

# UNIDAD 2: MINERÍA DE ELEMENTOS FRECUENTES

## ALGORITMO APRIORI

---

Blanca Vázquez y Gibran Fuentes-Pineda

25 de agosto de 2021

1. Se encuentran los conjuntos de elementos frecuentes con soporte mínimo  $\mathcal{F}$
2. Se generan las reglas de asociación con una confianza mínima a partir de  $\mathcal{F}$

- Fuerza bruta
  1. Generar todas las combinaciones de elementos posibles (*candidatos*)
  2. Contar el soporte de cada uno verificando si es un subconjunto de cada transacción  $T_i \in \mathcal{T}$
- Si existen  $d$  elementos distintos en las transacciones, serían  $2^d - 1$  combinaciones
  - Cuando  $d$  se vuelve grande, esta estrategia no es práctica
  - Por ej. si  $d = 500$ , se necesitarían generar  $2^{500} \gg 10^{80}$

- Procedimiento general
  1. Generar candidatos atravesando el espacio de búsqueda (red o *lattice*)
  2. Calcular el soporte de los candidatos en la base de datos
- Optimizaciones
  - Reducción del espacio de búsqueda podando candidatos
  - Conteo eficiente podando transacciones
  - Uso de estructuras de datos compactas para representar candidatos y transacciones

- El soporte de todos los subconjuntos de  $I$  es igual o mayor al de  $I$ , esto es,

$$\text{sup}(J) \geq \text{sup}(I), \forall J \subseteq I$$

- A esta propiedad se le conoce como *monotonicidad del soporte*

- La propiedad de monotonidad tiene como consecuencia que todos los subconjuntos de un conjunto frecuente sean también frecuentes
- A esta propiedad se le conoce como *cerradura hacia abajo*
- Se aprovecha en distintos algoritmos para hacer la búsqueda más eficiente

- Un conjunto de elementos frecuentes es máximo si ningún superconjunto de este es frecuente
- Todos los conjuntos de elementos frecuentes se pueden derivar de los conjuntos máximos
- Los conjuntos frecuentes máximos son una forma resumida de todos los conjuntos frecuentes, aunque sin la información del soporte de sus subconjuntos

## CONJUNTOS FRECUENTES MÁXIMOS: EJEMPLO

- Ejemplo: ¿Cuáles son los conjuntos frecuentes máximos con soporte mínimo de 0.3 de las siguientes transacciones?

ID	Transacción
1	{Pan, Mantequilla, Leche}
2	{Huevo, Leche, Yogurt}
3	{Pan, Queso, Huevo, Leche}
4	{Huevo, Leche, Yogurt}
5	{Queso, Leche, Yogurt}



## ACELERANDO LA BÚSQUEDA

- Ningún conjunto de  $k + 1$  elementos es frecuente si ninguno de sus subconjuntos lo es
  - Es posible generar candidatos de forma incremental:  
 $k = 1, k = 2, \dots, k = d$
- En muchas bases de datos el número de elementos máximo  $m$  en cualquier transacción es mucho menor que  $d$ , por lo que el número de candidatos sería

$$\sum_{i=1}^m \binom{d}{i} \ll 2^d$$

- Con esta estrategia, si tenemos  $d = 1000$  y  $m = 20$

$$\sum_{i=1}^{20} \binom{1000}{i} \approx 3.465404 \times 10^{41}$$

## ALGORITMO APRIORI (1)

1. Empieza contando el soporte de los candidatos de tamaño  $k = 1$  (elementos individuales)  $\mathcal{C}_1$
2. Genera candidatos  $\mathcal{C}_{k+1}$  de tamaño  $k + 1$  a partir de los candidatos frecuentes  $\mathcal{F}_k$  de tamaño  $k$
3. Cuenta el soporte de estos candidatos  $\mathcal{C}_{k+1}$ , guardando solo aquellos que sean frecuentes  $\mathcal{F}_{k+1}$
4. Repite 2-3

## ALGORITMO APRIORI (2)

**función** APRIORI( $\mathcal{T}, \text{minsup}$ )

$\mathcal{F}_1 \leftarrow$  elementos con soporte  $\geq \text{minsup}$

$k \leftarrow 2$

**mientras**  $\mathcal{F}_k \neq \emptyset$  **hacer**

$\mathcal{C}_{k+1} \leftarrow \{C = A \cup \{b\} \mid A \in \mathcal{F}_k \wedge b \notin A, \{S \subseteq C \mid |S| = k\} \in \mathcal{F}_k\}$

**para** transacciones  $T \in \mathcal{T}$  **hacer**

**para** candidatos  $C \in \mathcal{C}_{k+1}$  **hacer**SC

**si**  $C \subseteq T$  **entonces**

$\text{cuenta}[C] \leftarrow \text{cuenta}[C] + 1$

**fin de si**

**fin de para**

**fin de para**

$\mathcal{F}_{k+1} \leftarrow \{C \in \mathcal{C}_{k+1} \mid \text{cuenta}[C] \geq \text{minsup}\}$

$k \leftarrow k + 1$

**fin de mientras**

$\bigcup_{i=1}^k \mathcal{F}_i$

**fin de función**

## EJEMPLO

- Encontrar conjuntos de elementos con soporte mínimo de 0.4

ID	Transacción
1	{1, 2, 3, 4}
2	{1, 2, 4}
3	{1, 2}
4	{2, 3, 4}
5	{2, 3}
6	{3, 4}
7	{2, 4}

## EJEMPLO

- Búsqueda de conjuntos con soporte mínimo de 0.4
  - Candidatos de tamaño  $k = 1$  (elementos individuales):  
 $\mathcal{C}_1 = \{\{1\}, \{2\}, \{3\}, \{4\}\}$
  - Conjuntos frecuentes:  $\mathcal{F}_1 = \{\{1\}, \{2\}, \{3\}, \{4\}\}$

ID	Transacción
1	{1, 2, 3, 4}
2	{1, 2, 4}
3	{1, 2}
4	{2, 3, 4}
5	{2, 3}
6	{3, 4}
7	{2, 4}

## EJEMPLO

- Búsqueda de conjuntos con soporte mínimo de 0.4
  - Candidatos de tamaño  $k = 2$ :  
 $\mathcal{C}_1 = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$
  - Conjuntos frecuentes:  
 $\mathcal{F}_1 = \{\{1\}, \{2\}, \{3\}, \{4\}\}, \mathcal{F}_2 = \{\{1, 2\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$

ID	Transacción
1	{1, 2, 3, 4}
2	{1, 2, 4}
3	{1, 2}
4	{2, 3, 4}
5	{2, 3}
6	{3, 4}
7	{2, 4}

## EJEMPLO

- Búsqueda de conjuntos con soporte mínimo de 0.4
  - Candidatos de tamaño  $k = 3$ :  $\mathcal{C}_3 = \{2, 3, 4\}$
  - Conjuntos frecuentes:  $\mathcal{F}_1 = \{\{1\}, \{2\}, \{3\}, \{4\}\}$ ,  
 $\mathcal{F}_2 = \{\{1, 2\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$ ,  $\mathcal{F}_3 = \{\}$

ID	Transacción
1	$\{1, 2, 3, 4\}$
2	$\{1, 2, 4\}$
3	$\{1, 2\}$
4	$\{2, 3, 4\}$
5	$\{2, 3\}$
6	$\{3, 4\}$
7	$\{2, 4\}$

- Dados los conjuntos de elementos frecuentes con soporte mínimo  $\mathcal{F}$ 
  1. Se generan todos los pares  $\{X, Y\}, Y = I - X$  de cada conjunto  $I \in \mathcal{F}$
  2. Se calculan las confianzas de las reglas correspondientes a  $X \implies Y$
- Se mantienen únicamente las reglas con una confianza mínima



- Para cualquier regla de asociación

$$\text{conf}(X_2 \implies I - X_2) \geq \text{conf}(X_1 \implies I - X_1)$$

donde  $X_1, X_2$  e  $I$  son conjuntos frecuentes tales que  $X_1 \subset X_2 \subset I$ .

- A esta propiedad se le conoce como *monotonidad de la confianza* y nos permite descartar reglas de asociación redundantes.
  - $\{Pan\} \implies \{Leche, Huevo\}$  y  $\{Pan, Leche\} \implies \{Huevo\}$

- Para contar las ocurrencias de cada posible par de  $d$  elementos, se requerirían  $\binom{d}{2}$  enteros

## ESTRUCTURAS DE DATOS PARA CONTADORES

- Para contar las ocurrencias de cada posible par de  $d$  elementos, se requerirían  $\binom{d}{2}$  enteros
  - Si cada contador es de 4 bytes, ¿cuánta memoria es necesaria para todos los pares de 100,000 elementos?

## ESTRUCTURAS DE DATOS PARA CONTADORES

- Para contar las ocurrencias de cada posible par de  $d$  elementos, se requerirían  $\binom{d}{2}$  enteros
  - Si cada contador es de 4 bytes, ¿cuánta memoria es necesaria para todos los pares de 100,000 elementos?
    - $19999800000 \times 4 \approx 20GB$

## ESTRUCTURAS DE DATOS PARA CONTADORES

- Para contar las ocurrencias de cada posible par de  $d$  elementos, se requerirían  $\binom{d}{2}$  enteros
  - Si cada contador es de 4 bytes, ¿cuánta memoria es necesaria para todos los pares de 100,000 elementos?
    - $19999800000 \times 4 \approx 20GB$
- ¿Qué estructura de datos sería conveniente para los contadores de todos los pares  $\{i, j\}$ ?
  - Matriz triangular: se ordenan los pares tal que  $1 \leq i < j \leq d$ . El contador para  $\{i, j\}$  se almacena en  $a[k]$

$$k = (i - 1)(d - \frac{i}{2}) + j - i$$

- Tabla de dispersión con  $i$  y  $j$  como llave y la ocurrencia como valor