Reglas de Asociación (I)

A. Mora, M. Ojeda

Departamento de Matemática Aplicada
Universidad de Málaga

Reglas de asociación

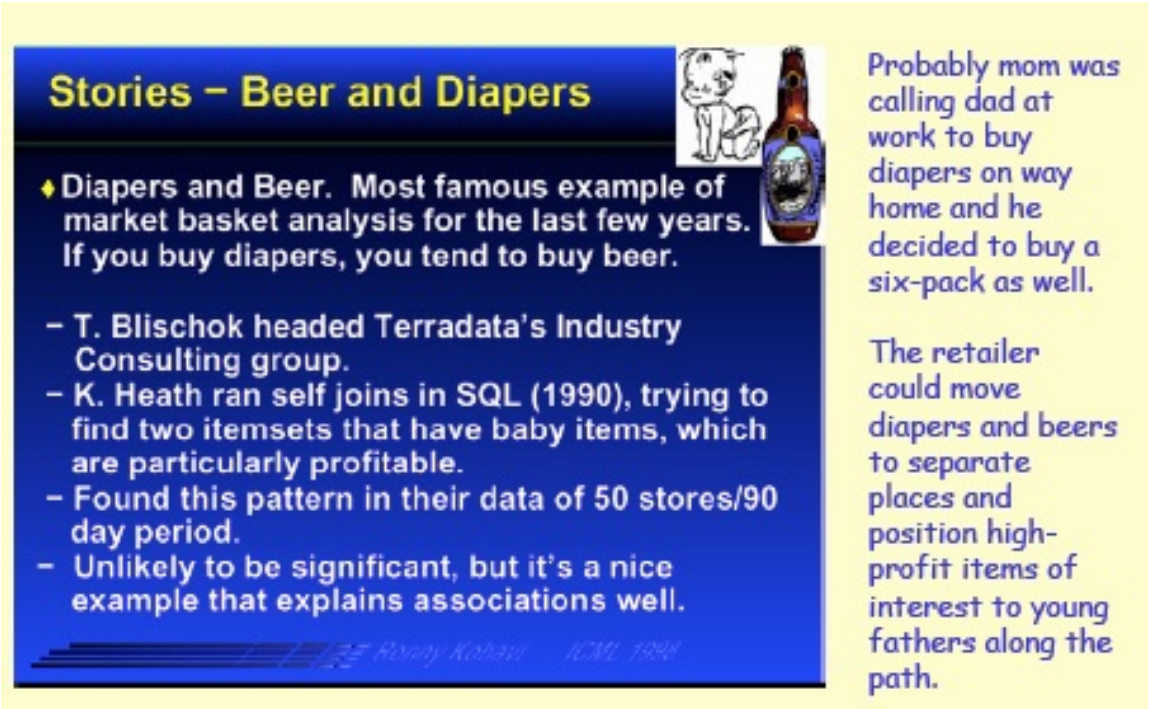
Búsqueda de patrones

- *Objetivo:* encontrar relaciones entre datos: market basket, graphs, streams, etc.
- Son el elemento central en muchas técnicas en muy diversas áreas desde base de datos, IA, data mining, machine learning, etc.
- Los métodos de búsqueda de patrones frecuentes surgen en el área de data mining.
- El primer algoritmo a destacar fue el denominado Apriori.

Reglas de asociación

Orígenes

- Surgen con el estudio del llamado `Market Basket Analysis`
- Encontrar coincidencias entre productos comprados a la vez.
- Predecir cuando la compra de un producto puede incrementar la compra de otros.
- Cada fila sería una transacción (comprador) y las columnas los distintos atributos (productos).



Reglas de asociación: Aplicaciones

- Descubrir hábitos de clientes al encontrar relaciones entre diferentes items que los consumidores colocan en su cesta de la compra en grandes supermercados.
- Análisis de compras en grandes tiendas on-line -Amazon (miles de productos con millones de consumidores), Google, etc, con el objetivo de ofrecer recomendaciones.
- Detectar relaciones para dar sugerencias entre las canciones, series, películas, etc. que un usuario escucha/ve en una aplicación de internet de música on-line, o de videos - Spotify, Netflix (25 millones de usuarios), HBO, etc.
- Marketing, diseño de catálogos.
- Análisis de prestamos, detección de fraudes, etc.

Patrones

Notación en market basket analysis

- Datos: Típicamente una matriz binaria D dispersa de tamaño muy grande.
- Item: Cada columna de dataset. Un dato del dataset que puede ser un atributo, una palabra, un documento, un biomarcador, etc.
- Transacción: Cada fila del dataset representando típicamente un producto.
- Itemset: una colección de cero o más items.

$$D_{ij} = 1 \text{ si item } j \text{ ha comprado el producto } i$$

Objetivo: Dado un vector de bits nuevo representado lo que el consumidor ha comprado, se trata de predecir en qué otros productos podría estar interesado en comprar.

- Sistemas de recomendación.
- Modelizar dependencias en sistemas complejos de sw: si se sabe que ficheros han sido cambiados que otros pueden ser necesarios adaptar.

Frequent Itemset mining

Minería de Reglas de Asociación

- Confianza : Una medida de la incertidumbre del patrón descubierto.
- Soporte: Una medida de en qué porcentaje el patrón encontrado aparece respecto al tamaño del dataset.
- Frequent itemset : Establecer un umbral respecto al soporte que debe satisfacer el itemset encontrado.
- Strong Association rules: Reglas que satisfacen umbrales de soporte y de confianza.

Reglas de asociación

Id	*Leche*	*Pan*	*Mantequilla*	*Cerveza*
1	1	1	0	0
2	0	1	1	0
3	0	0	0	1
4	1	1	1	0
5	0	1	0	0

Regla de Asociación: $X \rightarrow Y$ **()**

$$Leche, Pan \rightarrow Mantequilla$$

Definition

Una regla de asociación es una relación entre atributos que representa un patrón frecuente, una correlación estructural entre dichos atributos (items) en una tabla (base de datos, dataset).

Reglas de asociación

Id	*Leche*	*Pan*	*Mantequilla*	*Cerveza*
1	1	1	0	0
2	0	1	1	0
3	0	0	0	1
4	1	1	1	0
5	0	1	0	0

Regla de Asociación: $X \rightarrow Y$ **()**

$$Leche, Pan \rightarrow Mantequilla$$

Definition

Una regla de asociación es una relación entre atributos que representa un patrón frecuente, una correlación estructural entre dichos atributos (items) en una tabla (base de datos, dataset).

Reglas de asociación: Forma

Regla de Asociación: $X \rightarrow Y$ (Sop,Conf)

$$Antecedente \rightarrow Consecuente, \quad [Soporte, Confianza]$$

Definition (Soporte)

$$Sop(X) = \frac{|\ X\ |}{|\ D\ |}$$

$|\ X\ |$ número de filas con X y $|\ D\ |$ el número de filas de la tabla.

El soporte denota la frecuencia de la regla en el dataset, $P(X \cup Y)$.

Definition (Confianza)

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

La confianza es un estimador de la $P(Y|X)$, es decir el porcentaje del dataset que conteniendo X también contiene a Y.

Reglas de asociación

Id	*Leche*	*Pan*	*Mantequilla*	*Cerveza*
1	1	1	0	0
2	0	1	1	0
3	0	0	0	1
4	1	1	1	0
5	0	1	0	0

$$Sop(Leche, Pan) = \frac{2}{5} = 0.4$$

$$Conf\,(Leche, Pan \rightarrow Mantequilla) = \frac{1}{2} = 0.5$$

Regla de Asociación: $X \rightarrow Y$, (Sop, Conf)

$$Leche, Pan \rightarrow Mantequilla, \quad (0.4, 0.5)$$

Reglas de asociación: Item-set

- Buscar combinaciones de pares atributo-valor con suficiente soporte. *Se define un* $soporte_{min}$: **frequent itemset**
- **Generar, a partir de ellas, reglas con suficiente confianza.**

Item-set

- Item: par atributo-valor
- Item set: conjunto de pares atributo-valor
- K-item set: conjunto con k items
- Generación de item-sets con soporte mínimo: 1-item sets, 2-item sets, 3-item sets...
- Motivación de estrategias de búsqueda de reglas:
  - Para que un k-item set tenga soporte mínimo es necesario que todos sus (k-1) item sets tengan soporte mínimo.
  - Pero esto no es suficiente: hay comprobarlo en el dataset cada vez.

Reglas de asociación: Estrategia Apriori

Problemas:

El espacio de todas las reglas de asociación es exponencial, $O(2^n)$, donde n es el número de atributos en el dataset.
Se pueden producir un gran número de reglas, miles, decenas de miles, millones, etc.

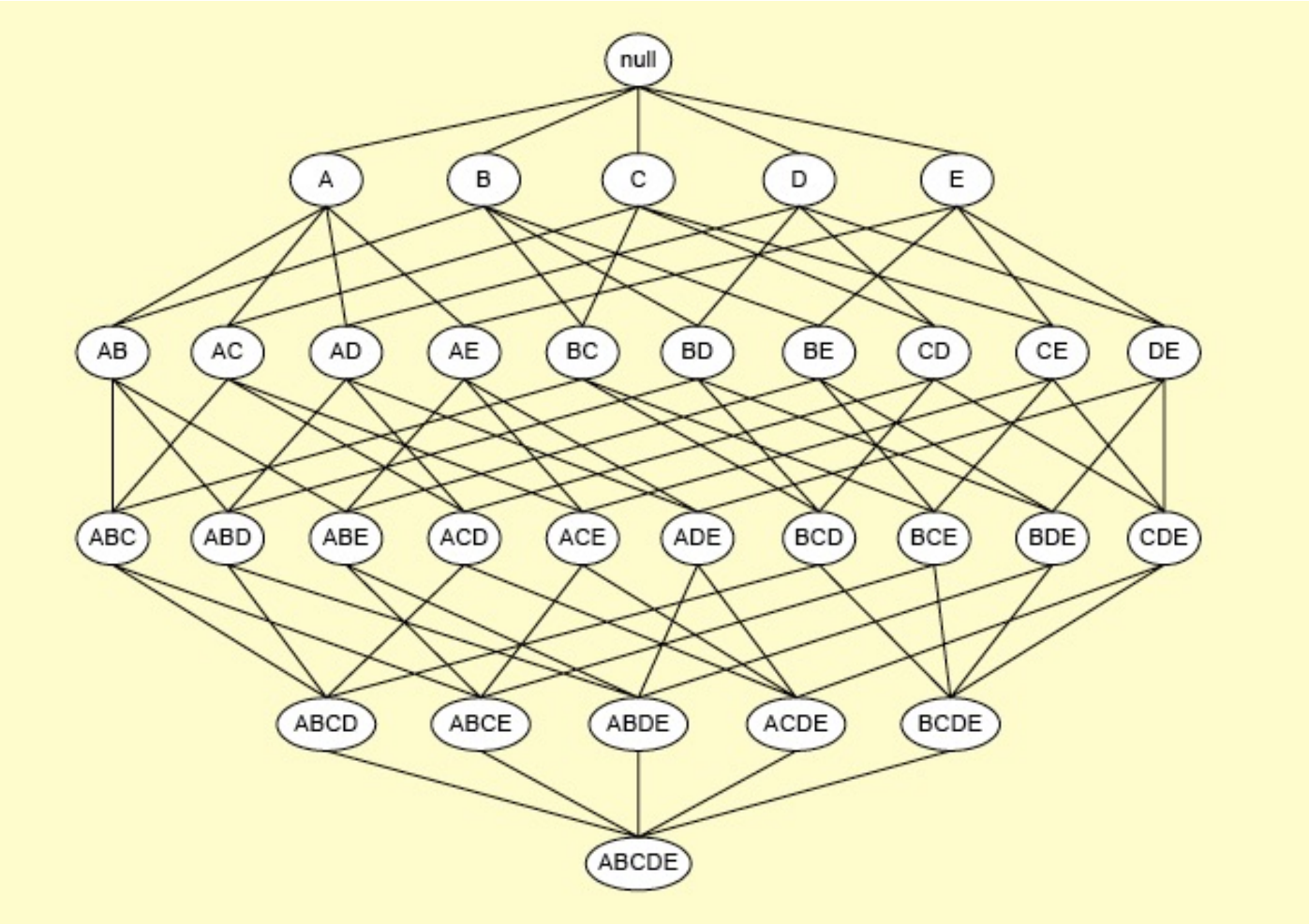
El objetivo será evitar ir al dataset para comprobar el soporte y podar un gran número de conjuntos frecuentes.

Estrategia de poda: "Apriori principle"

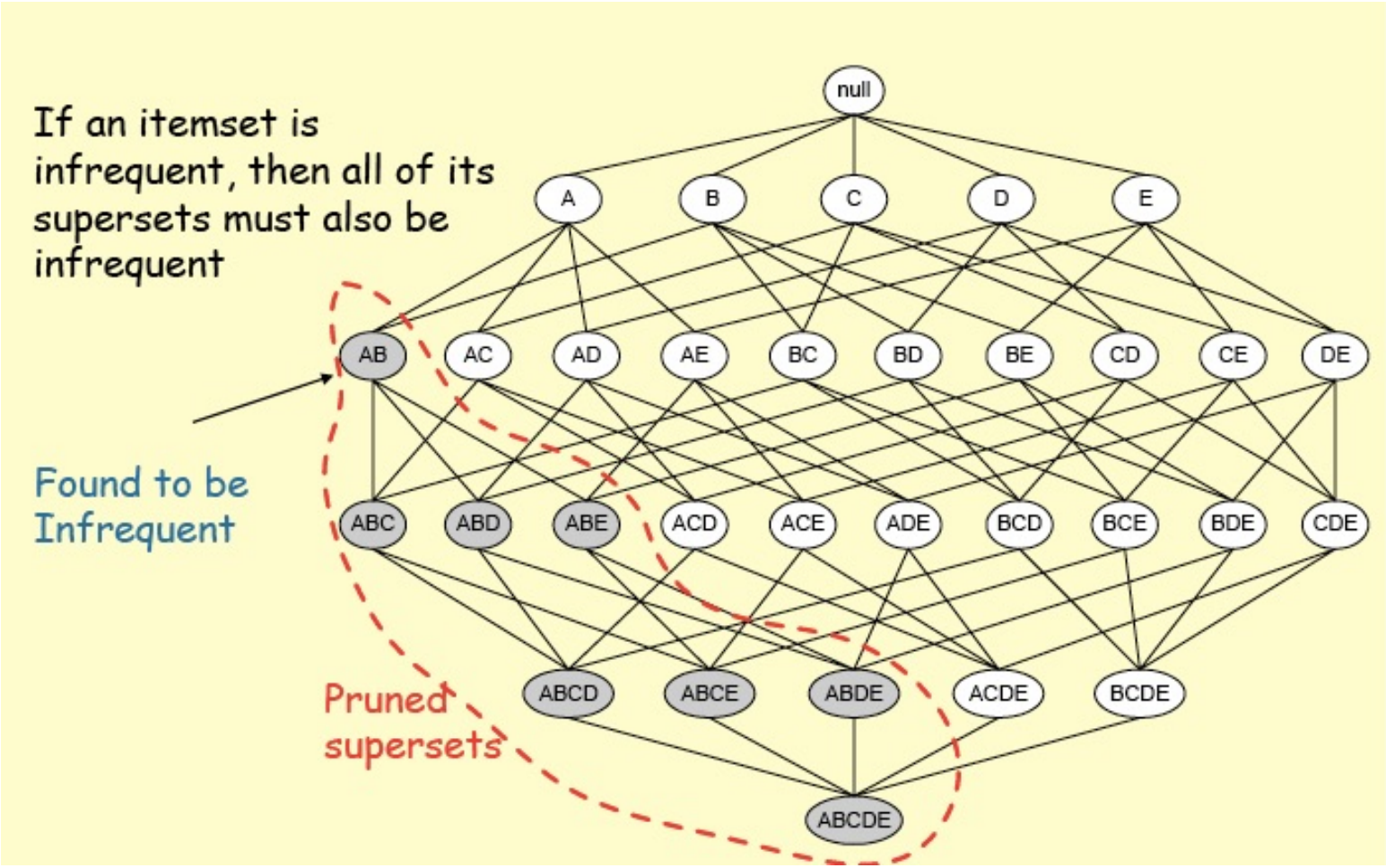
Un subconjunto de un itemset frecuente debe ser frecuente.

- Una fila que contenga $\{beer, diaper, nuts\}$ contiene $\{beer, diaper\}$.
- Si $\{beer, diaper, nuts\}$ es frecuente entonces $\{beer, diaper\}$ es frecuente.

Reglas de asociación: Retículo de items



Reglas de asociación: Retículo de items



Reglas de asociación: Algoritmo Apriori

Rakesh Agrawal

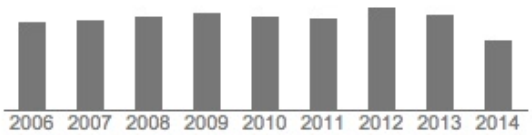
Follow

Technical Fellow, Microsoft Research
Data Mining, Web Search, Education, Privacy
Verified email at microsoft.com

Title 1–20	Cited by	Year
Fast algorithms for mining association rules R Agrawal, R Srikant Proc. 20th int. conf. very large data bases, VLDB 1215, 487-499	16995	1994
Mining association rules between sets of items in large databases R Agrawal, T Imieliński, A Swami ACM SIGMOD Record 22 (2), 207-216	14975	1993
Mining sequential patterns R Agrawal, R Srikant Data Engineering, 1995. Proceedings of the Eleventh International Conference ...	5281	1995
Fast Discovery of Association Rules. R Agrawal, H Mannila, R Srikant, H Toivonen, AI Verkamo Advances in knowledge discovery and data mining 12 (1), 307-328	2813	1996

Google Scholar

Citation indices	All	Since 2009
Citations	84005	36468
h-index	89	61
i10-index	203	151



Reglas de asociación: Algoritmo Apriori

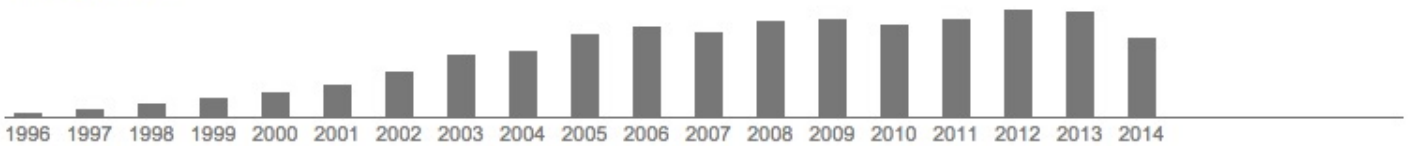
Scholar ← Export

Rakesh Agrawal

Fast algorithms for mining association rules

[PDF] from uu.se

Authors	Rakesh Agrawal, Ramakrishnan Srikant
Publication date	1994/9/12
Journal	Proc. 20th int. conf. very large data bases, VLDB
Volume	1215
Pages	487-499
Description	Abstract We consider the problem of discovering association rules between items in a large database of sales transactions. We present two new algorithms for solving this problem that are fundamentally di erent from the known algorithms. Experiments with synthetic as well as real-life data show that these algorithms outperform the known algorithms by factors ranging from three for small problems to more than an order of magnitude for large problems. We also show how the best features of the two proposed algorithms can be combined into a ...
Total citations	Cited by 16995



Reglas de asociación: Algoritmo Apriori

- Encontrar los itemset frecuentes: los conjuntos de elementos que tienen apoyo mínimo.
  - Un subconjunto de un itemset frecuente también debe ser un itemset frecuente
    si AB es un itemset frecuente, ambos A y B deben ser itemset frecuentes
  - Iterativamente encontrar itemset frecuentes con cardinalidad 1 a k (k-itemset)
- Utilizar los itemset frecuentes para generar reglas de asociación.

Algoritmo Apriori en R - Conjuntos frecuentes

```
Apriori <- function (data, I, MIN_SUP, parameter = NULL){
        f <- CreateItemsets()
        c <- FindFrequentItemset(data,I,1, MIN_SUP)
        k <- 2
        len4data <- GetDatasetSize(data)
        while( !IsEmpty(c[[k-1]]) ){
           f[[k]] <- AprioriGen(c[k-1])
           for( idx in 1: len4data ){
               ft <- GetSubSet(f[[k]],data[[idx]])
               len4ft <- GetDatasetSize(ft)
               for( jdx in 1:len4ft ){
                     IncreaseSupportCount(f[[k]],ft[jdx])
               }
            }
           c[[k]] <- FindFrequentItemset(f[[k]],I,k,MIN_SUP)
           k <- k+1
           }
        c
}
```

Ejemplo: Algoritmo Apriori

Id	$a1$	$a2$	$a3$	$a4$	$a5$
id_1	1	1	0	0	1
id_2	0	1	0	1	0
id_3	0	1	1	0	0
id_4	1	1	0	1	0
id_5	1	0	1	0	0
id_6	0	1	1	0	0
id_7	1	0	1	0	0
id_8	1	1	1	0	1
id_9	1	1	1	0	0

Algoritmo Apriori

- Vamos a establecer soporte mínimo de 2 (2 de 9) y una confianza del 70 por ciento.
- Encontrar los conjuntos frecuentes.
- Generar las reglas de asociación teniendo en cuenta el soporte y la confianza determinadas.

Ejemplo: Algoritmo Apriori

Id	*a*1	*a*2	*a*3	*a*4	*a*5
id_1	1	1	0	0	1
id_2	0	1	0	1	0
id_3	0	1	1	0	0
id_4	1	1	0	1	0
id_5	1	0	1	0	0
id_6	0	1	1	0	0
id_7	1	0	1	0	0
id_8	1	1	1	0	1
id_9	1	1	1	0	0

Itemset	*Sop.Paso*1
a1	6
a2	7
a3	6
a4	2
a5	2

Ejemplo: Algoritmo Apriori

Id	*a*1	*a*2	*a*3	*a*4	*a*5
id_1	1	1	0	0	1
id_2	0	1	0	1	0
id_3	0	1	1	0	0
id_4	1	1	0	1	0
id_5	1	0	1	0	0
id_6	0	1	1	0	0
id_7	1	0	1	0	0
id_8	1	1	1	0	1
id_9	1	1	1	0	0

Acceso a dataset	*Itemset*	*Sop.Paso*1
1	a1	6 ✓
2	a2	7✓
3	a3	6✓
4	a4	2✓
5	a5	2✓

Paso 1: candidatos a siguiente nivel

$$\text{1-itemset: } C_1 = \{a1, a2, a3, a4, a5\}$$

5 lecturas enteras del dataset

Ejemplo: Algoritmo Apriori

Id	*a*1	*a*2	*a*3	*a*4	*a*5
id_1	1	1	0	0	1
id_2	0	1	0	1	0
id_3	0	1	1	0	0
id_4	1	1	0	1	0
id_5	1	0	1	0	0
id_6	0	1	1	0	0
id_7	1	0	1	0	0
id_8	1	1	1	0	1
id_9	1	1	1	0	0

Acceso	*Itemset*	*Sop.Paso2*
6	a1a2	4 ✓
7	a1a3	4✓
8	a1a4	1✗
9	a1a5	2✓
10	a2a3	4✓
11	a2a4	2✓
12	a2a5	2✓
13	a3a4	0✗
14	a3a5	1✗
15	a4a5	0✗

Paso 2: candidatos a siguiente nivel

$$\text{2-itemset: } C_2 = \{a1a2, a1a3, a1a5, a2a3, a2a4, a2a5\}$$

15 lecturas del dataset

Ejemplo: Algoritmo Apriori

Id	*a*1	*a*2	*a*3	*a*4	*a*5
id_1	1	1	0	0	1
id_2	0	1	0	1	0
id_3	0	1	1	0	0
id_4	1	1	0	1	0
id_5	1	0	1	0	0
id_6	0	1	1	0	0
id_7	1	0	1	0	0
id_8	1	1	1	0	1
id_9	1	1	1	0	0

Acceso	*Itemset*	*Sop.Paso2*
6	a1a2	4 ✓
7	a1a3	4✓
8	a1a4	1✗
9	a1a5	2✓
10	a2a3	4✓
11	a2a4	2✓
12	a2a5	2✓
13	a3a4	0✗
14	a3a5	1✗
15	a4a5	0✗

Paso 2: candidatos a siguiente nivel

$$\text{2-itemset: } C_2 = \{a1a2, a1a3, a1a5, a2a3, a2a4, a2a5\}$$

15 lecturas del dataset

Ejemplo: Algoritmo Apriori

Generación de candidatos en nivel 3

- Partimos de $C_2 = \{a1a2, a1a3, a1a5, a2a3, a2a4, a2a5\}$ el conjunto 2-itemset.
- Para todo $A, B \in C_2$ el conjunto estará formado por candidatos $A \cup B$.
- Posibles candidatos en C_3 serían $Candidatos = \{a1a2a3, a1a2a5, a1a3a5, a2a3a4, a2a3a5, a2a4a5\}$.
- Utilizamos la propiedad esencial de **poda** del algoritmo **Apriori**:
- Tras la poda los conjuntos frecuentes de tamaño 3 (3-itemset) son $C_3 = \{a1a2a3, a1a2a5\}$.

Ejemplo: Algoritmo Apriori

Generación de candidatos en nivel 3

- Partimos de $C_2 = \{a1a2, a1a3, a1a5, a2a3, a2a4, a2a5\}$ el conjunto 2-itemset.
- Para todo $A, B \in C_2$ el conjunto estará formado por candidatos $A \cup B$.
- Posibles candidatos en C_3 serían
  $Candidatos = \{a1a2a3, a1a2a5, a1a3a5, a2a3a4, a2a3a5, a2a4a5\}$.
- Utilizamos la propiedad esencial de **poda** del algoritmo **Apriori**:
  - $a1a2a3 \in C_3$ porque $a1a2, a1a3, a2a3$ es decir todos sus subconjuntos de tamaño 2 pertenecen a C_2.
- Tras la poda los conjuntos frecuentes de tamaño 3 (3-itemset) son
  $C_3 = \{a1a2a3, a1a2a5\}$.

Ejemplo: Algoritmo Apriori

Generación de candidatos en nivel 3

- Partimos de $C_2 = \{a1a2, a1a3, a1a5, a2a3, a2a4, a2a5\}$ el conjunto 2-itemset.
- Para todo $A, B \in C_2$ el conjunto estará formado por candidatos $A \cup B$.
- Posibles candidatos en C_3 serían $Candidatos = \{a1a2a3, a1a2a5, a1a3a5, a2a3a4, a2a3a5, a2a4a5\}$.
- Utilizamos la propiedad esencial de **poda** del algoritmo **Apriori**:
  1. $a1a2a3 \in C_3$ porque $a1a2, a1a3, a2a3$ es decir todos sus subconjuntos de tamaño 2 pertenecen a C_2.
  2. $a2a3a5 \notin C_3$ porque $a3a5$ no está en C_2.
- Tras la poda los conjuntos frecuentes de tamaño 3 (3-itemset) son $C_3 = \{a1a2a3, a1a2a5\}$.

Ejemplo: Algoritmo Apriori

Generación de candidatos en nivel 3

- Partimos de $C_2 = \{a1a2, a1a3, a1a5, a2a3, a2a4, a2a5\}$ el conjunto 2-itemset.
- Para todo $A, B \in C_2$ el conjunto estará formado por candidatos $A \cup B$.
- Posibles candidatos en C_3 serían $Candidatos = \{a1a2a3, a1a2a5, a1a3a5, a2a3a4, a2a3a5, a2a4a5\}$.
- Utilizamos la propiedad esencial de **poda** del algoritmo **Apriori**:
  1. $a1a2a3 \in C_3$ porque $a1a2, a1a3, a2a3$ es decir todos sus subconjuntos de tamaño 2 pertenecen a C_2.
  2. $a2a3a5 \notin C_3$ porque $a3a5$ no está en C_2.
- Tras la poda los conjuntos frecuentes de tamaño 3 (3-itemset) son $C_3 = \{a1a2a3, a1a2a5\}$.

Ejemplo: Algoritmo Apriori

Generación de candidatos en nivel 3

- Partimos de $C_2 = \{a1a2, a1a3, a1a5, a2a3, a2a4, a2a5\}$ el conjunto 2-itemset.
- Para todo $A, B \in C_2$ el conjunto estará formado por candidatos $A \cup B$.
- Posibles candidatos en C_3 serían $Candidatos = \{a1a2a3, a1a2a5, a1a3a5, a2a3a4, a2a3a5, a2a4a5\}$.
- Utilizamos la propiedad esencial de **poda** del algoritmo **Apriori**:
  1. $a1a2a3 \in C_3$ porque $a1a2, a1a3, a2a3$ es decir todos sus subconjuntos de tamaño 2 pertenecen a C_2.
  2. $a2a3a5 \notin C_3$ porque $a3a5$ no está en C_2.
- Tras la poda los conjuntos frecuentes de tamaño 3 (3-itemset) son $C_3 = \{a1a2a3, a1a2a5\}$.

Ejemplo: Algoritmo Apriori

Generación de candidatos en nivel 3

- Partimos de $C_2 = \{a1a2, a1a3, a1a5, a2a3, a2a4, a2a5\}$ el conjunto 2-itemset.
- Para todo $A, B \in C_2$ el conjunto estará formado por candidatos $A \cup B$.
- Posibles candidatos en C_3 serían $Candidatos = \{a1a2a3, a1a2a5, a1a3a5, a2a3a4, a2a3a5, a2a4a5\}$.
- Utilizamos la propiedad esencial de **poda** del algoritmo **Apriori**:
  1. $a1a2a3 \in C_3$ porque $a1a2, a1a3, a2a3$ es decir todos sus subconjuntos de tamaño 2 pertenecen a C_2.
  2. $a2a3a5 \notin C_3$ porque $a3a5$ no está en C_2.
- Tras la poda los conjuntos frecuentes de tamaño 3 (3-itemset) son $C_3 = \{a1a2a3, a1a2a5\}$.

Ejemplo: Algoritmo Apriori

Generación de candidatos en nivel 3

- Partimos de $C_2 = \{a1a2, a1a3, a1a5, a2a3, a2a4, a2a5\}$ el conjunto 2-itemset.
- Para todo $A, B \in C_2$ el conjunto estará formado por candidatos $A \cup B$.
- Posibles candidatos en C_3 serían $Candidatos = \{a1a2a3, a1a2a5, a1a3a5, a2a3a4, a2a3a5, a2a4a5\}$.
- Utilizamos la propiedad esencial de **poda** del algoritmo **Apriori**:
  1. $a1a2a3 \in C_3$ porque $a1a2, a1a3, a2a3$ es decir todos sus subconjuntos de tamaño 2 pertenecen a C_2.
  2. $a2a3a5 \notin C_3$ porque $a3a5$ no está en C_2.
- Tras la poda los conjuntos frecuentes de tamaño 3 (3-itemset) son $C_3 = \{a1a2a3, a1a2a5\}$.

Ejemplo: Algoritmo Apriori

Id	*a*1	*a*2	*a*3	*a*4	*a*5
id_1	1	1	0	0	1
id_2	0	1	0	1	0
id_3	0	1	1	0	0
id_4	1	1	0	1	0
id_5	1	0	1	0	0
id_6	0	1	1	0	0
id_7	1	0	1	0	0
id_8	1	1	1	0	1
id_9	1	1	1	0	0

Acceso	*Itemset*	*Sop.Paso*3
16	a1a2a3	2 ✓
17	a1a2a5	2 ✓

Paso 3: candidatos a siguiente nivel

$$3 - itemset = \{a1a2a3, a1a2a5\}$$

17 lecturas completas del dataset

Ejemplo: Algoritmo Apriori

Generación de candidatos en nivel 4

- Partimos de $C_3 = \{a1a2a3, a1a2a5\}$ el conjunto 2-itemset.
- Para todo $A, B \in C_3$ el conjunto estará formado por candidatos $A \cup B$.
- Posibles candidatos en C_4 serían $Candidatos = \{a1a2a3a5\}$.
- Utilizamos la propiedad esencial de **poda** del algoritmo **Apriori**:
- Tras la poda los conjuntos frecuentes de tamaño 4 (5-itemset) son $C_4 = \varnothing$.

Ejemplo: Algoritmo Apriori

Generación de candidatos en nivel 4

- Partimos de $C_3 = \{a1a2a3, a1a2a5\}$ el conjunto 2-itemset.
- Para todo $A, B \in C_3$ el conjunto estará formado por candidatos $A \cup B$.
- Posibles candidatos en C_4 serían $Candidatos = \{a1a2a3a5\}$.
- Utilizamos la propiedad esencial de **poda** del algoritmo **Apriori**:
  $a1a2a3a5 \notin C_4$ porque $a2a3a5$ no está en C_3 (no es frecuente).
- Tras la poda los conjuntos frecuentes de tamaño 4 (5-itemset) son $C_4 = \varnothing$.

Ejemplo: Algoritmo Apriori

Generación de candidatos en nivel 4

- Partimos de $C_3 = \{a1a2a3, a1a2a5\}$ el conjunto 2-itemset.
- Para todo $A, B \in C_3$ el conjunto estará formado por candidatos $A \cup B$.
- Posibles candidatos en C_4 serían $Candidatos = \{a1a2a3a5\}$.
- Utilizamos la propiedad esencial de **poda** del algoritmo **Apriori**:
  1. $a1a2a3a5 \notin C_4$ porque $a2a3a5$ no está en C_3, *noesfrecuente*.
- Tras la poda los conjuntos frecuentes de tamaño 4 (5-itemset) son $C_4 = \varnothing$.

Ejemplo: Algoritmo Apriori

Generación de candidatos en nivel 4

- Partimos de $C_3 = \{a1a2a3, a1a2a5\}$ el conjunto 2-itemset.
- Para todo $A, B \in C_3$ el conjunto estará formado por candidatos $A \cup B$.
- Posibles candidatos en C_4 serían $Candidatos = \{a1a2a3a5\}$.
- Utilizamos la propiedad esencial de **poda** del algoritmo **Apriori**:
  1. $a1a2a3a5 \notin C_4$ porque $a2a3a5$ no está en C_3, *noesfrecuente*.
- Tras la poda los conjuntos frecuentes de tamaño 4 (5-itemset) son $C_4 = \varnothing$.

Ejemplo: Algoritmo Apriori

Generación de candidatos en nivel 4

- Partimos de $C_3 = \{a1a2a3, a1a2a5\}$ el conjunto 2-itemset.
- Para todo $A, B \in C_3$ el conjunto estará formado por candidatos $A \cup B$.
- Posibles candidatos en C_4 serían $Candidatos = \{a1a2a3a5\}$.
- Utilizamos la propiedad esencial de **poda** del algoritmo **Apriori**:
  1. $a1a2a3a5 \notin C_4$ porque $a2a3a5$ no está en C_3, *noesfrecuente*.
- Tras la poda los conjuntos frecuentes de tamaño 4 (5-itemset) son $C_4 = \varnothing$.

Ejemplo: Algoritmo Apriori

Generación de candidatos en nivel 4

- Partimos de $C_3 = \{a1a2a3, a1a2a5\}$ el conjunto 2-itemset.
- Para todo $A, B \in C_3$ el conjunto estará formado por candidatos $A \cup B$.
- Posibles candidatos en C_4 serían $Candidatos = \{a1a2a3a5\}$.
- Utilizamos la propiedad esencial de **poda** del algoritmo **Apriori**:
  1. $a1a2a3a5 \notin C_4$ porque $a2a3a5$ no está en C_3, *noesfrecuente*.
- Tras la poda los conjuntos frecuentes de tamaño 4 (5-itemset) son $C_4 = \varnothing$.

Ejemplo: Algoritmo Apriori

Id	$a1$	$a2$	$a3$	$a4$	$a5$
id_1	1	1	0	0	1
id_2	0	1	0	1	0
id_3	0	1	1	0	0
id_4	1	1	0	1	0
id_5	1	0	1	0	0
id_6	0	1	1	0	0
id_7	1	0	1	0	0
id_8	1	1	1	0	1
id_9	1	1	1	0	0

Conjuntos frecuentes calculados

$C = \{a1, a2, a3, a4, a5, a1a2, a1a3, a1a5, a2a3, a2a4, a2a5, a1a2a3, a1a2a5\}$

Algoritmo Apriori en R - Conjuntos frecuentes

```
AprioriGenerateRules <- function (D,F,MIN_SUP,MIN_CONF)
     #create empty rule set
     r <- CreateRuleSets()
     len4f <- length(F)
     for(idx in 1:len4f){
          #add rule F[[idx]] => {}
          AddRule2RuleSets(r,F[[idx]],NULL)
          c <- list()
          c[[1]] <- CreateItemSets(F[[idx]])
          h <- list()
          k <-1
          while( !IsEmptyItemSets(c[[k]]) ){
               #get heads of confident association rule in c[[k]]
               h[[k]] <- getPrefixOfConfidentRules(c[[k]], F[[idx]],D,MIN_CONF)
               c[[k+1]] <- CreateItemSets()

               #get candidate heads
               len4hk <- length(h[[k]])
               for(jdx in 1:(len4hk-1)){
                    if( Match4Itemsets(h[[k]][jdx], h[[k]][jdx+1]) ){
                         tempItemset <- CreateItemset(h[[k]][jdx],h[[k]][jdx+1][k])
                         if( IsSubset2Itemsets(h[[k]], tempItemset) ){
                              Append2ItemSets(c[[k+1]],tempItemset)
                         }
                    }
               }
          }
          #append all new association rules to rule set
          AddRule2RuleSets(r,F[[idx]],h)
          }
     r
}
```

Ejemplo: Algoritmo Apriori

Definition

De un itemset $A = X \cup Y$ calculado con el método anterior, extraemos la regla $X \rightarrow Y$ si:

- Para todo $X \subseteq A$, calculamos $Conf(X \rightarrow Y) = \frac{Sop(X \cup Y)}{Sop(X)}$.
- Si Soporte supera el umbral se admite la regla de asociación.

Itemset A	X	Y	$Sop(X)$	$Sop(X \cup Y)$	$Conf(X \rightarrow Y)$	$Umbral = 0.7$
a1a2a5						
	a1a2	a5	4	2	0.5	✗
	a1a5	a2	2	2	1	✓ $a1a5 \rightarrow a2$
	a2a5	a1	2	2	1	✓ $a2a5 \rightarrow a1$
	a1	a2a5	6	2	0.33	✗
	a2	a1a5	7	2	0.29	✗
	a5	a1a2	2	2	1	✓ $a5 \rightarrow a1a2$

Ejemplo: Algoritmo Apriori

Id	$a1$	$a2$	$a3$	$a4$	$a5$
id_1	1	1	0	0	1
id_2	0	1	0	1	0
id_3	0	1	1	0	0
id_4	1	1	0	1	0
id_5	1	0	1	0	0
id_6	0	1	1	0	0
id_7	1	0	1	0	0
id_8	1	1	1	0	1
id_9	1	1	1	0	0

Conjuntos frecuentes y Reglas de asociación

$$C = \{a1, a2, a3, a4, a5, a1a2, a1a3, a1a5, a2a3, a2a4,\ a2a5, a1a2a3, a1a2a5\}$$

$$\Gamma = \{a1a5 \rightarrow a2, a2a5 \rightarrow a1, a1a5 \rightarrow a2, a5 \rightarrow a1a2, \ldots\}$$

Extracción de reglas

- El número de posibles reglas con d atributos es

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

- Con un soporte del 20% y confianza 50% se puede reducir en un 80% el número de reglas.
- Los algoritmos que mejoran el algoritmo Apriori pretenden ir a la tabla el menor número de veces, es decir, podar las reglas sin tener que calcular el soporte y la confianza.

Confianza

Definition (Soporte)

$$Sop(X) = \frac{|\ X\ |}{|\ D\ |}$$

$|\ X\ |$ número de filas con X y $|\ D\ |$ el número de filas de la tabla.

- El soporte denota la frecuencia de la regla en el dataset, $P(X \cup Y)$.
- Un alto valor indica que la regla es cierta en gran parte de la base de datos.
- Un bajo valor indica una regla poco frecuente. Podríamos eliminar las reglas con bajo soporte.

Definition (Confianza)

$$Conf(X \to Y) = \frac{Sop(X \cup Y)}{Sop(X)} = \frac{|\ X \cup Y\ |}{|\ X\ |}$$

- La confianza es un estimador de la $P(Y|X)$, es decir el porcentaje del dataset que conteniendo X también contiene a Y.
- Indicador de la fiabilidad de la regla.

Lift

La confianza de una regla puede ser muy alta pero el soporte del consecuente muy bajo.
En la definición de confianza no se tiene en cuenta cuantas veces aparece el itemset del consecuente en la tabla.
Son necesarias nuevas medidas

Definition (Lift)

$$Lift(X \to Y) = \frac{Sop(X \cup Y)}{Sop(X)Sop(Y)}$$

- X e Y estarán negativamente correlacionados si el valor es menor que 1.
- Problema: reglas que se cumplen casi al 100%: el 5% compra leche y el 90% compran cerveza, entonces $Lift(Leche \to Cerveza) = 1.11$.
- Es simétrica: $X \to Y, Y \to X$ tienen igual Lift.

Conv, Lev

Definition (Conviction)

$$Conv(X \rightarrow Y) = \frac{Sop(X)Sop(\overline{Y})}{Sop(X \cup \overline{Y})}$$

- *Conv* valdrá 1 si los items X e Y no están relacionados.

Definition (Leverage - Piatetsky-Shapiro)

$$Lev(X \rightarrow Y) = Sop(X \cup Y) - Sop(X)Sop(Y)$$

Ejercicio

Dada la siguiente tabla, aplica el algoritmo Apriori (a mano) con un soporte de 60% y confianza de 80%. Aplica todos los estimadores de la calidad de las reglas de asociación con alguna de las reglas que obtengas: Soporte, Confianza, Lift,...

Transaction	Items
1	Pan, Leche, Patatas, Mostaza
2	Cerveza, Pañales, Pan, Huevos
3	Cerveza,Cola,Pañales,Leche
4	Cerveza, Pan, Pañales,Leche, Patatas
5	Cola, Pan, Pañales, Leche
6	Cerveza, Pan, Pañales, Leche, Mostaza
7	Cola, Pan, Pañales, Leche