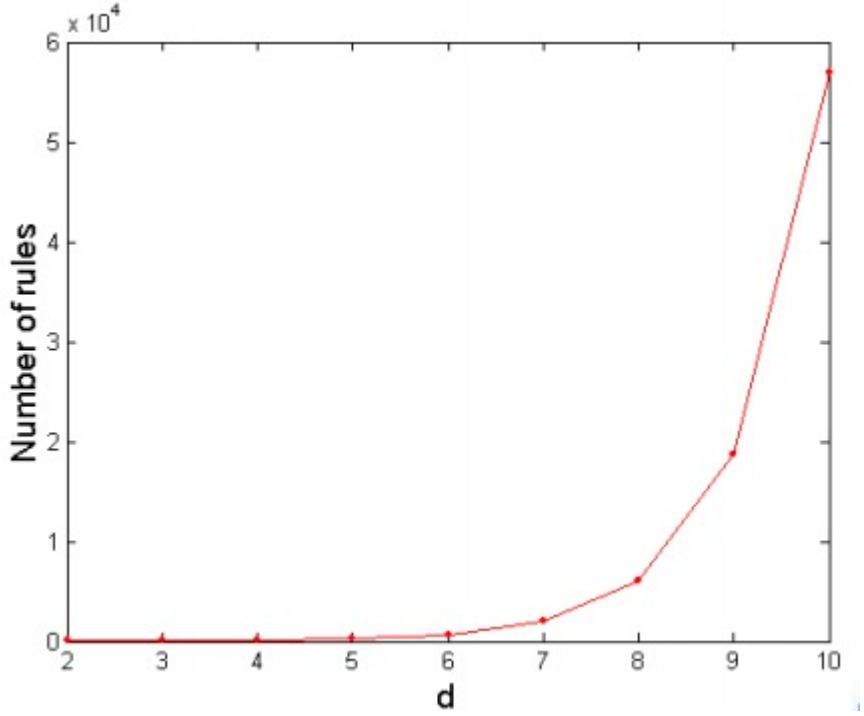
Complejidad computacional:

– El número total de conjuntos es 2^d

– El número total de reglas de asociación es

$$R = \sum_{k=1}^{d-1} \left[\binom{d}{k} \times \sum_{j=1}^{d-k} \binom{d-k}{j} \right]$$

Si d=6, R=602 reglas.



Complejidad

Reducir el número de candidatos (M):

– Se pueden emplear técnicas de poda para reducir M.

Reducir el número de transacciones (N):

– Reducir N conforme aumenta el tamaño del conjunto de transacciones.
– Usando Direct Hashing and Pruning (DHP)

Reducir el número de comparaciones:

– Usando estructuras de datos eficientes para almacenar candidatos y transacciones.
– Evitar chequear cada candidato frente a cada transacción.

Complejidad

Método de los **dos pasos**:

– Generación de **itemsets frecuentes**: itemsets cuyo **soporte** supere o iguale el umbral mínimo.

– Generación de **reglas**: generar reglas de alta confianza (con **confianza** >= umbral mínimo) sobre estos itemset frecuentes.

Algoritmo Apriori

Algoritmo Apriori:

1. Empezando por los ítems individuales, k=1
   - Generar itemsets frecuentes de longitud 1.

2. Repetir hasta que no haya más **itemsets frecuentes**
   - Generar itemsets candidatos de longitud k+1 de los itemsets k-frecuentes.
   - Podar candidatos que contengan subconjuntos de longitud k que no sean frecuentes (propiedad antimonótona).
   - Calcula el soporte de cada candidato.
   - Elimina los candidatos que no sean frecuentes, dejando sólo los frecuentes.

Algoritmo Apriori

3. Extraer y evaluar las **reglas**
   - Generar todas las combinaciones posibles a partir de los itemsets frecuentes.
   - Para cada regla calcular la confianza.
   - Aquellas que no cumplan con el umbral necesario, se descartan.

Itemsets frecuentes

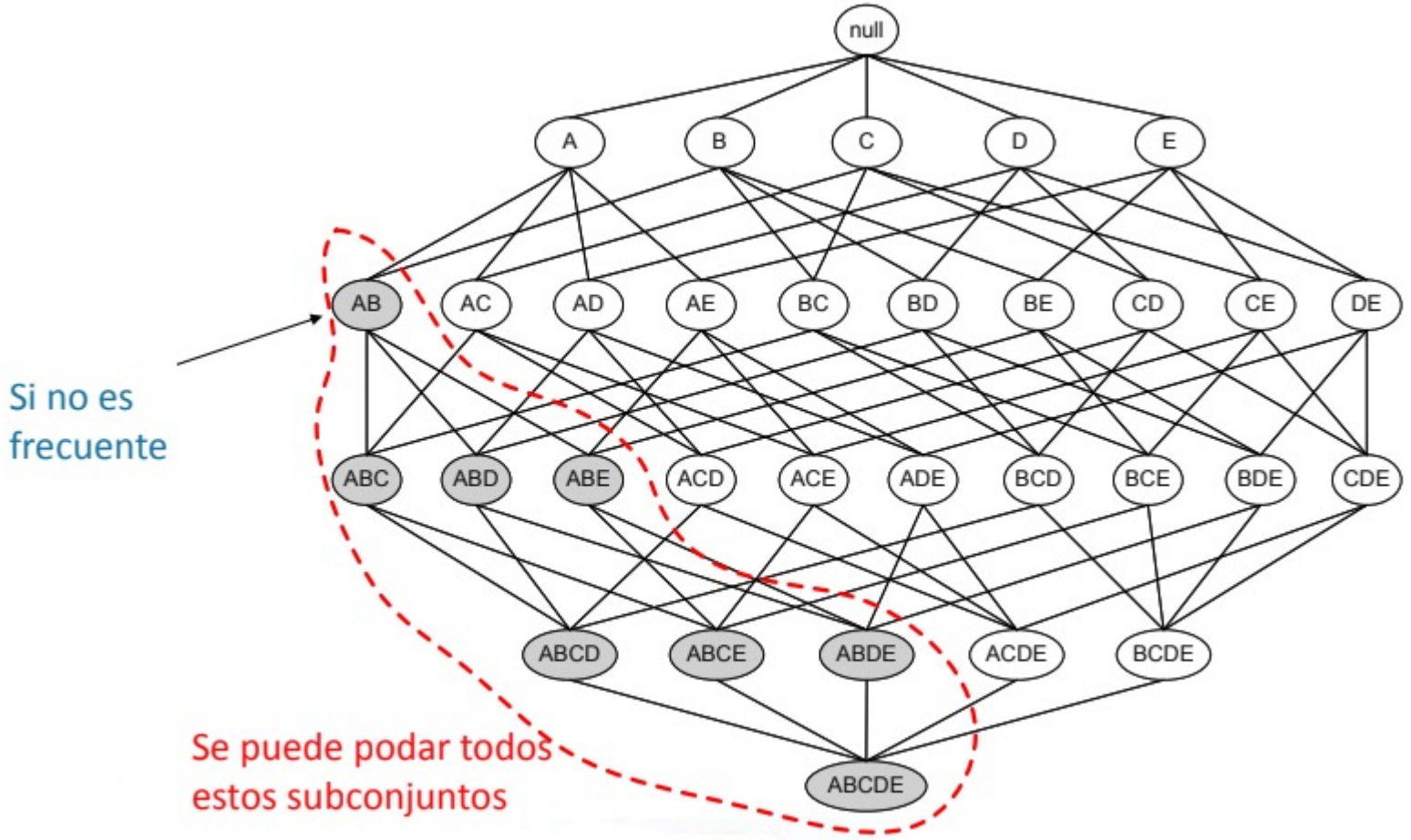
Principio Apriori:

- Si un conjunto de elementos no es frecuente, entonces todos sus subconjuntos tampoco son frecuentes.

- Se aplica la siguiente propiedad: $\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$
  - El **soporte** de un **conjunto** de elementos **nunca excede** el nivel de **soporte** de sus **subconjuntos**.
  - Se conoce como la **propiedad antimonótona** del nivel de soporte.

  Ej: A,B está contenido en A,B,C. El soporte de (A,B) es mayor que el de (A,B,C)

Itemsets frecuentes



Itemsets frecuentes

Principio Apriori paso a paso:

Para frecuencia de soporte >= 3

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

Pairs (2-itemsets)

(No es necesario generar candidatos que impliquen a cola o huevos)

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

Extracción de reglas

Una vez obtenidos los conjuntos frecuentes:

- Para cada conjunto L de ítems se generan todos sus subconjuntos.

- Para cada subconjunto $s \subset L$, se genera una regla: $s \rightarrow (L-s)$ si: $\sigma(L)/\sigma(s) >= \text{confianza}$

Todas las reglas satisfacen las condiciones mínimas de soporte.

Extracción de reglas

A partir del itemset frecuente {A,B,C,D} se generan las siguientes reglas:

$A \rightarrow B,C,D$	$B,C \rightarrow A,D$
$B \rightarrow A,C,D$	$B,D \rightarrow A,C$
$C \rightarrow A,B,D$	$C,D \rightarrow A,B$
$D \rightarrow A,B,C$	$A,B,C \rightarrow D$
$A,B \rightarrow C,D$	$A,B,D \rightarrow C$
$A,C \rightarrow B,D$	$A,C,D \rightarrow B$
$A,D \rightarrow B,C$	$B,C,D \rightarrow A$

Si $|L|=k$ hay 2^k-2 reglas

Extracción de reglas

Generar reglas de forma eficiente:

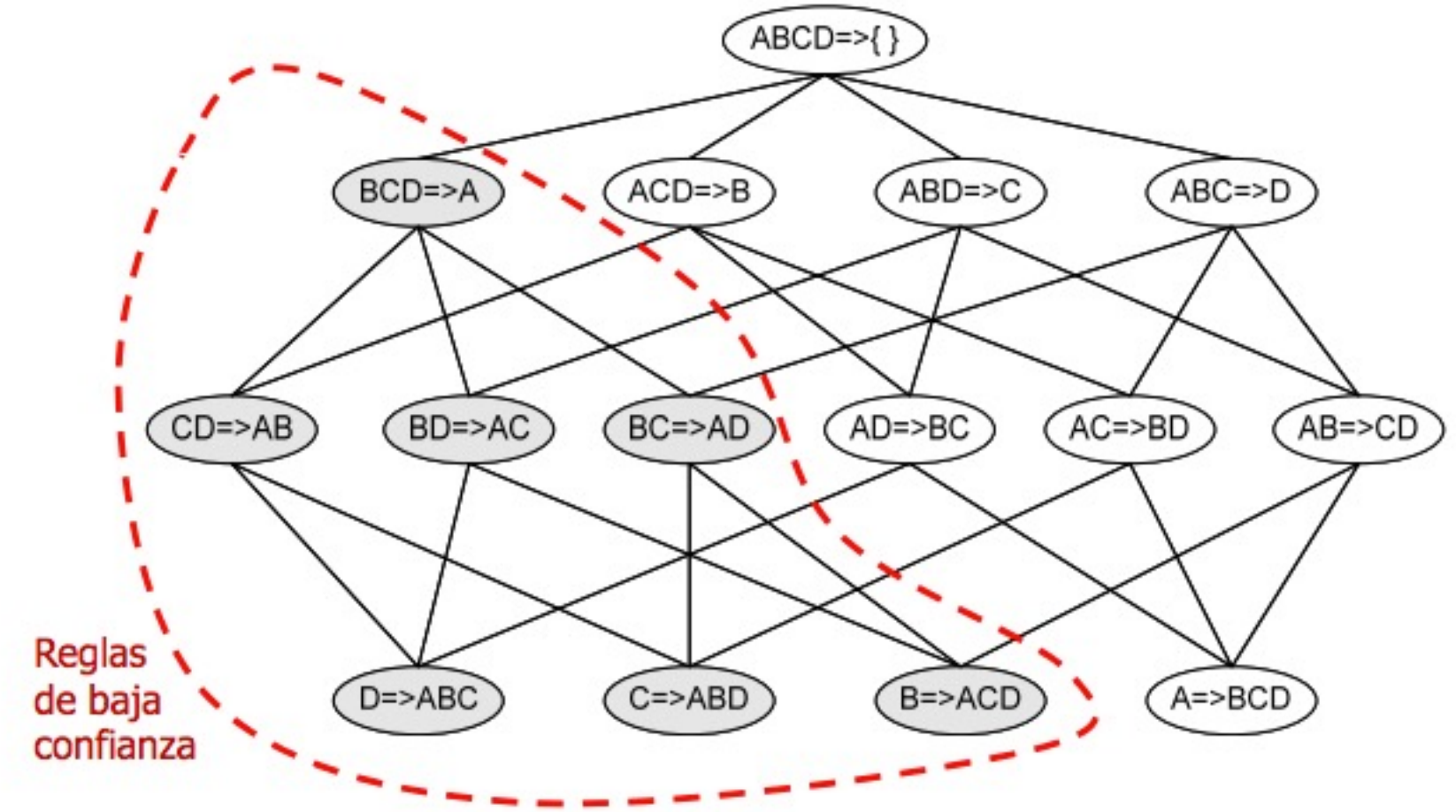
- La confianza no cumple la regla antimonótona.

  La confianza de A,B,C → D puede ser mayor o menor que la de A,B → D

- Si la regla ha sido generada de un mismo itemset entonces si se cumple la regla. L = {A,B,C,D} c(A,B,C → D) ≥ c(A,B → C,D)

- **OJO!!!** La confianza es antimonótona con respecto al número de items en la parte derecha de la regla.

Extracción de reglas



Algoritmo Apriori

Paso a paso (simplificado):

Dado el siguiente Dataset. Para un **frecuencia de soporte** mínimo de 2 y una **confianza mínima** del 60%.

ID	Transacción
T1	I1,I2,I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

Algoritmo Apriori

k=1

Se crea una tabla con el conteo de soporte de cada ítem en el Dataset (conjunto candidato).

Itemset	frec_sop
I1	6
I2	7
I3	6
I4	2
I5	2

Comparar con la frec. soporte mínima. Si la frecuencia de soporte es menor, eliminar los ítems. En este caso no se elimina ninguno.

Algoritmo Apriori

k=2. Se crea una tabla con la frecuencia de soporte de cada ítem en el dataset (conjunto candidato).

Itemset	frec_sop
I1,I2	4
I1,I3	4
I1,I4	1
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I3,I4	0
I3,I5	1
I4,I5	0

Algoritmo Apriori

Comparar con el soporte mínimo. Si la frecuencia de soporte es menor, eliminar los ítems.

Itemset	cont_sop
I1,I2	4
I1,I3	4
~~I1,I4~~	~~1~~
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
~~I3,I4~~	~~0~~
~~I3,I5~~	~~1~~
~~I4,I5~~	~~0~~

Algoritmo Apriori

Para k=3. Ej: {I1,I2,I4} no se tiene en cuenta por ser subset de {I1,I4}

Itemset	frec_sop
I1,I2,I3	2
I1,I2,I5	2
~~I1,I2,I4~~	
~~I2,I3,I4~~	
~~I2,I3,I5~~	
~~I3,14,15~~	
~~I1,I3,I5~~	

Todos tienen frec_sop>=sop_min=2

Para k=4.{I1,I2,I3,I5} contiene {I1,I3,I5}. Cómo este último no es frecuente, no lo podemos agregar. **HEMOS TERMINADO!!!!**

Algoritmo Apriori

Evaluamos las reglas:

Itemset	frec_sop
I1,I2,I3	2
I1,I2,I5	2

Para el itemset (I1,I2,I3), es decir su confianza:

$I1,I2 \rightarrow I3 = \sigma(I1,I2,I3)/\sigma(I1,I2) = 2/4 = 0,5 = 50\%$

$I1,I3 \rightarrow I2 = \sigma(I1,I2,I3)/\sigma(I1,I3) = 2/4 = 0,5 = 50\%$

$I2,I3 \rightarrow I1 = \sigma(I1,I2,I3)/\sigma(I2,I3) = 2/4 = 0,5 = 50\%$

$I1 \rightarrow I2,I3 = \sigma(I1,I2,I3)/\sigma(I1) = 2/6 = 0,33 = 33\%$

$I2 \rightarrow I1,I3 = \sigma(I1,I2,I3)/\sigma(I2) = 2/7 = 0,28 = 28\%$

$I3 \rightarrow I1,I2 = \sigma(I1,I2,I3)/\sigma(I3) = 2/6 = 0,33 = 33\%$

Algoritmo Apriori

Ejemplo con la lista de la compra **soporte mínimo** = 0,6 y **confianza** = 0,7

ID	Transacciones
1	Pan, leche, pañales
2	Pan, pañales, cerveza, huevos
3	Leche, pañales, cerveza, refresco, café
4	Pan, leche, pañales, cerveza
5	Pan, refresco, leche, pañales

Algoritmo Apriori

Para k=1. Se calcula la frecuencia.

Conjuntos	frec_sop
Cerveza	3
Pan	4
Refresco	2
Pañales	5
Leche	4
Huevos	1
Café	1

Algoritmo Apriori

Luego calculamos el soporte (frec_sop/num_transacciones).

Conjuntos	Soporte
Cerveza	0,6
Pan	0,8
Refresco	0,4
Pañales	1,0
Leche	0,8
Huevos	0,2
Café	0,2

Algoritmo Apriori

Eliminamos aquellas que no cumplen con el soporte mínimo. Refresco, huevos y café. Generamos k=2 con cerveza, pan, pañales y leche.

Conjunto	frec_sop	Soporte
Cerveza, Pan	2	0,4
Cerveza, Pañales	3	0,6
Cerveza, Leche	2	0,4
Pan, Pañales	4	0,8
Pan, Leche	3	0,6
Pañales, Leche	4	0,8

Algoritmo Apriori

Podamos aquellos que contienen subsets de k-1 que no cumplen con el soporte mínimo. Eliminamos aquellas que no cumplen con el soporte mínimo. Generamos k=3.

Conjunto	Conteo	Soporte
~~Cerveza, Pañales, Pan~~		
~~Cerveza, Pañales, Leche~~		
Pan, Pañales, Leche	3	0,6
~~Pan, Leche, Cerveza~~		

Para k=4 no podemos seguir. Por lo que generamos las reglas a partir de k=3.

Algoritmo Apriori

Por último obtenemos las posibles reglas y calculamos su confianza.

Teniendo en cuenta que conf_min es 0,7

Pan → Pañales, leche = 0,75

~~Pañales → Pan, leche = 0,6~~

Leche → Pan, pañales = 0,75

Pan, pañales → Leche = 0,75

Pan, leche → Pañales = 1

Pañales, leche → Pan = 0,75

Algoritmo Apriori

Ejercicio con comercios visitados en un centro comercial con el fin de optimizar el tiempo de los clientes. **Soporte mínimo** = 0,5 y **confianza** = 0,9

ID	Comercios visitados
1	Zapatería, electrónica, alimentación
2	Estética, electrónica, ropa
3	Zapatería, estética, electrónica, ropa
4	Estética, ropa

Algoritmo Apriori

Factores que influyen en la complejidad:

- Elegir el umbral de soporte mínimo:
  - Bajar el umbral de soporte resulta en mas itemsets frecuentes.
  - Esto puede incrementar el número de candidatos y la longitud máxima de itemsets frecuentes.
- Dimensionalidad del dataset (cantidad de items):
  - Se necesita más espacio para almacenar la frecuencia de soporte de cada uno de los items.
  - Si el número de items frecuentes aumenta, tanto el costo computacional como las operaciones se incrementan.
- Tamaño de la base de datos (transacciones):
  - Dado que Apriori hace varias pasadas, el tiempo de ejecución del algoritmo puede aumentar con el número de transacciones

Evaluación de patrones de asociación

Interpretación de las reglas:

- El número de reglas obtenidas en casos reales puede ser del orden de miles o millones.
- Muchas de ellas no tienen interés práctico al no revelar nada novedoso.
- Las conclusiones derivadas de una regla pueden ser débiles.

Evaluación de patrones de asociación

Tablas de contingencia (regla X → Y):

	Y	Not Y	
X	f_{11}	f_{10}	f_{1+}
Not X	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	T

f_{11}: número de veces que aparecen conjuntamente de X e Y

f_{10}: número de veces que aparece X y no Y

f_{01}: número de veces que aparece Y y no X

f_{00}: número de veces que no aparece X, ni Y

T=Total

Evaluación de patrones de asociación

Tablas de contingencia

	Café	Not Café	
Té	150	50	200
Not Té	750	50	800
	900	100	1000

Dada la regla Té → Café

Soporte (Té → Café) = 0,15

La confianza de Té → Café es de 75%.

¿Qué pensarías que ocurre con los que beben té?

Evaluación de patrones de asociación

Hipótesis: La gente que bebe té tienden a beber café.

	Café	Not Café	
Té	150	50	200
Not Té	750	50	800
	900	100	1000

Según la tabla de contingencia:

- Los bebedores de café son el 90% independientemente de si toman té o no. 900/1000=0,9
- Los bebedores de café en el caso de comprar té disminuye a un 15%. 150/1000=0,15

Por lo tanto la hipótesis sobre la regla no cuadra.

Evaluación de patrones de asociación

Independencia estadística. En una muestra de 1000 consumidores:

- 900 compran café (C).
- 200 compran té (T).
- 150 compran café y té (C AND T).

P(C AND T) = 150/1000 = 0,15

P(C)*P(T) = (900/1000)*(200/1000) = 0,9*0,2 = 0,18

P(C AND T) ≈ P(C)*P(T) → Hay independencia estadística. Azar.

P(C AND T) > P(C)*P(T) → Positivamente correlados. Complementarios.

P(C AND T) < P(C)*P(T) → Negativamente correlados. Sustitutivos

Evaluación de patrones de asociación

Medidas de evaluación que tienen en cuenta la dependencia estadística:

- Lift (también llamado Interest): cuantifica la relación existente entre X e Y.

$$\text{Lift} = \frac{\text{conf}(X,Y)}{\text{sop}(Y)}$$

Evaluación de patrones de asociación

Interpretación de Lift:

- Lift($X \rightarrow Y$) = 1, si X e Y son independientes.
- Lift($X \rightarrow Y$) < 1, si X e Y están negativamente correlados.
- Lift($X \rightarrow Y$) > 1, si X e Y están positivamente correlados.

Características de Lift:

- Cuando es igual a 1 quiere decir que hay independencia estadística. No nos dice nada.
- Es simétrica, Lift($X \rightarrow Y$) = Lift ($Y \rightarrow X$)

Evaluación de patrones de asociación

Ejemplo de Lift:

Para Té → Café

$$\text{Lift} = \frac{\text{conf(Té,Café)}}{\text{sop(Té)}} = 0{,}75/0{,}9 = 0{,}833 < 1 \rightarrow \text{Correlación negativa!!!}$$

Por lo tanto no es buena regla.

Patrones secuenciales

- **Objetivo:** extracción de patrones frecuentes relacionados con el tiempo u otro tipo de secuencia. Si sucede el evento X en el instante t de tiempo, el evento Y sucederá en el t+n.

Ejemplo: Si se compra una casa hay un 65% de posibilidades de comprar una nevera a las 2 semanas.

Patrones secuenciales

Set de datos	Secuencia	Transacción	Evento(Elemento)
Clientes	Historial de compras de un determinado cliente	Una lista de elementos comprados por el cliente	Libros, periódicos, CDs, etc
Datos Web	Actividad de navegación de un visitante de la página	Conjunto de páginas visualizados por el visitante una vez carga la Web	Página inicio, índice, info de contacto, etc.
Datos de eventos	Historia de eventos generados en un sensor	Eventos capturados por un sensor en un instante t	Tipos de alarmas generadas por el sensor
Secuencias genoma	Secuencia ADN de determinadas especies	Elemento de la secuencia del ADN	Bases A,T, G, C

Patrones secuenciales

Cliente	Fecha	Items
Martín	20/01/2017 10:13	Zumo, tarta
Martín	20/01/2017 11:47	Cerveza
Ramón	20/01/2017 14:32	Cerveza
Martín	21/01/2017 9:22	Vino, agua, sidra
Manolo	21/01/2017 10:34	Cerveza
Manolo	21/01/2017 17:27	Brandy
Ramón	21/01/2017 18:17	Vino, sidra
Ramón	22/01/2017 17:03	Brandy

Cliente	Items
Martín	(Zumo, tarta) (Cerveza) (Vino, agua, sidra)
Ramón	(Cerveza) (Vino, sidra) (Brandy)
Manolo	(Cerveza) (Brandy)

Patrones secuenciales

- Una secuencia es una lista ordenada de **transacciones**

$$s = < e_1\ e_2\ e_3 \ldots >$$

Longitud de secuencia, |s|: número de transacciones de la secuencia.
Una secuencia-k es una secuencia de k transacciones.

- Cada transacción tiene una colección de eventos (items) que se asignan a un instante.

$$e_i = \{i_1, i_2, \ldots, i_k\}$$

Patrones secuenciales

Cliente	Items
Martín	(Zumo, tarta) (Cerveza) (Vino, agua, sidra)
Ramón	(Cerveza) (Vino, sidra) (Brandy)
Manolo	(Cerveza) (Brandy)

Evento (item): Vino

Transacción: (Vino, agua, sidra)

Secuencia: (Zumo, tarta) (Cerveza) (Vino, agua, sidra)

Patrones secuenciales

• Subsecuencia: Una secuencia $<a_1\ a_2\ \ldots\ a_n>$ está contenida en otra secuencia $<b_1\ b_2\ \ldots\ b_m>$ $(m \geq n)$ si existen eventos $i_1 < i_2 < \ldots < i_n$ tales que $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, …, $a_n \subseteq b_{in}$

• La frecuencia de soporte de la subsecuencia, w, se define como la cantidad de secuencias que contienen w.

• Un patrón secuencial es una subsecuencia frecuente, o sea, una subsecuencia cuya frecuencia de soporte es ≥ soporte mínimo.

Patrones secuenciales

Secuencia	Subsecuencia	¿Contenida?
< {2,4} {3,5,6} {8} >	< {2} {3,5} >	Sí
< {1,2} {3,4} >	< {1} {2} >	No
< {2,4} {2,4} {2,5} >	< {2} {4} >	Sí

Ejemplo:

Secuencia
{Azúcar} {Dátiles}
{Azúcar} {Cacao}
{Azúcar, Espelta} {Cacao}
{Azúcar} {Boquerones} {Cacao, Dátiles, Fresas}

Frecuencia de soporte {Azúcar, Cacao}: 3

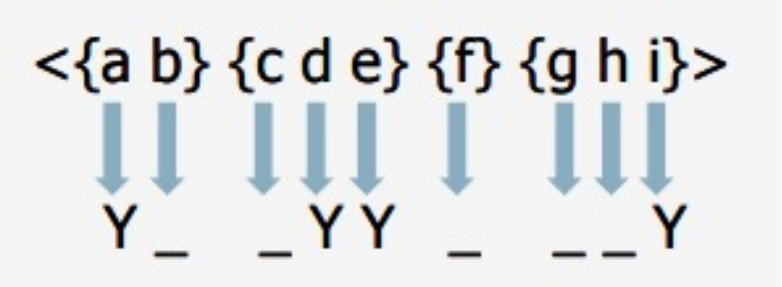
Será frecuente cuando el soporte mínimo sea por ejemplo 2

Patrones secuenciales

Dado un conjunto de secuencias y un umbral mínimo encontrar todas las subsecuencias frecuentes.

Complejidad computacional:

- Para un número de ítems n
- Para un número de transacciones k.



Si n=9 y k=4 entonces

$$\binom{n}{k} = \binom{9}{4} = 126$$

Algoritmo Generalized Sequential Patterns

Algoritmo GSP:

1. Obtener todas las secuencias frecuentes de 1 elemento.
2. Para k>=2. Mientras se encuentren nuevas secuencias frecuentes:
   - Generar k-secuencias candidatas a partir de las (k-1)-secuencias frecuentes.
   - Podar k-secuencias candidatas que contengan alguna (k-1)-secuencia no frecuente.
   - Recorrer nuevamente el conjunto de datos para obtener el soporte de las candidatas.
   - Eliminar las k-secuencias candidatas cuyo soporte real esté por debajo de MinSupp.

Algoritmo Generalized Sequential Patterns

Ejemplo:

Dado un set de datos con las siguientes secuencias de k=1: i_1 , i_2

Al emparejar 2 secuencias-1 $<\{i_1\}>$ e $<\{i_2\}>$ se producen 3 secuencias-2 candidatas:

$$<\{i_1\}\ \{i_2\}>,\ <\{i_2\}\ \{i_1\}>\ \text{e}\ <\{i_1\ i_2\}>$$

Algoritmo Generalized Sequential Patterns

Caso general (k>2):

• Las secuencias-k-1 frecuentes, w1 y w2 pueden emparejarse para producir una secuencia-k candidata si la subsecuencia obtenida al quitar el primer evento de w1 es la misma que la subsecuencia obtenida de quitar el ultimo evento en w2

• Entonces la secuencia-k candidata resultante viene dada por la secuencia w1 extendiendo el último evento de la w2

Si los dos últimos eventos de w2 pertenecen a la misma transacción, entonces el ultimo evento de w2 pasa a formar parte de la última transacción de w1.

En cualquier otro caso, el ultimo evento de w2 pasa a ser una transacción añadida al final de w1.

Algoritmo Generalized Sequential Patterns

w1=<{1} {2 3} {4}> y w2=<{2 3} {4 5}> producen <{1} {2 3} {4 5}> porque los 2 últimos eventos de w2 (4 y 5) son de la misma transacción.

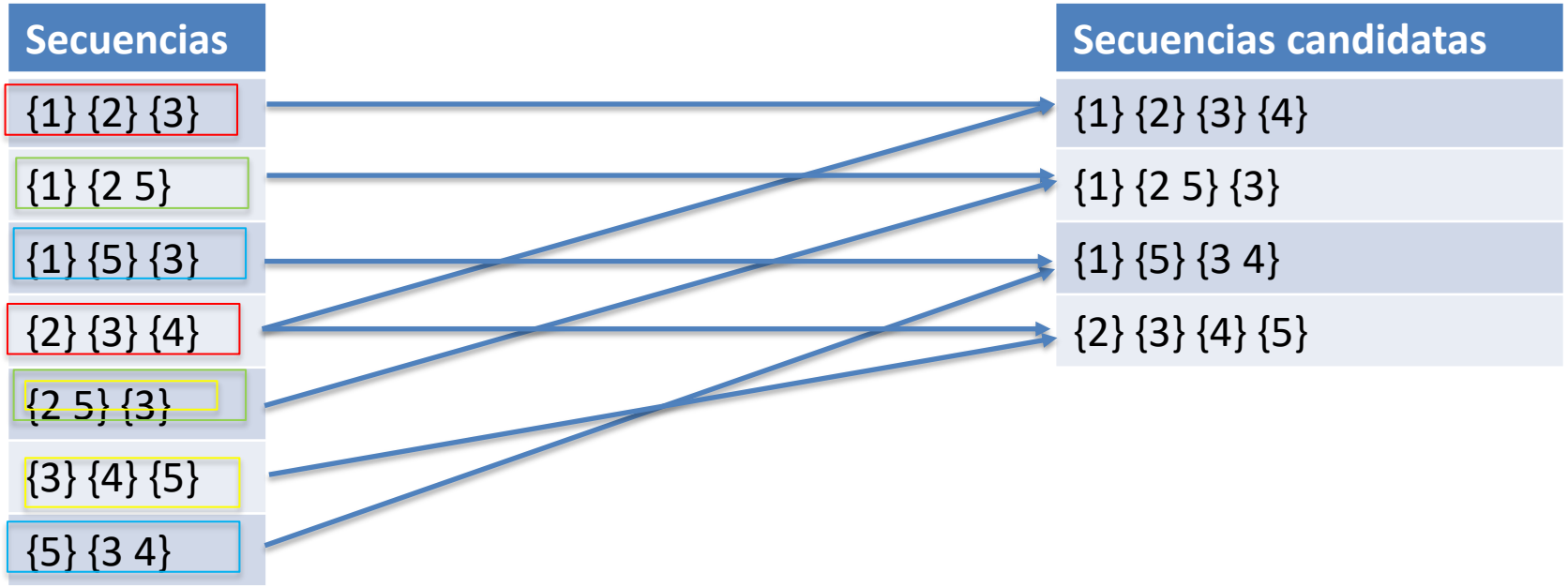
w1=<{1} {2 3} {4}> y w2=<{2 3} {4} {5}> producen <{1} {2 3} {4} {5}> porque los 2 últimos elementos de w2 (4 y 5) no son de la misma transacción.

En el caso de las secuencias w1=<{1} {2 6} {4}> y w2=<{1} {2} {4 5}> para producir la candidata <{1} {2 6} {4 5}> no emparejamos. Porque la subsecuencia obtenida no coincide. Se obtendría emparejando w1 con <{2 6} {4 5}>.

Algoritmo Generalized Sequential Patterns

1) Dadas las siguientes secuencias

2) Genero las secuencias candidatas



3) Calculo las secuencias frecuentes mediante el soporte.

Algoritmo Generalized Sequential Patterns

Dadas las siguientes secuencias y una frecuencia de soporte =2

Transaction Date	Customer ID	Items Purchased	Customer Sequence
1	01	A	<AB(FG)CD>
3	01	B	
7	01	FG	
9	01	C	
10	01	D	
1	02	B	<BGD>
4	02	G	
6	02	D	
1	03	B	<BFG(AB)>
5	03	F	
8	03	G	
9	03	AB	
2	04	F	<F(AB)CD>
6	04	AB	
8	04	C	
10	04	D	
3	05	A	<A(BC)GF(DE)>
4	05	BC	
7	05	G	
9	05	F	
10	05	DE	

Algoritmo Generalized Sequential Patterns

Para k=1 calculamos su frecuencia de soporte

Item	Support
A	4
B	5
C	3
D	4
~~E~~	~~1~~
F	4
G	4

Algoritmo Generalized Sequential Patterns

Generamos todas las posibles secuencias candidatas para k=2

Items comprados por separado

	A	B	C	D	F	G
A	AA	AB	AC	AD	AF	AG
B	BA	BB	BC	BD	BF	BG
C	CA	CB	CC	CD	CF	CG
D	DA	DB	DC	DD	DF	DG
F	FA	FB	FC	FD	FF	FG
G	GA	GB	GC	GD	GF	GG

Items comprados a la vez (OJO a los paréntesis)

	A	B	C	D	F	G
A		(AB)	(AC)	(AD)	(AF)	(AG)
B			(BC)	(BD)	(BF)	(BG)
C				(CD)	(CF)	(CG)
D					(DF)	(DG)
F						(FG)
G						

Algoritmo Generalized Sequential Patterns

Para las secuencias candidatas contamos su aparición en las secuencias del dataset.

Con la primera fila sale:

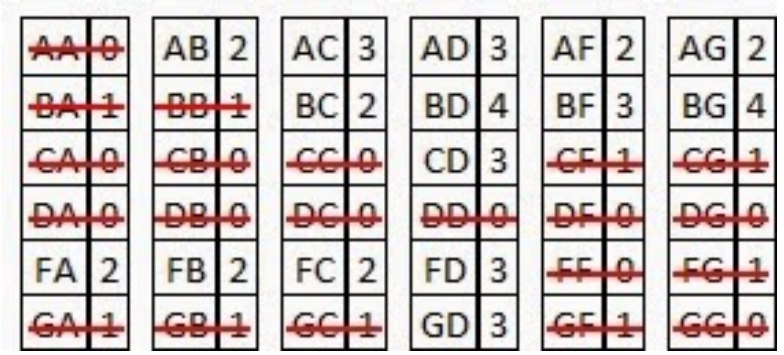
AA	AB	AC	AD	AF	AG
<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>
<BGD>	<BGD>	<BGD>	<BGD>	<BGD>	<BGD>
<BFG(AB)>	<BFG(AB)>	<BFG(AB)>	<BFG(AB)>	<BFG(AB)>	<BFG(AB)>
<F(AB)CD>	<F(AB)CD>	<F(AB)CD>	<F(AB)CD>	<F(AB)CD>	<F(AB)CD>
<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>

Repetir para el resto de secuencias del apartado anterior.

Algoritmo Generalized Sequential Patterns

Aquellas que no cumplan con el soporte mínimo se eliminan.

Con los que van por separado:



~~AA~~	~~0~~	AB	2	AC	3	AD	3	AF	2	AG	2
~~BA~~	~~1~~	~~BB~~	~~1~~	BC	2	BD	4	BF	3	BG	4
~~CA~~	~~0~~	~~CB~~	~~0~~	~~CC~~	~~0~~	CD	3	~~CF~~	~~1~~	~~CG~~	~~1~~
~~DA~~	~~0~~	~~DB~~	~~0~~	~~DC~~	~~0~~	~~DD~~	~~0~~	~~DF~~	~~0~~	~~DG~~	~~0~~
FA	2	FB	2	FC	2	FD	3	~~FF~~	~~0~~	~~FG~~	~~1~~
~~GA~~	~~1~~	~~GB~~	~~1~~	~~GC~~	~~1~~	GD	3	~~GF~~	~~1~~	~~GG~~	~~0~~

En el caso de los que van juntos, solo (AB) aparece en 2 secuencias del dataset.

Algoritmo Generalized Sequential Patterns

Generar secuencias candidatas para k>2.

2-seq	-1st	-Last
AB	B	A
AC	C	A
AD	D	A
AF	F	A
AG	G	A
BC	C	B
BD	D	B
BF	F	B
BG	G	B
CD	D	C
FA	A	F
FB	B	F
FC	C	F
FD	D	F
GD	D	G
(AB)	B	A
	A	B

-1st → Menos el primero.

- Last → Menos el último.

OJO!!! Con (AB) hay dos combinaciones al no tener orden.

Para AB cómo -1st es B, se puede combinar con aquellas que –Last es también B. En este caso BC, BD, BF, BG y (AB)

Algoritmo Generalized Sequential Patterns

Podar aquellas que sean subsecuencias de secuencias que no cumplieron con la condición de soporte mínimo en k=2 y calcular el soporte.

2-seq. (1)	2-seq. -1st	2-seq. (2)	2-seq. -Last	3-seq after join	3-seq. after prune	Support Count	3-seq. Supported
AB	B	BC	B	ABC	ABC	1	
AB	B	BD	B	ABD	ABD	2	ABD
AB	B	BF	B	ABF	ABF	2	ABF
AB	B	BG	B	ABG	ABG	2	ABG
AB	B	(AB)	B	A(AB)			
AC	C	CD	C	ACD	ACD	3	ACD
AF	F	FA	F	AFA			
AF	F	FB	F	AFB	AFB	0	
AF	F	FC	F	AFC	AFC	1	
AF	F	FD	F	AFD	AFD	2	AFD
AG	G	GD	G	AGD	AGD	2	AGD
BC	C	CD	C	BCD	BCD	2	BCD
BF	F	FA	F	BFA			
BF	F	FB	F	BFB			
BF	F	FC	F	BFC	BFC	1	
BF	F	FD	F	BFD	BFD	2	BFD
BG	G	GD	G	BGD	BGD	3	BGD

A(AB) se poda porque AA no cumplió con el soporte mínimo

Algoritmo Generalized Sequential Patterns

Segunda parte de la tabla anterior.

FA	A	AB	A	FAB	FAB	0	
FA	A	AC	A	FAC	FAC	1	
FA	A	AD	A	FAD	FAD	1	
FA	A	AF	A	FAF			
FA	A	AG	A	FAG			
FA	A	(AB)	A	F(AB)	F(AB)	2	F(AB)
FB	B	BC	B	FBC	FBC	1	
FB	B	BD	B	FBD	FBD	1	
FB	B	BF	B	FBF			
FB	B	BG	B	FBG			
FC	C	CD	C	FCD	FCD	2	FCD
(AB)	B	BC	B	(AB)C	(AB)C	1	
(AB)	B	BD	B	(AB)D	(AB)D	1	
(AB)	B	BF	B	(AB)F	(AB)F	0	
(AB)	B	BG	B	(AB)G	(AB)G	0	
(AB)	A	AB	A	(AB)B			

Aquí FAF se poda porque FF no cumplió con el soporte mínimo

Algoritmo Generalized Sequential Patterns

Llegados a este punto volvemos a repetir los pasos anteriores.

3-seq	-1st	-Last
ABD	BD	AB
ABF	BF	AB
ABG	BG	AB
ACD	CD	AC
AFD	FD	AF
AGD	GD	AG
BCD	CD	BC
BFD	FD	BF
BGD	GD	BG
F(AB)	(AB)	FA
		FB
FCD	CD	FC

3-seq. (1)	3-seq. -1st	3-seq. (2)	3-seq. -Last	4-seq. after join	4-seq. after prune	Support Count	4-seq. Supported
ABF	BF	BFD	BF	ABFD	ABFD	2	ABFD
ABG	BG	BGD	BG	ABGD	ABGD	2	ABGD

Sólo podemos generar nuevas secuencias con ABF, BFD y ABG, BGD. En el resto de los casos no coinciden -1st y –Last.

Algoritmo Generalized Sequential Patterns

Para k=5 no se pueden generar más secuencias.

Sequences			
1-Item	2-Items	3-Items	4-Items
A	AB	ABD	ABFD
B	AC	ABF	ABGD
C	AD	ABG	
D	AF	ACD	
F	AG	AFD	
G	BC	AGD	
	BD	BCD	
	BF	BFD	
	BG	BGD	
	CD	F(AB)	
	FA	FCD	
	FB		
	FC		
	FD		
	GD		
	(AB)		