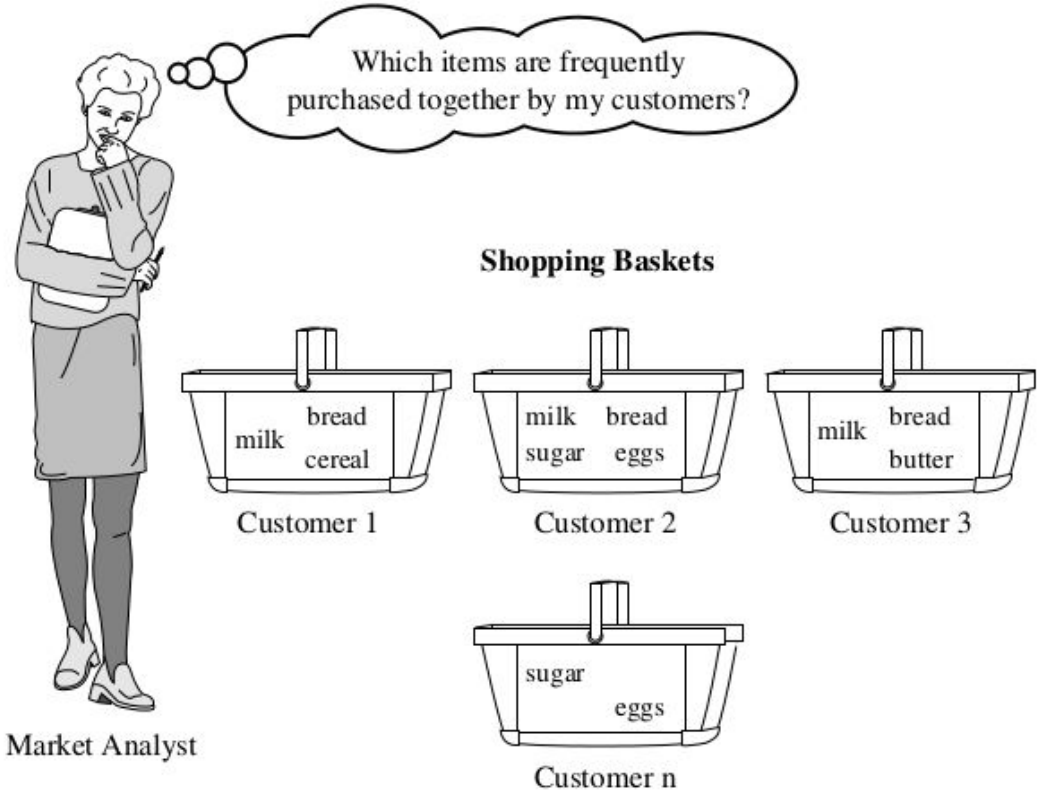
Data Mining

Reglas de Asociación

Temas

- Reglas de asociación
- Conceptos básicos:
  - itemset, itemset frecuente,
  - itemset máximo e itemset cerrado.
- Algoritmo Apriori.

Reglas de Asociación



Problemática

Dado un conjunto de transacciones, encontrar reglas que puedan predecir la ocurrencia de un ítem basado en la presencia de otros.

Análisis del “Carrito de compras”

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Ejemplo de Reglas de Asociación:

{Diaper} → {Beer}
{Milk, Bread} → {Eggs,Coke}
{Beer, Bread} → {Milk}

La implicación indica **“co-ocurrencia”** no causalidad

Itemsets

Itemset: Una colección de 1 o más ítems:

Ejemplo: `{Milk, Bread, Diaper}`

k-itemset: Un itemset que contiene k items

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

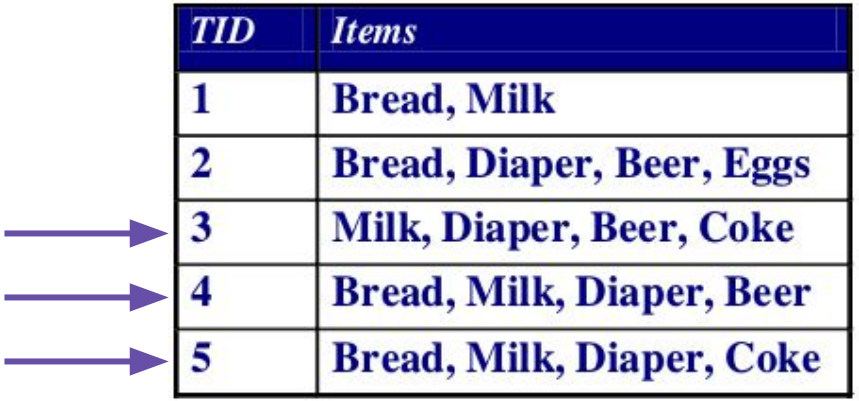
- **1-itemset**: {Bread}, {Beer}
- **2-itemset:** {Milk, Eggs}, {Diaper, Beer}
- **3-itemset**: {Diaper, Beer, Bread}

Support Count (σ)

Support count (σ): Cantidad de ocurrencias de un itemset.

Ejemplo:

```
σ({Milk, Diaper}) = 3
```

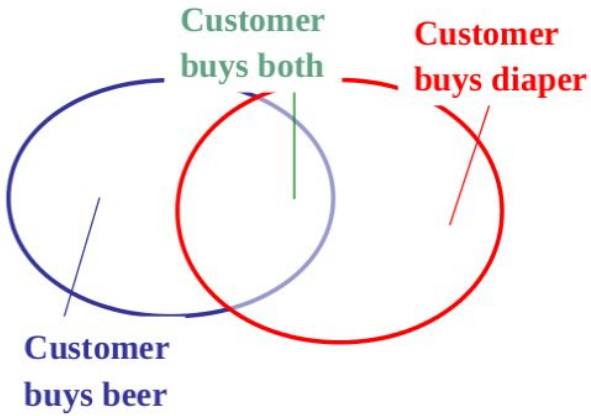


TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Support

Support (s): Fracción de transacciones que contiene a un itemset.

$$s(x) = \frac{\sigma(x)}{|T|}$$



Support

Support (s): Fracción de transacciones que contiene a un itemset.

$$s(x) = \frac{\sigma(x)}{|T|}$$

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Ejemplo:

s({Milk, Diaper}) = $\frac{3}{5}$

¿Cuál es el soporte de Beer?

Itemsets frecuentes

Itemset frecuente: Un itemset cuyo ***support*** es mayor o igual al **umbral** establecido como **mínimo soporte** (minsup).

Supongamos que **minsup = 0.6**

¿Es frecuente? `s({Milk, Diaper})` $= \frac{3}{5}$

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

¿Y el 1-itemset `{Beer}`?

¿Y `{Milk, Diaper, Beer}`?

¿Cuales son los **1-itemsets** infrecuentes?

Regla de Asociación

Regla de Asociación: Una expresión de la forma **X** → **Y**, donde X e Y son itemsets frecuentes.

Ejemplo:

{Milk, Diaper} → {Beer}

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Métricas de Evaluación de Reglas

- **Support (s)**: Fracción de transacciones que contiene a X e Y.
- **Confidence (c)**: Mide con qué frecuencia Y aparece en transacciones en las que también aparece X.

$$C(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

Ejemplo:

`{Milk, Diaper}` $\rightarrow$ `{Beer}`

$$S = \frac{\sigma(Milk, Diaper, Beer)}{|T|} = \frac{2}{5} = 0.4$$

$$C = \frac{\sigma(Milk, Diaper, Beer)}{\sigma(Milk, Diaper)} = \frac{2}{3} = 0.67$$

Ejemplos de Reglas de Asociación

Reglas	S	C
{Milk,Diaper} → {Beer}	0.4	0.67
{Milk,Beer} → {Diaper}	0.4	1.0
{Diaper,Beer} → {Milk}	0.4	0.67
{Beer} → {Milk,Diaper}	0.4	0.67
{Diaper} → {Milk,Beer}	0.4	0.5
{Milk} → {Diaper,Beer}	0.4	0.5

Reglas creadas a partir del itemset:

{Milk, Diaper, Beer}

- Tienen el mismo soporte.
- ¿Por qué tienen distinta confianza?

Descubrimiento de reglas

Dado un conjunto de transacciones **T**, el objetivo del descubrimiento de reglas de asociación es encontrar todas las reglas que cumplen:

- `Soporte ≥ minsup`
- `Confianza ≥ minconf`

Aproximación de fuerza bruta:

1. Listar todas las posibles reglas de asociación.
2. Calcular el soporte y la confianza para cada una.
3. Eliminar las que no satisfacen los umbrales predefinidos.

⇒ Computacionalmente Prohibitivo!

Problema de escala

- Un supermercado como **WalMart**
  - vende unos **100K** items y puede guardar millones de canastas!

- **La Web**
  - tiene más de **100M** de palabras y miles de millones de páginas.

¿Por qué es prohibitivo?

- Supongamos que tenemos itemset frecuentes de 100 items:

$$a_1 , a_2 , ..., a_{100}$$

- Vamos a tener para:
  - **1-itemset** frecuentes $\binom{100}{1} = 100$
  - **2-itemset** frecuentes $\binom{100}{2} = 4950$
  - **3-itemset** frecuentes $\binom{100}{3} =$ 

Association Rule Mining

ARM puede ser visto como un problema de 2 pasos:

1. Generación de los **itemsets frecuentes**
    - Generar todos los itemsets con `support ≥ minsup`
2. **Generación de reglas**
    - Generar a partir de la división de los itemsets frecuentes los subconjuntos con las reglas que satisfacen la confianza

> El primer paso es computacionalmente muy caro

Aplicaciones I

- **Encontrar conceptos relacionados**:
  - Supongamos que las palabras son los ítems y los “documentos” las canastas (los itemsets).
  - Podríamos buscar términos del problema que aparecen juntos con altos valores de soporte y confianza.

- **Plagio**: En este caso los ítems son los documentos y las canastas las oraciones.
  - Donde un “ítem/documento” está en una “canasta/oración” si la oración pertenece al documento.
  - Una o dos oraciones en común en distintos documentos son un buen indicador de plagio.

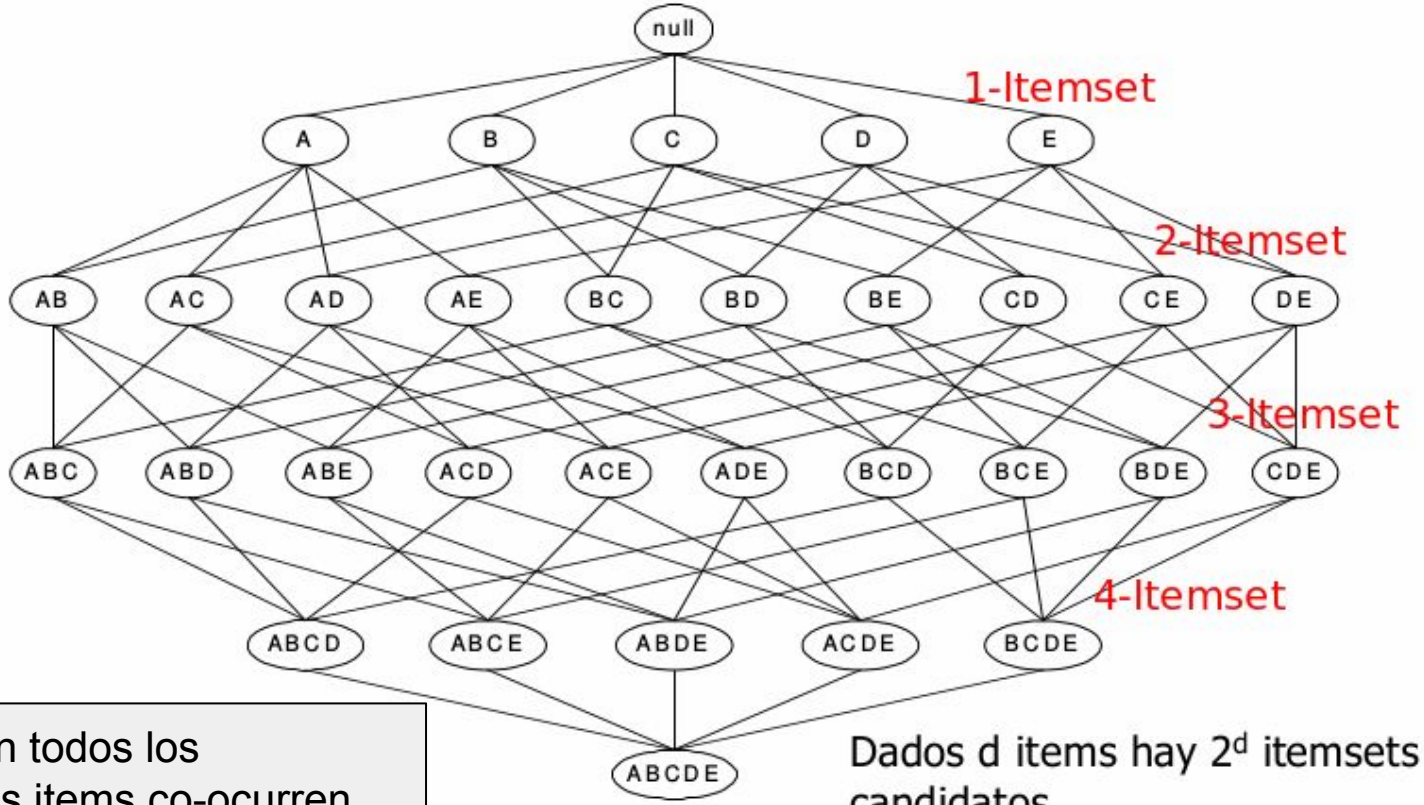
Association Rule Mining

ARM puede ser visto como un problema de 2 pasos:

1. Generación de los **itemsets frecuentes**
    - Generar todos los itemsets con `support ≥ minsup`
2. **Generación de reglas**
    - Generar a partir de la división de los itemsets frecuentes los subconjuntos con las reglas que satisfacen la confianza

> El primer paso es computacionalmente muy caro

Generación de Itemsets Frecuentes



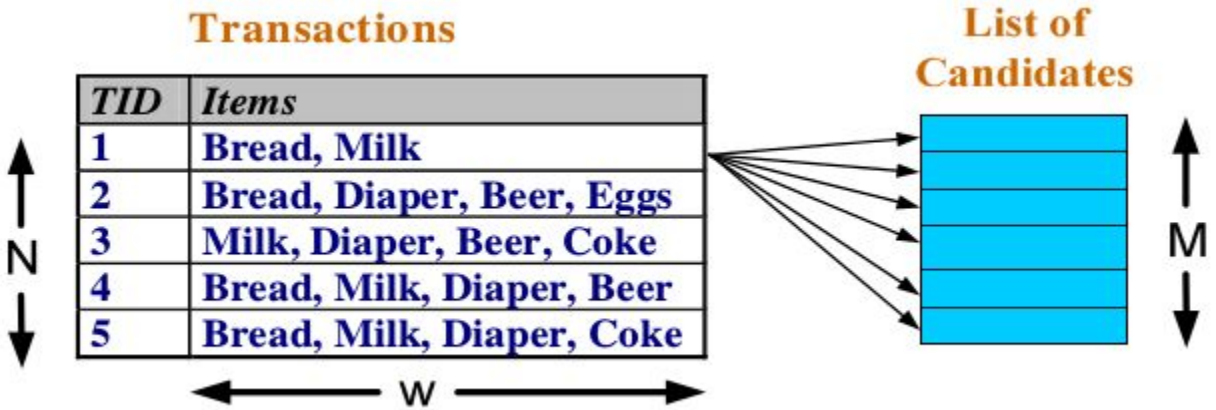
Los **candidatos** son todos los k-itemsets donde los items co-ocurren.

Dados d items hay 2^d itemsets candidatos

Generación de Itemsets Frecuentes

Aproximación de fuerza bruta:

- Cada itemset en el lattice es un candidato
- Contar el soporte de cada itemset barriendo las transacciones



- Comparar cada transacción contra cada itemset
- Complejidad ~ **O(NMw)** ⇒ Caro porque **M = 2^d !!!**

Estrategias para la generación de itemsets

Reducir el número de candidatos (**M**)

- Búsqueda completa: **M = 2^d**
- Utilice **técnicas de poda** para reducir **M**

Reducir el número de comparaciones (**NM**)

- Utilice las estructuras de datos eficientes para almacenar los candidatos y las transacciones.
- No hay necesidad de comparar cada candidato contra cada transacción.

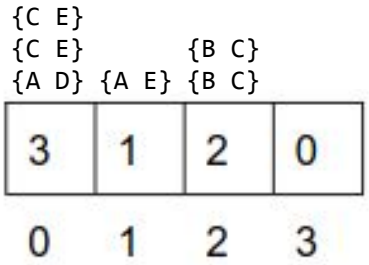
Reducir el número de transacciones (**N**)

- Reducir el tamaño de **N** como el incremento del tamaño de los itemsets.
- Esto es utilizando algoritmos como **Direct Hashing and Pruning** (DHP)

Utiliza una **función de hash** para filtrar la próxima generación candidatos

Database D

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E



Principio Apriori

Es una estrategia de reducción del número de candidatos.

Si un itemset es frecuente, entonces todos sus subsets deben además ser frecuentes.

El Principio Apriori se sostiene debido a las siguiente propiedades de la medida de support:

$$\forall\ X,Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

- El support de un itemset (Y) nunca excede el support de sus subsets (X).
- Esto es conocido como la propiedad de antimonotonía del *support*.

Anti-Monotonía

Si un itemset X no satisface el umbral de *min_support* entonces X no es frecuente.

Es decir:

$$S(X) < min\_support$$

Si agrego X_2 al itemset X, $(X \cup X_2)$ entonces el resultado del itemset no puede ser más frecuente que X.

$X \cup X_2$ es no frecuente, por lo tanto:

$$S(X \cup X_2) < min\_support$$

Algoritmo Apriori

El algoritmo Apriori fue propuesto por R. Agrawal and R. Srikant en 1994 para ***mining frecuent itemsets*** de reglas de asociación binarias (como los ejemplos de la canasta).

El algoritmo maneja 2 conjuntos de itemsets:

- Candidatos (C_k)
- Frecuentes (L_k)

Y se divide en dos pasos: uno de **Join** y otro de **Prune**

Algoritmo Apriori: Pseudo Código

Join Step: C_k es generado uniendo L_{k-1} con sigo mismo.
Prune Step: Un (k-1)-itemset que no es frecuente no puede ser un subset de un k-itemset frecuente.

Pseudo-code:

C_k: Candidate itemset of size k
L_k: frequent itemset of size k
L_1 = {frequent items};
for(k = 1; L k !=∅; k ++) **do begin**
  C_{k+1} = **candidates generated** from L_k
  for each transaction **t** in database **do**
    increment the count of all candidates
    in C_{k+1} that are contained in **t**
    L_{k+1} = candidates in C_{k+1} with *min_support*
  end
return U_k L_k

Generación de Candidatos: Ejemplo

Tenemos una lista de frecuentes:

L_3 = { abc, abd, acd, ace, bcd }

Join: $L_3 * L_3$

abcd de **abc** y **abd**

acde de **acd** y **ace**

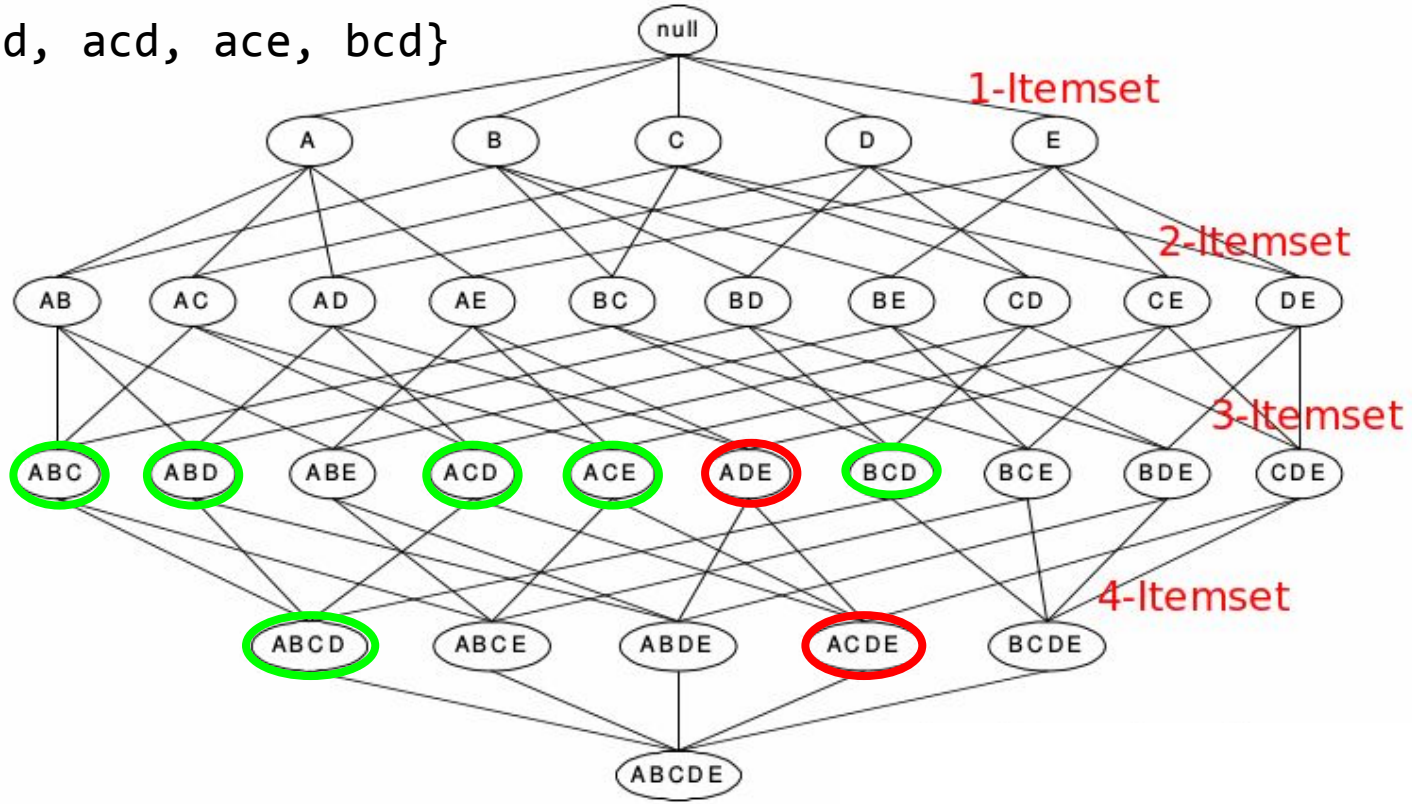
Pruning: Como **ade** no está en L_3 se elimina **acde**

C_4 ={ abcd }

Generación de Candidatos: Ejemplo

L_3 = {abc, abd, acd, ace, bcd}

abcd de **abc** y **abd**
acde de **acd** y **ace**



C_4 ={ abcd }

Obtención de las reglas

```
Para cada frequent itemset X ,
  Para cada subset A <> ∅ de X,
     Dado B = X - A
     A ⇒ B es una regla de asociación SI
        Confidence(A ⇒ B) ≥ minConf,

Donde:
   support(A ⇒ B) = support(AB) y
   confidence(A ⇒ B) = support(AB)/support(A)
```

Como las reglas se construyen a partir de los itemsets frecuentes, todas satisfacen el *min_support*.

Factores que afectan la complejidad

- Elegir el umbral de ***min_support***
  - Bajar el umbral de support resulta en más itemsets frecuentes
  - Esto puede incrementar el número de candidatos y la longitud máxima de itemsets frecuentes
- Dimensionalidad del dataset (cantidad de ítems)
  - Se necesita más espacio para almacenar el count support de cada uno de los items.
  - Si la cantidad de items frecuentes aumenta, tanto el costo computacional como las operaciones de I/O se incrementan.
- Tamaño de la base de datos (# de transacciones)
  - Dado que **Apriori hace varias pasadas**, el tiempo de ejecución del algoritmo va a aumentar con el número de transacciones

Bibliografía

- Jiawei Han, Micheline Kamber, Jian Pei. 2012. Tercera edición. Data Mining: Concepts and Techniques. Cap. 2 y Cap. 3
- Daniel T. Larose. 2014. Segunda edición. Discovering Knowledge in Data: An Introduction to Data Mining.