Unidad 2: Minería de elementos frecuentes

Algoritmos de árboles de enumeración

Blanca Vázquez y Gibran Fuentes-Pineda

Octubre 2020

Algoritmos de minería de elementos frecuentes

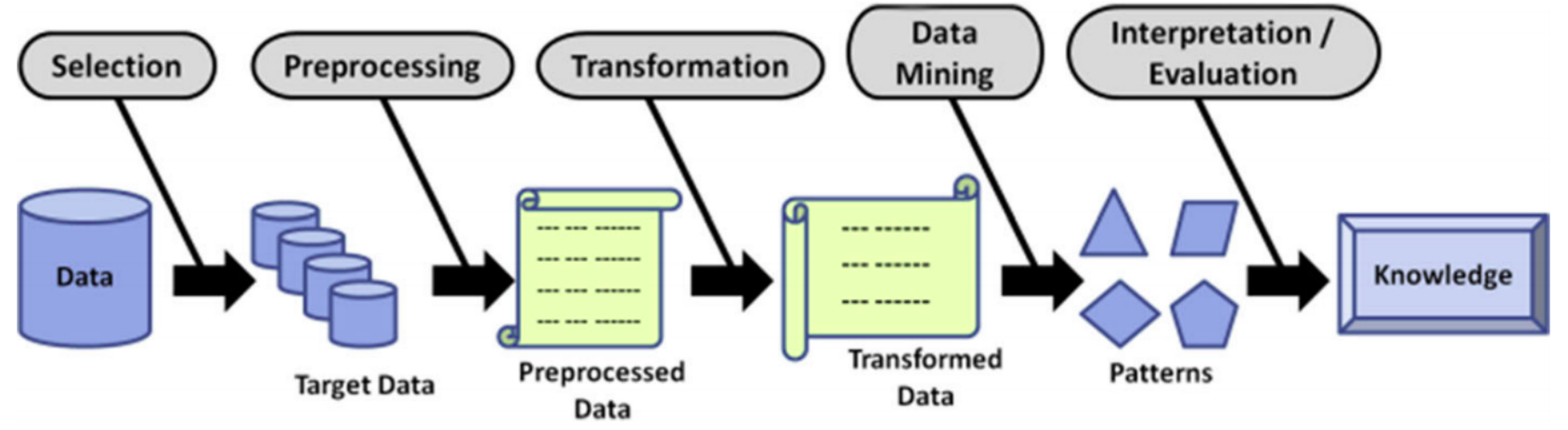


Imagen tomada de Chin-Hoong Chee [2019].

Algoritmos de árboles de enumeración

- Es un algoritmo de minería de elementos frecuentes
- Los conjuntos de elementos candidatos se generan en una estructura de árbol, conocida como árboles de enumeración.
- El crecimiento del árbol puede compensar diferentes estrategias entre almacenamiento, acceso a disco y eficiencia computacional.

Características

- Los patrones candidatos son generados a medida que crece el árbol lexicográfico.
- Proporciona una representación abstracta jerárquica de los conjuntos de elementos.
- Esta representación, es aprovechada para explorar de forma sistemática los patrones candidatos (evitando repeticiones).
- La salida del algoritmo son los conjuntos de elementos frecuentes definidos en la estructura de árbol.

Definición

El algoritmo de árbol de enumeración se define sobre los conjuntos de elementos frecuentes de la siguiente manera:

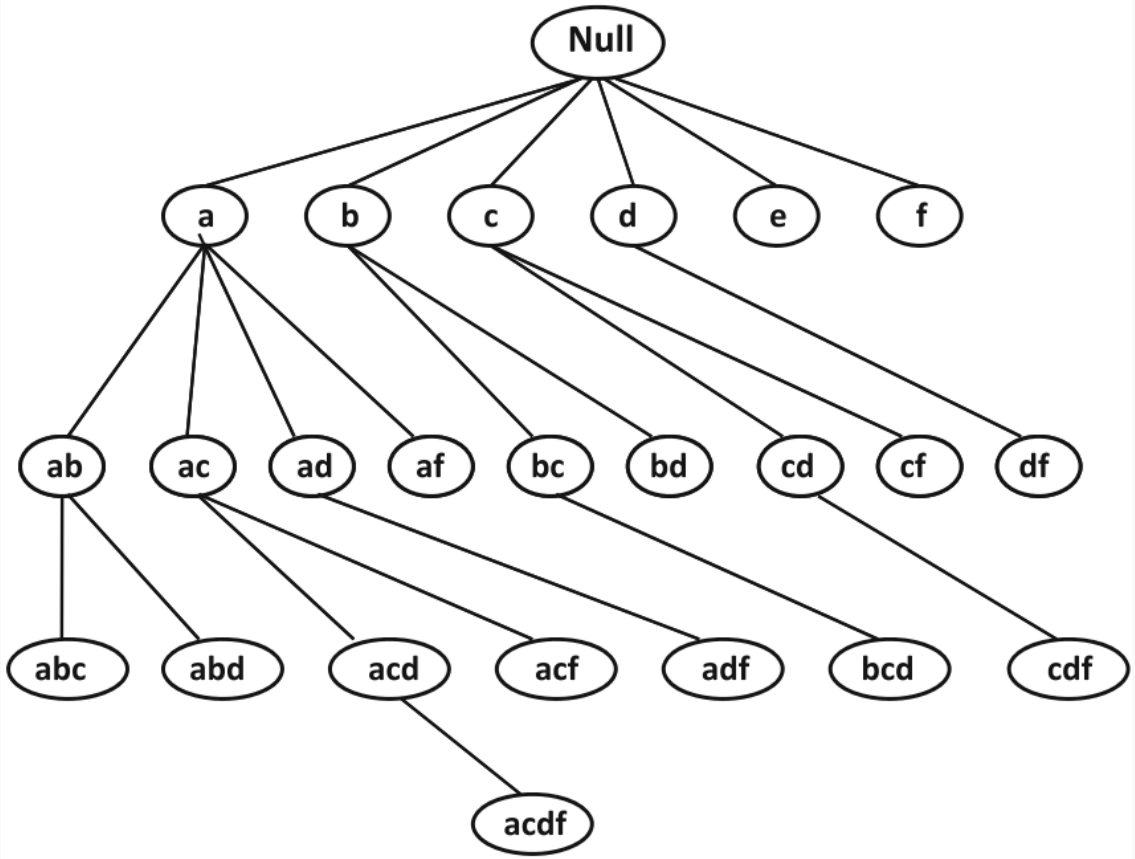
- Existe un nodo en el árbol correspondiente a cada conjunto de elementos frecuentes.
- La raíz del árbol corresponde al conjunto de elementos vacío.

Definición

El algoritmo de árbol de enumeración se define sobre los conjuntos de elementos frecuentes de la siguiente manera:

- Sea $I = \{i_1, ..., i_k\}$ un conjunto de elementos frecuentes, donde $i_1, i_2 ..., i_k$ son listados en orden lexicográfico
  - El padre del nodo I es el conjunto $\{i_1, ..., i_{k-1}\}$, por lo tanto el hijo de un nodo solamente se puede extender con elementos que aparecen lexicográficamente después de todos los elementos que ocurren en ese nodo.

Ejemplo de algoritmo de árboles de enumeración



A esta estructura, también se le conoce como: árbol lexicográfico ($a < b, b < c...$)

¿Cómo crecen los árboles de enumeración?

Dependerá del algoritmo selección.

- Los algoritmos indicarán el orden y la estructura para el descubrimiento de conjuntos de elementos frecuentes
- Una extensión frecuente del árbol es: un elemento que se usa para extender un nodo.

De forma general:

- El nodo de un árbol se extiende al encontrar los elementos frecuentes en 1 (nodos candidatos).
- Los nodos candidatos, son extendidos al encontrar los elementos frecuentes en 1
- y así sucesivamente.

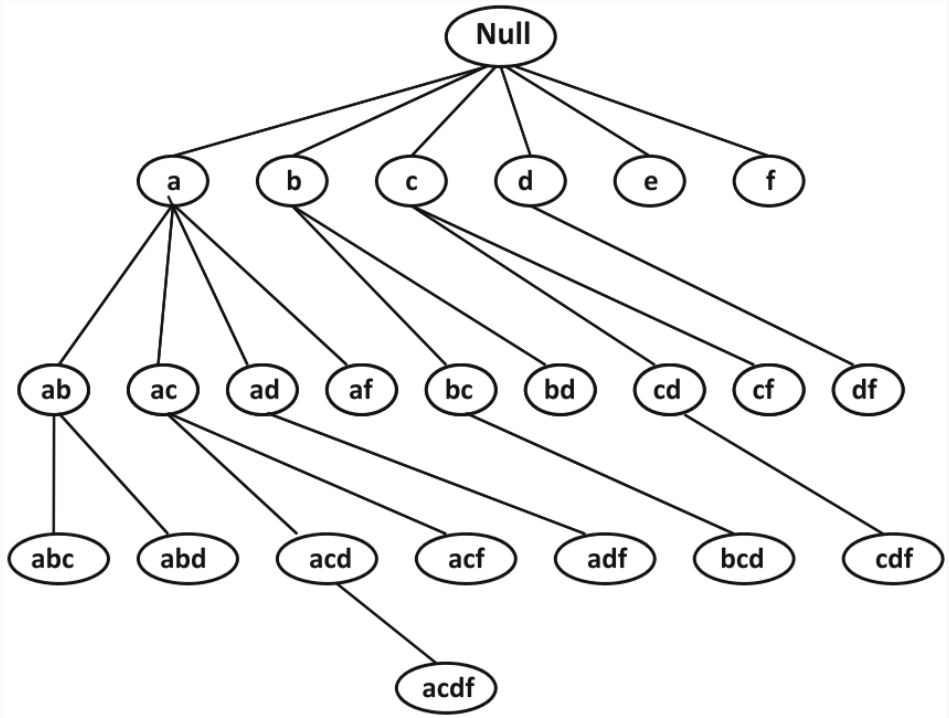
Algoritmo de árbol de enumeración genérico (parte 1)

- Seleccionamos uno o más nodo P en ET
- Determinamos las extensiones candidatas $C(P)$ para cada nodo P
- Calculamos el soporte de los candidatos
- Añadimos los elementos frecuentes a ET

Algoritmo de árbol de enumeración genérico (parte 2)

función GET($\mathcal{T}$, *minsup*)
    inicializamos el *ET* con un nodo *nulo*
    mientras cualquier nodo en *ET* no haya sido examinado **hacer**
        selecciona uno o más nodos P no examinados del *ET*
        genera extensiones candidatas $C(P)$ para cada nodo P
        determina las extensiones frecuentes $F(P) \subseteq C(P)$ para cada P con soporte
        extiende cada nodo P en *ET* con sus extensiones frecuentes en $F(P)$
    fin de mientras
    devolver *ET*
fin de función

Ejemplo de un algoritmo de árboles de enumeración



Descubrimiento de patrones

- El descubrimiento de patrones de manera temprana es útil para la eficiencia computacional en la minería de patrones.
- También la estrategia de primero profundidad ayuda en la gestión de la memoria de algoritmos basados en proyecciones.

En cuanto a las estrategias de conteo, estas tienden a ser costosas, por lo tanto, las estrategias de crecimiento tratan de optimizar el conteo mientras recorren el árbol.

Método recursivo de crecimiento de patrones basados en sufijos

- Es un caso especial del algoritmo de árboles de enumeración
- Un ejemplo, son los árboles de patrones frecuentes

Árboles de patrones frecuentes

- El árbol de patrones frecuentes (*Frequent-Pattern tree, FP-tree*) se basa en la metodología *divide y vencerás*
- Evita generar conjuntos de elementos candidatos para maximizar el uso de recursos.
- Busca comprimir una base de datos hacia un FP-tree

Algoritmo FP-tree

1. Escanear la base de datos y calcular la frecuencia de cada elemento
2. Ordenar los elementos por frecuencia (orden descendente) y que cumplan con un *minSup*
3. Por cada transacción, se construye una rama en el *FP-tree* . Si existe una ruta con prefijo común:
   3.1 Incrementar la frecuencia de los nodos en esa ruta y agregar el sufijo
   3.2 En caso contrario, crear una nueva rama

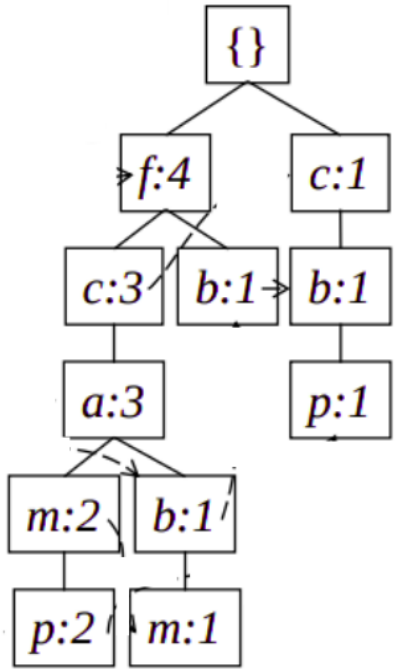
Ejemplo de FP-tree

TID	Elementos
1	{f, a, c, d, g, i, m, p}
2	{a, b, c, f, l, m, o}
3	{b, f, h, j, o}
4	{b, c, k, s, p}
5	{a, f, c, e, l, p, m, n}

Ejemplo de FP-tree

TID	Elementos	Elementos frecuentes
1	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
2	{a, b, c, f, l, m, o}	{f, c, a, b, m}
3	{b, f, h, j, o}	{f, b}
4	{b, c, k, s, p}	{c, b, p}
5	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

Elemento	Frecuencia
f	4
c	4
a	3
b	3
m	3
p	3



$$minSup = 0.5$$

Ejercicio

A partir de la BD de transacciones, generar el FP-tree ($minSup = 0.3$)

TID	Elementos
1	{leche,pan,mantequilla, pollo, huevos}
2	{huevos, leche, tocino, yogurt}
3	{pan, aguacates, cebolla, cereal}
4	{leche, yogurt, huevos, azúcar}
5	{huevos, aguacates, jamón, té, cereal, pan}
6	{pan, mantequilla, camarones, chocolate, leche}
7	{pasta, leche, pollo, yogurt, huevos}
8	{pan, jamón, cereal, leche, pollo}
9	{camarones, fresas, té, almendras, cereal, leche}
10	{mantequilla, leche, yogurt, jugo, pollo, huevos}

Ventajas del FP-Tree

- Ignora elementos no frecuentes
- orden descendente de frecuencia: es más probable que se compartan los elementos más frecuentes
- El FP-tree nunca será más grande que la BD original
- FP-Tree es frecuentemente usado en la minería de patrones frecuentes (divide y vencerás)