Fundamentals of Data Mining Algorithms
Itemset Mining (Chapter 10)

Loïc Cerf
September, 12th 2011
UFMG – ICEx – DCC
hello, world

I am:

- Loïc Cerf;
- Post-doc;
- French (#2);
- Still learning Portuguese;
- Your teacher for this course, the next one and some others.
Two applicative problems

Log analysis

A log contains Web pages accessed by every visitor during a session. What are the pages that are frequently loaded during a same session?
Two applicative problems

**Log analysis**

A log contains Web pages accessed by every visitor during a session. What are the pages that are frequently loaded during a same session?

**Basket analysis**

A shop knows what products were bought together. Which ones are frequently bought together?
Correlation

Frequent itemset mining can be seen as discovering correlations between an arbitrary number of Boolean attributes.
Frequent itemset mining can be seen as discovering correlations between an arbitrary number of Boolean attributes.
Outline

1. Frequent Itemset
2. Frequent Itemset Mining
3. Naive Extraction
4. APriori
5. Eclat
6. FP-Growth
7. Conclusion
Frequent Itemset

Outline

1. Frequent Itemset
2. Frequent Itemset Mining
3. Naive Extraction
4. APriori
5. Eclat
6. FPgrowth
7. Conclusion
Frequent Itemset

Itemset: definition

**Definition**

Given a set of attributes $\mathcal{A}$, an itemset $X$ is a subset of attributes, i.e., $X \subseteq \mathcal{A}$.

**Input:**

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>...</th>
<th>$a_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1$</td>
<td>$d_{1,1}$</td>
<td>$d_{1,2}$</td>
<td>...</td>
<td>$d_{1,n}$</td>
</tr>
<tr>
<td>$o_2$</td>
<td>$d_{2,1}$</td>
<td>$d_{2,2}$</td>
<td>...</td>
<td>$d_{2,n}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$o_m$</td>
<td>$d_{m,1}$</td>
<td>$d_{m,2}$</td>
<td>...</td>
<td>$d_{m,n}$</td>
</tr>
</tbody>
</table>

where $d_{i,j} \in \{\text{true, false}\}$

**Question**

How many itemsets are there?
Frequent Itemset

Itemset: definition

**Definition**

Given a set of attributes $\mathcal{A}$, an *itemset* $X$ is a subset of attributes, i.e., $X \subseteq \mathcal{A}$.

**Input:**

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>...</th>
<th>$a_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1$</td>
<td>$d_{1,1}$</td>
<td>$d_{1,2}$</td>
<td>...</td>
<td>$d_{1,n}$</td>
</tr>
<tr>
<td>$o_2$</td>
<td>$d_{2,1}$</td>
<td>$d_{2,2}$</td>
<td>...</td>
<td>$d_{2,n}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$o_m$</td>
<td>$d_{m,1}$</td>
<td>$d_{m,2}$</td>
<td>...</td>
<td>$d_{m,n}$</td>
</tr>
</tbody>
</table>

where $d_{i,j} \in \{\text{true}, \text{false}\}$

**Question**

How many itemsets are there? $2^{\left|\mathcal{A}\right|}$. 

---

Loïc Cerf

Fundamentals of Data Mining Algorithms
Frequent Itemset

Frequency: definition

**Definition (absolute frequency)**

Given the objects in $\mathcal{O}$ described with the Boolean attributes in $\mathcal{A}$, the absolute *frequency* of an itemset $X \subseteq \mathcal{A}$ in the dataset $\mathcal{D} \subseteq \mathcal{O} \times \mathcal{A}$ is $|\{o \in \mathcal{O} \mid \{o\} \times X \subseteq \mathcal{D}\}|$. 

The relative frequency is a joint probability.
Frequent Itemset

**Frequency: definition**

**Definition (absolute frequency)**

Given the objects in $\mathcal{O}$ described with the Boolean attributes in $\mathcal{A}$, the absolute frequency of an itemset $X \subseteq \mathcal{A}$ in the dataset $\mathcal{D} \subseteq \mathcal{O} \times \mathcal{A}$ is $|\{o \in \mathcal{O} \mid \{o\} \times X \subseteq \mathcal{D}\}|$.

**Definition (relative frequency)**

Given the objects in $\mathcal{O}$ described with the Boolean attributes in $\mathcal{A}$, the relative frequency of an itemset $X \subseteq \mathcal{A}$ in the dataset $\mathcal{D} \subseteq \mathcal{O} \times \mathcal{A}$ is $\frac{|\{o \in \mathcal{O} \mid \{o\} \times X \subseteq \mathcal{D}\}|}{|\mathcal{O}|}$.
**Frequent Itemset**

**Frequency: definition**

**Definition (absolute frequency)**

Given the objects in $\mathcal{O}$ described with the Boolean attributes in $\mathcal{A}$, the absolute *frequency* of an itemset $X \subseteq \mathcal{A}$ in the dataset $\mathcal{D} \subseteq \mathcal{O} \times \mathcal{A}$ is $|\{o \in \mathcal{O} | \{o\} \times X \subseteq \mathcal{D}\}|$.

**Definition (relative frequency)**

Given the objects in $\mathcal{O}$ described with the Boolean attributes in $\mathcal{A}$, the relative *frequency* of an itemset $X \subseteq \mathcal{A}$ in the dataset $\mathcal{D} \subseteq \mathcal{O} \times \mathcal{A}$ is $\frac{|\{o \in \mathcal{O} | \{o\} \times X \subseteq \mathcal{D}\}|}{|\mathcal{O}|}$.

The relative frequency is a joint probability.
Frequent Itemset Mining

Outline

1. Frequent Itemset
2. Frequent Itemset Mining
3. Naive Extraction
4. APriori
5. Eclat
6. FP Growth
7. Conclusion
Frequent Itemset Mining

**Definition**

Given the objects in \( \mathcal{O} \) described with the Boolean attributes in \( \mathcal{A} \), listing every itemset having a frequency above a given threshold \( \mu \in \mathbb{N} \).

**Input:**

\[
\begin{array}{c|cccc}
  & a_1 & a_2 & \ldots & a_n \\
\hline
o_1 & d_{1,1} & d_{1,2} & \ldots & d_{1,n} \\
o_2 & d_{2,1} & d_{2,2} & \ldots & d_{2,n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
o_m & d_{m,1} & d_{m,2} & \ldots & d_{m,n} \\
\end{array}
\]

where \( d_{i,j} \in \{\text{true, false}\} \) and a minimal frequency \( \mu \in \mathbb{N} \).
Definition

Given the objects in \( \mathcal{O} \) described with the Boolean attributes in \( \mathcal{A} \), listing every itemset having a frequency above a given threshold \( \mu \in \mathbb{N} \).

Output: every \( X \subseteq \mathcal{A} \) such that there are at least \( \mu \) objects having all attributes in \( X \).
Specifying a minimal absolute frequency $\mu = 2$ objects (or, equivalently, a minimal relative frequency of 50%).

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$o_2$</td>
<td>$\times$</td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>$o_3$</td>
<td></td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>$o_4$</td>
<td></td>
<td></td>
<td>$\times$</td>
</tr>
</tbody>
</table>
Frequent Itemset Mining

Frequent itemset mining: illustration

Specifying a minimal absolute frequency $\mu = 2$ objects (or, equivalently, a minimal relative frequency of 50%).

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$o_2$</td>
<td>$\times$</td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>$o_3$</td>
<td></td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>$o_4$</td>
<td></td>
<td></td>
<td>$\times$</td>
</tr>
</tbody>
</table>

The frequent itemsets are: $\emptyset$ (4), $\{a_1\}$ (2), $\{a_2\}$ (3), $\{a_3\}$ (2) and $\{a_1, a_2\}$ (2).
Frequent Itemset Mining

Completeness

Both the clustering and the classification schemes *globally* model the data: every object influences the output. That is the fundamental reason for these tasks to be solved in an *approximate* way.
Completeness

Both the clustering and the classification schemes *globally* model the data: every object influences the output. That is the fundamental reason for these tasks to be solved in an *approximate* way.

In contrast, *local* patterns, such as itemsets, describe “anomalies” in the data and all such anomalies usually can be *completely* listed.
Querying data:

\[ \{ d \in D \mid q(d, D) \} \]

where:

- \( D \) is a dataset (tuples),
- \( q \) is a query.
Frequent Itemset Mining

Inductive database vision

Querying patterns:

\[ \{ X \in P \mid Q(X, D) \} \]

where:

- \( D \) is the dataset,
- \( P \) is the pattern space,
- \( Q \) is an inductive query.
Querying the frequent itemsets:

\[
\{ X \in P \mid Q(X, D) \}\]

where:

- \( D \) is the dataset,
- \( P \) is the pattern space,
- \( Q \) is an inductive query.
Frequent Itemset Mining

Inductive database vision

Querying the frequent itemsets:

$$\{ X \in P \mid Q(X, D) \}$$

where:

- $D$ is a subset of $O \times A$, i.e., objects described with Boolean attributes,
- $P$ is the pattern space,
- $Q$ is an inductive query.
Querying the frequent itemsets:

\[ \{ X \in P \mid Q(X, D) \} \]

where:

- \( D \) is a subset of \( O \times A \), i.e., objects described with Boolean attributes,
- \( P \) is \( 2^A \),
- \( Q \) is an inductive query.
Querying the frequent itemsets:

\[ \{ X \in P \mid Q(X, D) \} \]

where:

- \( D \) is a subset of \( O \times A \), i.e., objects described with Boolean attributes,
- \( P \) is \( 2^A \),
- \( Q \) is \((X, D) \mapsto |\{o \in O \mid \{o\} \times X \subseteq D\}| \geq \mu\).
Querying the frequent itemsets:

\[ \{ X \in P \mid Q(X, D) \} \]

where:

- \( D \) is a subset of \( O \times A \), i.e., objects described with Boolean attributes,
- \( P \) is \( 2^A \),
- \( Q \) is \( (X, D) \mapsto f(X, D) \geq \mu \).
Frequent Itemset Mining

Inductive database vision

Querying the frequent itemsets:

\[ \{ X \in P \mid Q(X, D) \} \]

where:

- \( D \) is a subset of \( \mathcal{O} \times \mathcal{A} \), i.e., objects described with Boolean attributes,
- \( P \) is \( 2^\mathcal{A} \),
- \( Q \) is \( (X, D) \mapsto f(X, D) \geq \mu \).

Listing the frequent itemsets is NP-hard.
Outline

1. Frequent Itemset
2. Frequent Itemset Mining
3. Naive Extraction
4. APriori
5. Eclat
6. FP Growth
7. Conclusion
Naive Extraction

Naive algorithm

Input: $\mathcal{O}, \mathcal{A}, \mathcal{D} \subseteq \mathcal{O} \times \mathcal{A}, \mu \in \mathbb{N}$

Output: $\{X \subseteq \mathcal{A} \mid f(X, \mathcal{D}) \geq \mu\}$

for all $X \subseteq \mathcal{A}$ do
  if $f(X, \mathcal{D}) \geq \mu$ then
    output($X$)
  end if
end for

How many itemsets are enumerated?

$2^{\left|\mathcal{A}\right|}$. 
Naive Extraction

Naive algorithm

Input: \( O, A, D \subseteq O \times A, \mu \in \mathbb{N} \)
Output: \( \{ X \subseteq A \mid f(X, D) \geq \mu \} \)
for all \( X \subseteq A \) do
  if \( f(X, D) \geq \mu \) then
    output \( (X) \)
  end if
end for

Question
How many itemsets are enumerated?
Naive Extraction

Naive algorithm

Input: $O, A, D \subseteq O \times A, \mu \in \mathbb{N}$
Output: $\{X \subseteq A \mid f(X, D) \geq \mu\}$
for all $X \subseteq A$ do
    if $f(X, D) \geq \mu$ then
        output$(X)$
    end if
end for

Question

How many itemsets are enumerated? $2^{|A|}$. 
Naive Extraction

Naive algorithm

Input: \( O, A, D \subseteq O \times A, \mu \in \mathbb{N} \)
Output: \( \{ X \subseteq A \mid f(X, D) \geq \mu \} \)

for all \( X \subseteq A \) do
    if \( f(X, D) \geq \mu \) then
        output \( X \)
    end if
end for

Question

How many itemsets are enumerated? \( 2^{|A|} \).
**Transactional representation of the data**

**Relational representation:**
\[ D \subseteq O \times A \]

<table>
<thead>
<tr>
<th></th>
<th>( a_1 )</th>
<th>( a_2 )</th>
<th>...</th>
<th>( a_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( o_1 )</td>
<td>( d_{1,1} )</td>
<td>( d_{1,2} )</td>
<td>...</td>
<td>( d_{1,n} )</td>
</tr>
<tr>
<td>( o_2 )</td>
<td>( d_{2,1} )</td>
<td>( d_{2,2} )</td>
<td>...</td>
<td>( d_{2,n} )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( o_m )</td>
<td>( d_{m,1} )</td>
<td>( d_{m,2} )</td>
<td>...</td>
<td>( d_{m,n} )</td>
</tr>
</tbody>
</table>

where \( d_{i,j} \in \{ \text{true}, \text{false} \} \)

**Transactional representation:**
\[ D \]

is an array of subsets of \( A \)

\[ t_1 \]
\[ t_2 \]
\[ ... \]
\[ t_m \]

where \( t_i \subseteq A \)
Transactional representation of the data

Relational representation:
\[ D \subseteq O \times A \]

| \( o \) | \( a_1 \) | \( a_2 \) | \( \ldots \) | \( a_n \) |
|---|---|---|---|
| 1 | \( d_{1,1} \) | \( d_{1,2} \) | \( \ldots \) | \( d_{1,n} \) |
| 2 | \( d_{2,1} \) | \( d_{2,2} \) | \( \ldots \) | \( d_{2,n} \) |
| \vdots | \vdots | \vdots | \vdots | \vdots |
| \( o_m \) | \( d_{m,1} \) | \( d_{m,2} \) | \( \ldots \) | \( d_{m,n} \) |

Transactionally representation: \( D \) is a array of subsets of \( A \)

\[ t_1 \]
\[ t_2 \]
\[ \vdots \]
\[ t_m \]

where \( t_i \subseteq A \)

where \( d_{i,j} \in \{ \text{true}, \text{false} \} \)

For a linear time verification of “\( X \) being a subset of \( t_i \)”, the transactions are sorted (arbitrary order on \( A \)) in a pre-processing step and any enumerated itemset \( X \) respects this order.
Naive Extraction

Transaction representation of the data

Relational representation:
\[ D \subseteq \mathcal{O} \times \mathcal{A} \]

<table>
<thead>
<tr>
<th>( o_1 )</th>
<th>( a_1 )</th>
<th>( a_2 )</th>
<th>( \ldots )</th>
<th>( a_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d_{1,1} )</td>
<td>( d_{1,2} )</td>
<td>( \ldots )</td>
<td>( d_{1,n} )</td>
<td></td>
</tr>
<tr>
<td>( o_2 )</td>
<td>( d_{2,1} )</td>
<td>( d_{2,2} )</td>
<td>( \ldots )</td>
<td>( d_{2,n} )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \ddots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( o_m )</td>
<td>( d_{m,1} )</td>
<td>( d_{m,2} )</td>
<td>( \ldots )</td>
<td>( d_{m,n} )</td>
</tr>
</tbody>
</table>

where \( d_{i,j} \in \{ \text{true}, \text{false} \} \)

For a linear time verification of \( X \) being a subset of \( t_i \), the transactions are sorted (arbitrary order on \( \mathcal{A} \)) in a pre-processing step and any enumerated itemset \( X \) respects this order.
Prefix-based enumeration
Naive Extraction

Computation of \( f(X, D) \)

**Input:** \( X, D \) as an array of subsets of \( A \)

**Output:** \( f(X, D) \)

\[
s \leftarrow 0 \ \\
\text{for all } t \in D \text{ do} \ \\
\quad \text{if } X \subseteq t \text{ then} \ \\
\quad \quad s \leftarrow s + 1 \ \\
\quad \text{end if} \ \\
\text{end for} \ \\
\text{return } s
\]
Complexity of the naive approach

Question

How many itemsets are enumerated? $2^{|A|}$. 
Naive Extraction

Complexity of the naive approach

**Question**
How many itemsets are enumerated? $2^{|\mathcal{A}|}$.

**Question**
What is the worst-case complexity of computing $f(X, D)$?
Complexity of the naive approach

**Question**
How many itemsets are enumerated? $2^{|A|}$.

**Question**
What is the worst-case complexity of computing $f(X, D)$? $O(|\mathcal{O} \times A|)$. 
Question
How many itemsets are enumerated? $2^{|A|}$.

Question
What is the worst-case complexity of computing $f(X, D)$?
$O(|O \times A|)$.

Question
What is the worst-case complexity of the naive approach?
Question

How many itemsets are enumerated? $2^{|A|}$.

Question

What is the worst-case complexity of computing $f(X, D)$?
$O(|O \times A|)$.

Question

What is the worst-case complexity of the naive approach?
$O(2^{|A|} |O \times A|)$. 
Outline

1. Frequent Itemset
2. Frequent Itemset Mining
3. Naive Extraction
4. APriori
5. Eclat
6. FP-Growth
7. Conclusion
Anti-monotonicity of the frequency

**Theorem**

Given a dataset $\mathcal{D}$ of objects described with Boolean attributes in $\mathcal{A}$:

$$\forall (X, Y) \in 2^\mathcal{A} \times 2^\mathcal{A}, X \subseteq Y \Rightarrow f(X, \mathcal{D}) \geq f(Y, \mathcal{D}) .$$

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$o_2$</td>
<td>$\times$</td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>$o_3$</td>
<td>$\times$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$o_4$</td>
<td></td>
<td>$\times$</td>
<td></td>
</tr>
</tbody>
</table>

- $f(\emptyset, \mathcal{D}) = 4$
- $f(\{a_1\}, \mathcal{D}) = 2$
- $f(\{a_1, a_2\}, \mathcal{D}) = 2$
- $f(\{a_1, a_2, a_3\}, \mathcal{D}) = 1$
Anti-monotonicity of the frequency

**Theorem**

Given a dataset $\mathcal{D}$ of objects described with Boolean attributes in $\mathcal{A}$:

$$\forall (X, Y) \in 2^A \times 2^A, X \subseteq Y \Rightarrow f(X, \mathcal{D}) \geq f(Y, \mathcal{D}) .$$

<table>
<thead>
<tr>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$o_2$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$o_3$</td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>$o_4$</td>
<td>$\times$</td>
<td></td>
</tr>
</tbody>
</table>

- $f(\emptyset, \mathcal{D}) = 4$
- $f(\{a_3\}, \mathcal{D}) = 2$
- $f(\{a_1, a_3\}, \mathcal{D}) = 1$
- $f(\{a_1, a_2, a_3\}, \mathcal{D}) = 1$
Corollary

Given a dataset \( \mathcal{D} \) of objects described with Boolean attributes in \( \mathcal{A} \) and a minimal frequency \( \mu \in \mathbb{N} \):

\[
\forall (X, Y) \in 2^A \times 2^A, X \subseteq Y \Rightarrow \left( f(Y, \mathcal{D}) \geq \mu \Rightarrow f(X, \mathcal{D}) \geq \mu \right).
\]

<table>
<thead>
<tr>
<th>( \mathcal{O}_1 )</th>
<th>( \mathcal{O}_2 )</th>
<th>( \mathcal{O}_3 )</th>
<th>( \mathcal{O}_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
</tr>
<tr>
<td>( a_3 )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
  f(\emptyset, \mathcal{D}) &= 4 \\
  f(\{a_3\}, \mathcal{D}) &= 2 \\
  f(\{a_1, a_3\}, \mathcal{D}) &= 1 \\
  f(\{a_1, a_2, a_3\}, \mathcal{D}) &= 1
\end{align*}
\]
Anti-monotonicity of the frequency

**Corollary**

Given a dataset $D$ of objects described with Boolean attributes in $A$ and a minimal frequency $\mu \in \mathbb{N}$:

$$\forall (X, Y) \in 2^A \times 2^A, X \subseteq Y \Rightarrow \left( f(X, D) < \mu \Rightarrow f(Y, D) < \mu \right).$$

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1$</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>$o_2$</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>$o_3$</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>$o_4$</td>
<td></td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>

- $f(\emptyset, D) = 4$
- $f(\{a_3\}, D) = 2$
- $f(\{a_1, a_3\}, D) = 1$
- $f(\{a_1, a_2, a_3\}, D) = 1$
Pruning the enumeration tree ($\mu = 3$)
APriori

Pruning the enumeration tree ($\mu = 3$)
To check the frequency of every parent, the enumeration tree must be traversed breadth-first.
To check the frequency of every parent, the enumeration tree must be traversed breadth-first.

The two first parents (in the lexicographic order \( \preceq \)) are close to each other in the prefix-based tree. Indeed, they only differ by the last attribute. Instead of considering all possible children of a parent, APriori searches this second parent and, if found, enumerate, by union, their child.
Level-wise enumeration of the itemsets
APriori algorithm

**Input:** \( A, D \) as an array of subsets of \( A, \mu \in \mathbb{N} \)

**Output:** \( \{X \subseteq A \mid f(X, D) \geq \mu\} \)

\[ \mathcal{P} \leftarrow \{\{a\} \mid a \in A\} \]

**while** \( \mathcal{P} \neq \emptyset \) **do**

\[ \mathcal{P} \leftarrow \text{output\_frequent}(\mathcal{P}, D, \mu) \]

\[ \mathcal{P} \leftarrow \text{children}(\mathcal{P}) \]

**end while**
**APriori**

*children*

**Input:** A lexicographically ordered collection $\mathcal{P} \subseteq 2^A$

**Output:** \( \{X \subseteq 2^A \mid \forall a \in X, X \setminus \{a\} \in \mathcal{P}\} \) lexico. ordered

\( \mathcal{P}' \leftarrow \emptyset \)

for all \( P_1 \in \mathcal{P} \) do
  for all \( P_2 \in \{P_2 \in \mathcal{P} \mid P_1 \prec P_2 \land P_2 \setminus \{\text{last}(P_2)\} = P_1 \setminus \{\text{last}(P_1)\}\} \) do
    \( X \leftarrow P_1 \cup P_2 \)
    if \( \forall P \in \{X \setminus \{\text{member}(X)\} \mid P_2 \prec P\}, P \in \mathcal{P} \) then
      \( \mathcal{P}' \leftarrow \mathcal{P}' \cup \{X\} \)
    end if
  end if
end for

return \( \mathcal{P}' \)
**APriori**

**children**

**Input:** A lexicographically ordered collection $\mathcal{P} \subseteq 2^A$

**Output:** $\{ X \subseteq 2^A \mid \forall a \in X, X \setminus \{a\} \in \mathcal{P} \}$ lexico. ordered

$\mathcal{P}' \leftarrow \emptyset$

for all $P_1 \in \mathcal{P}$ do

for all $P_2 \in \{ P_2 \in \mathcal{P} \mid P_1 \prec P_2 \land P_2 \setminus \{\text{last}(P_2)\} = P_1 \setminus \{\text{last}(P_1)\} \}$

do

$X \leftarrow P_1 \cup P_2$

if $\forall P \in \{ X \setminus \{\text{member}(X)\} \mid P_2 \prec P \}, P \in \mathcal{P}$ then

$\mathcal{P}' \leftarrow \mathcal{P}' \cup \{ X \}$

end if

end for

end for

return $\mathcal{P}'$
Question

Assuming the largest frequent itemsets contain \( l \) attributes. How many itemsets are enumerated at worst?

\[
\sum_{k=1}^{l} (|A|^k) = \sum_{k=1}^{l} n^k
\]
**Question**

Assuming the largest frequent itemsets contain \( l \) attributes. How many itemsets are enumerated at worst? \[ \sum'_{k=1} \binom{|\mathcal{A}|}{k} \].
APriori

**output_frequent**

**Input:** A lexicographically ordered collection \( \mathcal{P} \subseteq 2^A \) containing itemsets of size \( k \), \( \mathcal{D} \) as an array of subsets of \( \mathcal{A} \), \( \mu \in \mathbb{N} \)

**Output:** \( \{X \subseteq \mathcal{P} \mid f(X, \mathcal{D}) \geq \mu\} \) lexicographically ordered

for all \( X \in \mathcal{P} \) do
  \( s[X] \leftarrow 0 \)
end for

for all \( t \in \mathcal{D} \) do
  for all \( X \in \{X \subseteq t \mid |X| = k\} \) do
    if \( X \in \mathcal{P} \) then
      \( s[X] \leftarrow s[X] + 1 \)
    end if
  end for
end for

return \( \text{aux}(\mathcal{P}, s, \mu) \)
**APriori**

**output_frequent**

**Input:** A lexicographically ordered collection $\mathcal{P} \subseteq 2^\mathcal{A}$ containing itemsets of size $k$, $\mathcal{D}$ as an array of subsets of $\mathcal{A}$, $\mu \in \mathbb{N}$

**Output:** $\{X \subseteq \mathcal{P} \mid f(X, \mathcal{D}) \geq \mu\}$ lexicographically ordered

for all $X \in \mathcal{P}$ do
  $s[X] \leftarrow 0$
end for

for all $t \in \mathcal{D}$ do
  for all $X \in \{X \subseteq t \mid |X| = k\}$ do
    if $X \in \mathcal{P}$ then
      $s[X] \leftarrow s[X] + 1$
    end if
  end for
end for

return aux($\mathcal{P}$, $s$, $\mu$)
Input: A lexicographically ordered collection $\mathcal{P} \subseteq 2^A$, $s$ associating every $X \in \mathcal{P}$ with its frequency, $\mu \in \mathbb{N}$

Output: $\{X \subseteq \mathcal{P} \mid s[X] \geq \mu\}$ lexicographically ordered

$\mathcal{F} \leftarrow \emptyset$

for all $X \in \mathcal{P}$ do
  if $s[X] \geq \mu$ then
    output($X$)
    $\mathcal{F} \leftarrow \mathcal{F} \cup \{X\}$
  end if
end for

return $\mathcal{F}$
Question

What is the worst-case complexity of APriori assuming the largest frequent itemsets contain $l$ attributes?

\[ O\left(\sum_{k=1}^{l} |A|^k\right) \] considering $O(k)$ searches of an itemset of size $k$... and if I don't forget anything!

The data are only read $l$ times. If they do not fit into main memory, that makes a huge difference with the naive algorithm.
What is the worst-case complexity of APriori assuming the largest frequent itemsets contain $l$ attributes? $O(|O| \sum_{k=1}^{l} k \binom{|A|}{k})$ considering $O(k)$ searches of an itemset of size $k$... and if I don’t forget anything!
Complexity of APriori

Question

What is the worst-case complexity of APriori assuming the largest frequent itemsets contain \( l \) attributes? \( O(|O| \sum_{k=1}^{l} k \left( \frac{|A|}{k} \right) ) \) considering \( O(k) \) searches of an itemset of size \( k \)... and if I don’t forget anything!

The data are only read \( l \) times. If they do not fit into main memory, that makes a huge difference with the naive algorithm.
Observation

An itemset has a greater probability to be infrequent if the frequencies of its attributes, taken individually, are low.
Observation
An itemset has a greater probability to be infrequent if the frequencies of its attributes, taken individually, are low.

Fail-first principle
Taking advantage of the anti-monotonicity of the frequency, it is better to enumerate the infrequent itemsets first.
The unbalanced enumeration tree
Input: \( A, D \) as an array of subsets of \( A, \mu \in \mathbb{N} \)
Output: \( \{ X \subseteq A \mid f(X, D) \geq \mu \} \)
\[ P \leftarrow \{ \{a\} \mid a \in A \} \]
\[ \text{while } P \neq \emptyset \text{ do} \]
\[ P \leftarrow \text{output\_frequent}(P, D, \mu) \]
\[ P \leftarrow \text{children}(P) \]
\[ \text{end while} \]

Whatever the order on \( A \), the frequent itemsets are correctly and completely listed...
APriori

Heuristic choice of a lexicographic order

Input: $\mathcal{A}, \mathcal{D}$ as an array of subsets of $\mathcal{A}, \mu \in \mathbb{N}$
Output: $\{X \subseteq \mathcal{A} \mid f(X, \mathcal{D}) \geq \mu\}$

$\mathcal{P} \leftarrow \{\{a\} \mid a \in \mathcal{A}\}$ ordered by increasing $f(\{a\}, \mathcal{D})$

while $\mathcal{P} \neq \emptyset$ do

$\mathcal{P} \leftarrow \text{output\_frequent}(\mathcal{P}, \mathcal{D}, \mu)$

$\mathcal{P} \leftarrow \text{children}(\mathcal{P})$

end while

Whatever the order on $\mathcal{A}$, the frequent itemsets are correctly and completely listed... but this heuristic choice usually leads to the enumeration of much less infrequent itemsets.
Outline

1. Frequent Itemset
2. Frequent Itemset Mining
3. Naive Extraction
4. APriori
5. Eclat
6. FP-Growth
7. Conclusion
Iterative computation of the supports

**Theorem**

Given the objects in $O$ described with the Boolean attributes in $A$, i.e., the dataset $D \subseteq O \times A$ and $k \in \mathbb{N}$ itemsets $(P_i)_{i=1..k} \in (2^A)^k$:

$$
\{o \in O \mid \{o\} \times \bigcup_{i=1}^{k} P_i \subseteq D\} = \bigcap_{i=1}^{k} \{o \in O \mid \{o\} \times P_i \subseteq D\}.
$$

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$o_2$</td>
<td>$\times$</td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>$o_3$</td>
<td></td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>$o_4$</td>
<td></td>
<td></td>
<td>$\times$</td>
</tr>
</tbody>
</table>

$$
\{o \in O \mid \{o\} \times \{a_1\} \subseteq D\} = \{o_1, o_2\},
$$

$$
\{o \in O \mid \{o\} \times \{a_2\} \subseteq D\} = \{o_1, o_2, o_3\},
$$

$$
\{o \in O \mid \{o\} \times \{a_3\} \subseteq D\} = \{o_1, o_4\},
$$

$$
\{o \in O \mid \{o\} \times \{a_1, a_2, a_3\} \subseteq D\} = \{o_1\}.
$$
Iterative computation of the supports

**Theorem**

Given the objects in $\mathcal{O}$ described with the Boolean attributes in $\mathcal{A}$, i.e., the dataset $\mathcal{D} \subseteq \mathcal{O} \times \mathcal{A}$ and $k \in \mathbb{N}$ itemsets $(P_i)_{i=1..k} \in (2^{\mathcal{A}})^k$:

$$\{o \in \mathcal{O} \mid \{o\} \times \bigcup_{i=1}^k P_i \subseteq \mathcal{D}\} = \bigcap_{i=1}^k \{o \in \mathcal{O} \mid \{o\} \times P_i \subseteq \mathcal{D}\}.$$

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$o_2$</td>
<td>$\times$</td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>$o_3$</td>
<td></td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>$o_4$</td>
<td></td>
<td></td>
<td>$\times$</td>
</tr>
</tbody>
</table>

$$\begin{align*}
\{o \in \mathcal{O} \mid \{o\} \times \{a_1, a_2\} \subseteq \mathcal{D}\} &= \{o_1, o_2\} \\
\{o \in \mathcal{O} \mid \{o\} \times \{a_3\} \subseteq \mathcal{D}\} &= \{o_1, o_4\} \\
\{o \in \mathcal{O} \mid \{o\} \times \{a_1, a_2, a_3\} \subseteq \mathcal{D}\} &= \{o_1\}
\end{align*}$$
Eclat

Iterative computation of the supports

Theorem

Given the objects in \( \mathcal{O} \) described with the Boolean attributes in \( \mathcal{A} \), i.e., the dataset \( \mathcal{D} \subseteq \mathcal{O} \times \mathcal{A} \) and \( k \in \mathbb{N} \) itemsets \((P_i)_{i=1..k} \in (2^\mathcal{A})^k\):

\[
\{o \in \mathcal{O} | \{o\} \times \bigcup_{i=1}^k P_i \subseteq \mathcal{D}\} = \bigcap_{i=1}^k \{o \in \mathcal{O} | \{o\} \times P_i \subseteq \mathcal{D}\}.
\]

<table>
<thead>
<tr>
<th></th>
<th>(a_1)</th>
<th>(a_2)</th>
<th>(a_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(o_1)</td>
<td>(\times)</td>
<td>(\times)</td>
<td>(\times)</td>
</tr>
<tr>
<td>(o_2)</td>
<td>(\times)</td>
<td>(\times)</td>
<td></td>
</tr>
<tr>
<td>(o_3)</td>
<td></td>
<td>(\times)</td>
<td></td>
</tr>
<tr>
<td>(o_4)</td>
<td></td>
<td></td>
<td>(\times)</td>
</tr>
</tbody>
</table>

\[
\{o \in \mathcal{O} | \{o\} \times \{a_1, a_2\} \subseteq \mathcal{D}\} = \{o_1, o_2\}
\]

\[
\{o \in \mathcal{O} | \{o\} \times \{a_1, a_3\} \subseteq \mathcal{D}\} = \{o_1\}
\]

\[
\{o \in \mathcal{O} | \{o\} \times \{a_1, a_2, a_3\} \subseteq \mathcal{D}\} = \{o_1\}
\]
Eclat

Vertical representation of the data

Relational representation:
\[ D \subseteq O \times A \]

Vertical representation: \( D \) is an array of subsets of \( O \)

\[
\begin{array}{c|cccc}
   & a_1 & a_2 & \ldots & a_n \\
\hline
o_1 & d_{1,1} & d_{1,2} & \ldots & d_{1,n} \\
o_2 & d_{2,1} & d_{2,2} & \ldots & d_{2,n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
o_m & d_{m,1} & d_{m,2} & \ldots & d_{m,n} \\
\end{array}
\]

where \( d_{i,j} \in \{\text{true}, \text{false}\} \)

where \( i_j \subseteq O \)
Eclat

Vertical representation of the data

Relational representation:
\[ D \subseteq O \times A \]

Vertical representation: \( D \) is an array of subsets of \( O \)

\[
\begin{array}{c|cccc}
   & a_1 & a_2 & \ldots & a_n \\
\hline
 o_1 & d_{1,1} & d_{1,2} & \ldots & d_{1,n} \\
 o_2 & d_{2,1} & d_{2,2} & \ldots & d_{2,n} \\
 \vdots & \vdots & \vdots & \ddots & \vdots \\
 o_m & d_{m,1} & d_{m,2} & \ldots & d_{m,n} \\
\end{array}
\]

where \( d_{i,j} \in \{\text{true, false}\} \)

For a linear time intersection of the \( i_j \), they are sorted (arbitrary order on \( O \)) in a pre-processing step and the support of any enumerated itemset \( X \) will respect this order.
Vertical representation of the data

Relational representation:
\[ D \subseteq O \times A \]

| \( o_1 \) | \( d_{1,1} \) | \( d_{1,2} \) | ... | \( d_{1,n} \) |
| \( o_2 \) | \( d_{2,1} \) | \( d_{2,2} \) | ... | \( d_{2,n} \) |
| ... | ... | ... | ... | ... |
| \( o_m \) | \( d_{m,1} \) | \( d_{m,2} \) | ... | \( d_{m,n} \) |

where \( d_{i,j} \in \{\text{true},\text{false}\} \)

Unless the minimal relative frequency is very low, storing the support on \textit{bitsets} provide the best space and time performances.
Eclat

Eclat enumeration

Like APriori:

- The anti-monotonicity of the frequency prunes the enumeration tree;
Eclat

Eclat enumeration

Like APriori:

- The anti-monotonicity of the frequency prunes the enumeration tree;
- the two first parents (in the lexicographic order \( \preceq \)) are searched to generate by union their child;
Eclat enumeration

Like APriori:

- The anti-monotonicity of the frequency prunes the enumeration tree;
- the two first parents (in the lexicographic order \( \preceq \)) are searched to generate by union their child;
- Ordering the attributes by increasing frequency heuristically leads to the enumeration of much less infrequent itemsets.
Eclat enumeration

Like APriori:

- The anti-monotonicity of the frequency prunes the enumeration tree;
- the two first parents (in the lexicographic order $\preceq$) are searched to generate by union their child;
- Ordering the attributes by increasing frequency heuristically leads to the enumeration of much less infrequent itemsets.

However:

- the frequency of the other parents is not checked;
Eclat enumeration

Like APriori:

- The anti-monotonicity of the frequency prunes the enumeration tree;
- the two first parents (in the lexicographic order $\leq$) are searched to generate by union their child;
- Ordering the attributes by increasing frequency heuristically leads to the enumeration of much less infrequent itemsets.

However:

- the frequency of the other parents is not checked;
- thanks to that, the enumeration tree is traversed in a less memory-hungry way (but, contrary to APriori, the supports of the frequent itemsets are stored too).
Eclat

Pruning the enumeration tree ($\mu = 3$)
Eclat algorithm

**Input:** \( A, D \) as an array of subsets of \( O, \mu \in \mathbb{N} \)

**Output:** \( \{ X \subseteq A \mid f(X, D) \geq \mu \} \)

**Eclat**\((P, \mu)\) \{Initial call: \( P = \{(\{a_j\}, i_j) \mid j = 1..m \land |i_j| \geq \mu \}\}\)

for all \((P_1, i_{P_1}) \in P\) do

output\((P_1)\)

\(P' \leftarrow \emptyset\)

for all \((P_2, i_{P_2}) \in \{(P_2, i_{P_2}) \in P \mid P_1 \prec P_2\}\) do

\(i \leftarrow i_{P_1} \cap i_{P_2}\)

if \(|i| \geq \mu\) then

\(P' \leftarrow P' \cup \{(P_1 \cup P_2, i)\}\)

end if

end for

Eclat\((P', \mu)\)
end for
Question

What is the worst-case complexity of Eclat assuming the largest frequent itemsets contain $l$ attributes?

$O\left(\sum_{k=1}^{l} |A|^k\right)$... if I don’t forget anything!

Still assuming I do not forget anything, the space complexity is $O\left(l|A|\left(l+|O|\right)\right)$, whereas that of Apriori is $O\left(\sum_{k=1}^{l} k |A|^k\right)$.
Question

What is the worst-case complexity of Eclat assuming the largest frequent itemsets contain \( l \) attributes? 

\[
O(|\mathcal{O}| \sum_{k=1}^{l} \left( \frac{|A|}{k} \right))...
\]

if I don’t forget anything!
Eclat

Complexity of Eclat

**Question**

What is the worst-case complexity of Eclat assuming the largest frequent itemsets contain \( l \) attributes?  
\[ O(|O| \sum_{k=1}^{l} \binom{|A|}{k}) \]... if I don’t forget anything!

Still assuming I do not forget anything, the space complexity is  
\[ O(l|A|(l + |O|)) \], whereas that of APriori is  
\[ O(\sum_{k=1}^{l} k \binom{|A|}{k}) \).
Outline

1. Frequent Itemset
2. Frequent Itemset Mining
3. Naive Extraction
4. APriori
5. Eclat
6. FPGrowth
7. Conclusion

FPGrowth
FPGrowth

A compact representation of the data

Given the objects in \( \mathcal{O} \) and the Boolean attributes in \( \mathcal{A} \), a dataset \( \mathcal{D} \subseteq \mathcal{O} \times \mathcal{A} \) can represented by a prefix tree where nodes are labeled with:

1. an attribute \( a \in \mathcal{A} \);
2. the number of objects including every attribute from the node up to the root and no other attribute before \( a \) in the chosen order on \( \mathcal{A} \).
Given the objects in $\mathcal{O}$ and the Boolean attributes in $\mathcal{A}$, a dataset $\mathcal{D} \subseteq \mathcal{O} \times \mathcal{A}$ can be represented by a prefix tree where nodes are labeled with:

1. an attribute $a \in \mathcal{A}$;
2. the number of objects including every attribute from the node up to the root and no other attribute before $a$ in the chosen order on $\mathcal{A}$.

Ordering the attributes by decreasing $f(\{a\}, \mathcal{D})$ heuristically provides the most compact tree.
Growing a FPTree

From $\mathcal{D}$ as an array of subsets of $\mathcal{A}$ ordered by decreasing $f(\{a\}, \mathcal{D})$:

\[
\begin{align*}
&\{B, E, A, D\} \\
&\{B, E, C\} \\
&\{B, E, A, D\} \\
&\{B, E, A, C\} \\
&\{B, E, A, C, D\} \\
&\{B, C, D\}
\end{align*}
\]
From $\mathcal{D}$ as an array of subsets of $\mathcal{A}$ ordered by decreasing $f(\{a\}, \mathcal{D})$:

- $\{B, E, A, D\}$
- $\{B, E, C\}$
- $\{B, E, A, D\}$
- $\{B, E, A, C\}$
- $\{B, E, A, C, D\}$
- $\{B, C, D\}$
Growing a FPTree

From $\mathcal{D}$ as an array of subsets of $\mathcal{A}$ ordered by decreasing $f(\{a\}, \mathcal{D})$:

\[
\begin{align*}
\{B, E, A, D\} \\
\{B, E, C\} \\
\{B, E, A, D\} \\
\{B, E, A, C\} \\
\{B, E, A, C, D\} \\
\{B, C, D\}
\end{align*}
\]
Growing a FPTree

From $\mathcal{D}$ as an array of subsets of $\mathcal{A}$ ordered by decreasing $f(\{a\}, \mathcal{D})$:

- $\{B, E, A, D\}$
- $\{B, E, C\}$
- $\{B, E, A, D\}$
- $\{B, E, A, C\}$
- $\{B, E, A, C, D\}$
- $\{B, C, D\}$
Growing a FP Tree

From $\mathcal{D}$ as an array of subsets of $\mathcal{A}$ ordered by decreasing $f(\{a\}, \mathcal{D})$:

\[
\begin{align*}
\{B, E, A, D\} \\
\{B, E, C\} \\
\{B, E, A, D\} \\
\{B, E, A, C\} \\
\{B, E, A, C, D\} \\
\{B, C, D\}
\end{align*}
\]
From $\mathcal{D}$ as an array of subsets of $\mathcal{A}$ ordered by decreasing $f(\{a\}, \mathcal{D})$:

\{B, E, A, D\}  
\{B, E, C\}  
\{B, E, A, D\}  
\{B, E, A, C\}  
\{B, E, A, C, D\}  
\{B, C, D\}
Growing a FP Tree

From $D$ as an array of subsets of $A$ ordered by decreasing $f(\{a\}, D)$:

\[
\begin{align*}
\{B, E, A, D\} \\
\{B, E, C\} \\
\{B, E, A, D\} \\
\{B, E, A, C\} \\
\{B, E, A, C, D\} \\
\{B, C, D\}
\end{align*}
\]
FPgrowth

FPgrowth algorithm

Input: $\mathcal{A}, \mathcal{D}$ as an array of subsets of $\mathcal{A}$ ordered by decreasing $f(\{a\}, \mathcal{D})$ in a pre-processing step, $\mu \in \mathbb{N}$

Output: $\{X \subseteq \mathcal{A} \mid f(X, \mathcal{D}) \geq \mu\}$

$T \leftarrow \emptyset$

for all $t \in \mathcal{D}$ do
    insert$(t, T)$
end for

output_frequent$(\mathcal{A}, T, \mu, \emptyset)$
**Input:** \( t \subseteq A \) ordered, \( T \) the children of the root a FPTree respecting the same order

if \( t \neq \emptyset \) then

\[
a \leftarrow \text{first}(t)
\]

if \( a \in T \) then

\[
s_T[a] \leftarrow s_T[a] + 1
\]

else

\[
T \leftarrow T \cup \{a\}
\]

\[
T[a] \leftarrow \emptyset
\]

\[
s_T[a] \leftarrow 1
\]

end if

insert\((t \setminus \{a\}, T[a])\)

end if
To project a FPTree on an attribute $a$ it contains:

1. search the nodes labeled with $a$ (pointers allow a constant time access to the next node with the same label);
FP Growth

Projection of a FPTree

To project a FPTree on an attribute \( a \) it contains:

1. search the nodes labeled with \( a \) (pointers allow a constant time access to the next node with the same label);
2. Reset the frequency of the nodes above those found;
FP Growth

Projection of a FPTree

To project a FPTree on an attribute $a$ it contains:

1. search the nodes labeled with $a$ (pointers allow a constant time access to the next node with the same label);
2. Reset the frequency of the nodes above those found;
3. Traverse the FPTree from every node labeled with $a$ to the root adding, to the frequencies of the traversed nodes, the frequency of the starting node;

The tree projected on $a$ is the representation of the dataset containing all objects having $a$ (and only these objects).
To project a FPTree on an attribute \( a \) it contains:

1. search the nodes labeled with \( a \) (pointers allow a constant time access to the next node with the same label);
2. Reset the frequency of the nodes above those found;
3. Traverse the FPTree from every node labeled with \( a \) to the root adding, to the frequencies of the traversed nodes, the frequency of the starting node;
4. Cut down the subtrees rooted by the nodes labeled with \( a \).
To project a FPTree on an attribute $a$ it contains:

1. search the nodes labeled with $a$ (pointers allow a constant time access to the next node with the same label);
2. Reset the frequency of the nodes above those found;
3. Traverse the FPTree from every node labeled with $a$ to the root adding, to the frequencies of the traversed nodes, the frequency of the starting node;
4. Cut down the subtrees rooted by the nodes labeled with $a$.

The tree projected on $a$ is the representation of the dataset containing all objects having $a$ (and only these objects).
Projection of a FPTree: illustration

From $\mathcal{D}$ as an array of subsets of $\mathcal{A}$ ordered by decreasing $f(\{a\}, \mathcal{D})$:

<table>
<thead>
<tr>
<th>Subset</th>
</tr>
</thead>
<tbody>
<tr>
<td>${B, E, A, D}$</td>
</tr>
<tr>
<td>${B, E, C}$</td>
</tr>
<tr>
<td>${B, E, A, D}$</td>
</tr>
<tr>
<td>${B, E, A, C}$</td>
</tr>
<tr>
<td>${B, E, A, C, D}$</td>
</tr>
<tr>
<td>${B, C, D}$</td>
</tr>
</tbody>
</table>

Projection on $D$ of:

```
{ }
B(6)

{ }   E(5)   C(1)   D(1)
     /     /     /  \
A(4)  C(1)  D(1)
     /     /     /
D(2)  C(2)  D(1)
     /     /     /
D(1)  D(1)  D(1)
```
From $\mathcal{D}$ as an array of subsets of $\mathcal{A}$ ordered by decreasing $f(\{a\}, \mathcal{D})$:

- $\{B, E, A, D\}$
- $\{B, E, C\}$
- $\{B, E, A, D\}$
- $\{B, E, A, C\}$
- $\{B, E, A, C, D\}$
- $\{B, C, D\}$

FP-Tree of $\mathcal{D}$ projected on $\{D\}$:
From $\mathcal{D}$ as an array of subsets of $\mathcal{A}$ ordered by decreasing $f(\{a\}, \mathcal{D})$:

- $\{B, E, A, D\}$
- $\{B, E, C\}$
- $\{B, E, A, D\}$
- $\{B, E, A, C\}$
- $\{B, E, A, C, D\}$
- $\{B, C, D\}$

Projection on $\mathcal{A}$ of:

```
{1}
B(4)
  E(3)
  A(3)
  C(1)
C(1)
```
From $\mathcal{D}$ as an array of subsets of $\mathcal{A}$ ordered by decreasing $f(\{a\}, \mathcal{D})$:

$$\{B, E, A, D\}$$
$$\{B, E, C\}$$
$$\{B, E, A, D\}$$
$$\{B, E, A, C\}$$
$$\{B, E, A, C, D\}$$
$$\{B, C, D\}$$

FPTree of $\mathcal{D}$ projected on $\{D, A\}$:
The FPTree is recursively projected on the attributes ordered in the opposite way than the one used to build the FPTree, i.e., by increasing \( f(\{a\}, \mathcal{D}) \) (hence, the same heuristic as before to enumerate much less infrequent itemsets).
FPGrowth

Enumeration of FPGrowth

The FPTree is recursively projected on the attributes ordered in the opposite way than the one used to build the FPTree, i.e., by increasing $f(\{a\}, D)$ (hence, the same heuristic as before to enumerate much less infrequent itemsets).

After projection, the frequency of the root is the frequency of the attributes on which the FPTree was projected. Below the minimal frequency, the recursion is aborted (anti-monotonicity of the frequency).
The FPTree is recursively projected on the attributes ordered in the opposite way than the one used to build the FPTree, i.e., by increasing $f(\{a\}, D)$ (hence, the same heuristic as before to enumerate much less infrequent itemsets).

After projection, the frequency of the root is the frequency of the attributes on which the FPTree was projected. Below the minimal frequency, the recursion is aborted (anti-monotonicity of the frequency).

Once the projected FPTree is a path (i.e., has one single leaf), any subset of the attributes in the path union the attributes used for the projections is an itemset whose frequency is that of the leaf.
FPGrowth

output_frequent

Input: A ordered, T a FP Tree respecting the same order, μ ∈ N, P a prefix itemset
Output: \{P \cup Q \mid Q \subseteq A \land f(P \cup Q, \mathcal{D}) \geq \mu\}
if \ s_T[\{\} \geq \mu \ then
    output(P)
    while A \neq \emptyset do
        a, \leftarrow \text{last}(A)
        A \leftarrow A \setminus \{a\}
        aux(A, T, \mu, P, a)
    end while
end if
Input: \( A \) ordered, \( T \) a FP Tree respecting the same order, \( \mu \in \mathbb{N}, P \) a prefix itemset, \( a \) a new attribute for the prefix

Output: \( \{ P \cup \{ a \} \cup Q \mid Q \subseteq A \land f(P \cup \{ a \} \cup Q, D) \geq \mu \} \)

\( N \leftarrow \text{search\_nodes\_labeled}(a, T) \)

if \( N = \{ n \} \) then

if \( s[n] \geq \mu \) then

for all \( Q \subseteq \text{attributes\_to\_root}(n, T) \) do

output(\( P \cup \{ a \} \cup Q \))

end for

end if

else

output\_frequent(\( A, \text{project}(T, N), \mu, P \cup \{ a \} \))

end if
Question

What is the worst-case complexity of FPGrowth assuming the largest frequent itemsets contain / attributes?

\[ O(\left| O \right| \left| A \right| \log(\left| A \right|) + \sum_{l_k=1} \min(2^{\left| A \right| - k}, \left| D \right|) \left( \left| A \right| - k \right) \left( \left| A \right| - k \right) ) \]... if I don’t forget anything!

Still assuming I do not forget anything, the space complexity is \( O(\sum_{l_k=1} \min(2^{\left| A \right| - k}, \left| D \right|)) \).

All in all, FPGrowth is particularly efficient when many objects are described with a few Boolean attributes.
Complexity of FP-Growth

**Question**

What is the worst-case complexity of FP-Growth assuming the largest frequent itemsets contain \( l \) attributes?

\[
O(|O| |A| \log(|A|)) + \sum_{k=1}^{l} \min(2^{|A|-k}, |\{O\}|)(|A| - k) \left( \begin{array}{c} |A| \\ k \end{array} \right) \]

... if I don’t forget anything!
FPGrowth

Complexity of FPGrowth

**Question**

What is the worst-case complexity of FPGrowth assuming the largest frequent itemsets contain \( l \) attributes?

\[
O(|O||A| \log(|A|)) + \sum_{k=1}^{l} \min(2^{|A| - k}, |O|)(|A| - k) \binom{|A|}{k} \ldots \text{if I don't forget anything!}
\]

Still assuming I do not forget anything, the space complexity is

\[
O\left(\sum_{k=1}^{l} \min(2^{|A| - k}, |D|)\right).
\]
Question

What is the worst-case complexity of FPGrowth assuming the largest frequent itemsets contain \( l \) attributes?

\[
O(|\mathcal{O}| \log(|\mathcal{A}|)) + \sum_{k=1}^{l} \min(2^{|\mathcal{A}| - k}, |\mathcal{O}|)(|\mathcal{A}| - k) \left( \begin{array}{c}
|\mathcal{A}| \\
k
\end{array} \right) \ldots \text{if I don’t forget anything!}
\]

Still assuming I do not forget anything, the space complexity is \( O(\sum_{k=1}^{l} \min(2^{|\mathcal{A}| - k}, |\mathcal{D}|)) \). All in all, FPGrowth is particularly efficient when many objects are described with a few Boolean attributes.
Conclusion

Outline

1. Frequent Itemset
2. Frequent Itemset Mining
3. Naive Extraction
4. APriori
5. Eclat
6. FP Growth
7. Conclusion
Summary

- Frequent itemsets are subsets of Boolean attributes shared by enough objects;
Frequent itemsets are subsets of Boolean attributes shared by enough objects;

- Mining every frequent itemset is NP-hard;

Once discovered an infrequent itemset, its supersets need not be enumerated;

The support of an itemset can be computed by intersection of the support of two parents;

The FPTree is a compact representation of the data that helps in efficiently extracting the frequent itemsets;

All efficient algorithms enumerate the itemsets traversing a prefix tree;

Ordering the attributes by increasing frequencies heuristically decrease the number of infrequent itemsets that are enumerated.
Frequent itemsets are subsets of Boolean attributes shared by enough objects;

Mining every frequent itemset is NP-hard;

Once discovered an infrequent itemset, its supersets need not be enumerated;
Summary

- Frequent itemsets are subsets of Boolean attributes shared by enough objects;
- Mining every frequent itemset is NP-hard;
- Once discovered an infrequent itemset, its supersets need not be enumerated;
- The support of an itemset can be computed by intersection of the support of two parents;
Frequent itemsets are subsets of Boolean attributes shared by enough objects;

Mining every frequent itemset is NP-hard;

Once discovered an infrequent itemset, its supersets need not be enumerated;

The support of an itemset can be computed by intersection of the support of two parents;

The FPTree is a compact representation of the data that helps in efficiently extracting the frequent itemsets;
Conclusion

Summary

- Frequent itemsets are subsets of Boolean attributes shared by enough objects;
- Mining every frequent itemset is NP-hard;
- Once discovered an infrequent itemset, its supersets need not be enumerated;
- The support of an itemset can be computed by intersection of the support of two parents;
- The FP-Tree is a compact representation of the data that helps in efficiently extracting the frequent itemsets;
- All efficient algorithms enumerate the itemsets traversing a prefix tree;
Frequent itemsets are subsets of Boolean attributes shared by enough objects;

Mining every frequent itemset is NP-hard;

Once discovered an infrequent itemset, its supersets need not be enumerated;

The support of an itemset can be computed by intersection of the support of two parents;

The FP-Tree is a compact representation of the data that helps in efficiently extracting the frequent itemsets;

All efficient algorithms enumerate the itemsets traversing a prefix tree;

Ordering the attributes by increasing frequencies heuristically decrease the number of infrequent itemsets that are enumerated.
Advanced topics

- The algorithm usually considered the most efficient for itemset mining: LCM;
The algorithm usually considered the most efficient for itemset mining: LCM;

Generalization of itemset mining toward $n$-ary relations;
Conclusion

Advanced topics

- The algorithm usually considered the most efficient for itemset mining: LCM;
- Generalization of itemset mining toward $n$-ary relations;
- Generalization of itemset mining toward noise tolerance;
The algorithm usually considered the most efficient for itemset mining: LCM;

Generalization of itemset mining toward $n$-ary relations;

Generalization of itemset mining toward noise tolerance;

Efficient enforcement of other constraints than a minimal frequency;
Conclusion

Advanced topics

- The algorithm usually considered the most efficient for itemset mining: LCM;
- Generalization of itemset mining toward $n$-ary relations;
- Generalization of itemset mining toward noise tolerance;
- Efficient enforcement of other constraints than a minimal frequency;
- Condensed representations of the frequent itemsets.
©2011 Loïc Cerf

These slides are licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License.