Apriori Algorithm
Example & Improvement

Pantelis Vikatos
## Transactional data

<table>
<thead>
<tr>
<th>TID</th>
<th>List of item IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>T100</td>
<td>I1, I2, I5</td>
</tr>
<tr>
<td>T200</td>
<td>I2, I4</td>
</tr>
<tr>
<td>T300</td>
<td>I2, I3</td>
</tr>
<tr>
<td>T400</td>
<td>I1, I2, I4</td>
</tr>
<tr>
<td>T500</td>
<td>I1, I3</td>
</tr>
<tr>
<td>T600</td>
<td>I2, I3</td>
</tr>
<tr>
<td>T700</td>
<td>I1, I3</td>
</tr>
<tr>
<td>T800</td>
<td>I1, I2, I3, I5</td>
</tr>
<tr>
<td>T900</td>
<td>I1, I2, I3</td>
</tr>
</tbody>
</table>

**Exercise**: Generate Association Rules using Apriori Algorithm
Step 1 (generate candidate and frequent itemsets)

\[ s(\{A,B\}) = P(A \cup B) = \frac{\sigma(A \cup B)}{\# \text{of trans}} \]

Frequent Itemsets
minimum support is 2

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup.count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{I1,I2,I3}</td>
<td>2</td>
</tr>
<tr>
<td>{I1,I2,I5}</td>
<td>2</td>
</tr>
</tbody>
</table>
Step 2 (generate rules)

confidence(A → B) = P(B | A) = \frac{s(A \cup B)}{s(A)}

\( \mathbf{F} \mathbf{l}_1 : \{ I_1, I_2, I_3 \} \)

- \( I_1I_2 \rightarrow I_3 \), confidence = 2/4 = 50%
- \( I_1I_3 \rightarrow I_2 \), confidence = 2/4 = 50%
- \( I_2I_3 \rightarrow I_1 \), confidence = 2/4 = 50%
- \( I_1 \rightarrow I_2I_3 \), confidence = 2/6 = 33%
- \( I_2 \rightarrow I_1I_3 \), confidence = 2/7 = 29%
- \( I_3 \rightarrow I_1I_2 \), confidence = 2/6 = 33%

\( \mathbf{F} \mathbf{l}_2 : \{ I_1, I_2, I_5 \} \)

- \( I_1I_2 \rightarrow I_5 \), confidence = 2/4 = 50%
- \( I_1I_5 \rightarrow I_2 \), confidence = 2/2 = 100%
- \( I_2I_5 \rightarrow I_1 \), confidence = 2/2 = 100%
- \( I_1 \rightarrow I_2I_5 \), confidence = 2/6 = 33%
- \( I_2 \rightarrow I_1I_5 \), confidence = 2/7 = 29%
- \( I_5 \rightarrow I_1I_2 \), confidence = 2/2 = 100%
Step 2 (generate rules)

\[
\text{confidence}(A \rightarrow B) = P(B \mid A) = \frac{s(A \cup B)}{s(A)}
\]

\[FI_1 : \{ I1, I2, I3 \} \]

- \(I1I2 \rightarrow I3\), confidence = \(\frac{2}{4} = 50\%\)
- \(I1I3 \rightarrow I2\), confidence = \(\frac{2}{4} = 50\%\)
- \(I2I3 \rightarrow I1\), confidence = \(\frac{2}{4} = 50\%\)
- \(I1 \rightarrow I2I3\), confidence = \(\frac{2}{6} = 33\%\)
- \(I2 \rightarrow I1I3\), confidence = \(\frac{2}{7} = 29\%\)
- \(I3 \rightarrow I1I2\), confidence = \(\frac{2}{6} = 33\%\)

\[FI_2 : \{ I1, I2, I5 \} \]

- \(I1I2 \rightarrow I5\), confidence = \(\frac{2}{4} = 50\%\)
- \(I1I5 \rightarrow I2\), confidence = \(\frac{2}{2} = 100\%\)
- \(I2I5 \rightarrow I1\), confidence = \(\frac{2}{2} = 100\%\)
- \(I1 \rightarrow I2I5\), confidence = \(\frac{2}{6} = 33\%\)
- \(I2 \rightarrow I1I5\), confidence = \(\frac{2}{7} = 29\%\)
- \(I5 \rightarrow I1I2\), confidence = \(\frac{2}{2} = 100\%\)

Minimum confidence = 70\%
Reducing number of comparisons 1

Step 1 on mining association rules:
Scan the database of transactions to determine the support of each candidate itemset.

<table>
<thead>
<tr>
<th>TID</th>
<th>List of item IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>1,4,5</td>
</tr>
<tr>
<td>T2</td>
<td>2,5,8,9</td>
</tr>
<tr>
<td>T3</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>T499</td>
<td>3,6,7</td>
</tr>
<tr>
<td>T500</td>
<td>4,6,8,9</td>
</tr>
</tbody>
</table>

Ci of length 3:
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

Idea: Hash tree
Hash Tree

• A node of the tree contains either a list of itemsets, if it is a leaf node or hash table if it is an internal node.
• Each bucket of hash table of an internal node points to another node

We need:
• Hash function
• Max leaf size: max number of itemsets stored in a leaf
Hash Tree

• A node of the tree contains either a list of itemsets, if it is a leaf node or hash table if it is an internal node.
• Each bucket of hash table of an internal node points to another node

We need:
• Hash function
• Max leaf size: max number of itemsets stored in a leaf

Example:
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7},
{3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}
Hash Tree

• A node of the tree contains either a list of itemsets, if it is a leaf node or hash table if it is an internal node.
• Each bucket of hash table of an internal node points to another node

We need:
• Hash function
• Max leaf size: max number of itemsets stored in a leaf

Example:
\{1 4 5\}, \{1 2 4\}, \{4 5 7\}, \{1 2 5\}, \{4 5 8\}, \{1 5 9\}, \{1 3 6\}, \{2 3 4\}, \{5 6 7\},
\{3 4 5\}, \{3 5 6\}, \{3 5 7\}, \{6 8 9\}, \{3 6 7\}, \{3 6 8\}
Reducing number of comparisons 2

- Compress a large database into a compact, Frequent-Pattern tree (FP-tree).
- Avoids costly sweepings database.

Building FP Tree (2 steps)
1. Scan the transaction DB for the first time, find frequent items (single item patterns) and order them into a list L in frequency descending order.

2. For each transaction, order its frequent items according to the order in L; Scan DB the second time, construct FP-tree by putting each frequency ordered transaction onto it.
FP-tree (example)

Step 1: Scan the DB for the first time to create the L

<table>
<thead>
<tr>
<th>TID</th>
<th>List of items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T100</td>
<td>f, a, c, d, g, i, m, p</td>
</tr>
<tr>
<td>T200</td>
<td>a, b, c, f, l, m, o</td>
</tr>
<tr>
<td>T300</td>
<td>b, f, h, j, o</td>
</tr>
<tr>
<td>T400</td>
<td>b, c, k, s, p</td>
</tr>
<tr>
<td>T500</td>
<td>a, f, c, e, l, p, m, n</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>4</td>
</tr>
<tr>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>m</td>
<td>3</td>
</tr>
<tr>
<td>p</td>
<td>3</td>
</tr>
</tbody>
</table>
FP-tree (example)

Step 2: Scan the DB for the second time ordering frequent itemsets in each transaction.

<table>
<thead>
<tr>
<th>TID</th>
<th>List of items</th>
<th>(ordered) frequent items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T100</td>
<td>f, a, c, d, g, i, m, p</td>
<td>f, c, a, m, p</td>
</tr>
<tr>
<td>T200</td>
<td>a, b, c, f, l, m, o</td>
<td>f, c, a, b, m</td>
</tr>
<tr>
<td>T300</td>
<td>b, f, h, j, o</td>
<td>f, b</td>
</tr>
<tr>
<td>T400</td>
<td>b, c, k, s, p</td>
<td>c, b, p</td>
</tr>
<tr>
<td>T500</td>
<td>a, f, c, e, l, p, m, n</td>
<td>f, c, a, m, p</td>
</tr>
</tbody>
</table>
FP-tree (example)
FP-tree (example)
FP-tree (example)
FP-tree (advantages)

✓ Scan the DB only twice

✓ Completeness: the FP-tree contains all the information related to mining frequent patterns (given the min_support threshold).

✓ Compactness: The size of the tree is bounded by the occurrences of frequent items.

✓ The height of the tree is bounded by the maximum number of items in a transaction.
FP-tree (thoughts)

Why in ordering?

<table>
<thead>
<tr>
<th>TID</th>
<th>List of items</th>
<th>frequent items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T100</td>
<td>$f, a, c, d, g, i, m, p$</td>
<td>$f, c, a, m, p$</td>
</tr>
<tr>
<td>T500</td>
<td>$a, f, c, e, l, p, m, n$</td>
<td>$a, f, c, p, m$</td>
</tr>
</tbody>
</table>
FP-tree (thoughts)

Why in descending ordering?

<table>
<thead>
<tr>
<th>TID</th>
<th>List of items</th>
<th>(ascended) frequent items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T100</td>
<td>f, a, c, d, g, i, m, p</td>
<td>p, m, a, c, f</td>
</tr>
<tr>
<td>T200</td>
<td>a, b, c, f, l, m, o</td>
<td>m, b, a, c, f</td>
</tr>
<tr>
<td>T300</td>
<td>b, f, h, j, o</td>
<td>b, f</td>
</tr>
<tr>
<td>T400</td>
<td>b, c, k, s, p</td>
<td>p, b, c</td>
</tr>
<tr>
<td>T500</td>
<td>a, f, c, e, l, p, m, n</td>
<td>p, m, a, c, f</td>
</tr>
</tbody>
</table>
Representatives Association Rules

✓ The number of rules generated are mostly very large if you do not apply some criteria about the importance of rules.

✓ M. Kryszkiewicz : Product a minimal set of association rules from which arise all the other rules.

✓ Representative Association Rules

✓ Rules from others implementing an operator coverage
AR (s, c) : set of all association rules that satisfy the requirements for minimal support s and confidence c.

Coverage C of a rule X ⇒ Y is defined as follows:

\[ C(X \rightarrow Y) = \{X \cup Z \rightarrow V \mid Z, V \subseteq Y \land Z \cap V = \emptyset \land V \neq \emptyset\} \]

Each rule in C (X => Y) consists of a subset of the items contained in the rule X => Y.

The first member of a rule set that belongs to C (X => Y) composed of the items X and possibly of Y.

The second member of such a rule is a non-empty subset of the remaining items from the Y.
Representatives Association Rules

Property 1
✓ Let r be an association rule with support s and confidence c. Each rule r’ belonging to the cover C (r) is an association rule has support not less than s and confidence not less than c. The immediate consequence of this property is that if a rule r belongs to AR (s, c), then each of the rule r’ of C (r) will also belongs to the AR (s, c).

Property 2
✓ Consider two association rules r: X ⇒ Y and r’= (X’⇒Y’). Then r ∈ C(r’) ⇔ XUY ⊆ X’∪Y’ ∧ X ⊇ X’.

Property 3
✓ If an association rule r contains more items from an association rule r’ then r ∉ C(r’).
✓ If an association rule r: (X ⇒ Y) is less than an association rule r’: (X’⇒Y’) then r ∈ C (r’) if and only if XUY ⊆ X’∪Y’ and X ⊇ X’.
✓ If r: (X ⇒ Y) and r’: (X’⇒Y’) are different association rules with the same number of objects then r ∈ C(r’) if and only if XUY = X’∪Y’ and X ⊇ X’.
Representatives Association Rules

✓ The set of representative association rules with a specified minimum support $s$ and minimum confidence is defined as:

$$RR(s, c) = \{ r \in AR(S, C) \mid \neg \exists r' \in AR(s, c), r' \neq r \& r \in C(r) \}$$

Property 1
✓ If $r \in RR(s, c)$ then $C(r) \subseteq AR(s, c)$.

Property 2
✓ $\forall r \in AR(s, c) \exists r' \in RR(s, c) : r \in C(r)$.
Production Representatives Rules

✓ The algorithm producing the representative association rules requires that they have found the frequent itemsets.

✓ Algorithm FastGenAllRepresentative based on 2 properties

Property 1
✓ Suppose $\emptyset \neq X \subseteq Z \subseteq I$ and $r$ is a rule of the form $r: (X \Rightarrow Y) \in AR (s, c)$. Then the rule will belong to the RR $(s, c)$, if the following two conditions apply:

(i) $\text{maxSup} \leq s$ or $\text{maxSup} / \text{sup} (X) < c$,
   
   $\text{maxSup} = \max(\{\text{sup}(Z') \mid Z \subseteq Z' (I) \cup \{0\})$

(ii) $\nexists X', \emptyset \neq X' \subseteq X$ such that $(X' \Rightarrow Z \setminus X') \notin AR (s, c)$

The first condition ensures that the rule $r$ is not in the coverage of a rule larger than $r$. The second condition ensures that the rule $r$ is not in the coverage of a rule with a length equal to $r$. 
Production Representatives Rules

✓ The algorithm producing the representative association rules requires that they have found the frequent itemsets

✓ Algorithm FastGenAllRepresentative based on 2 properties

Property 2
✓ Let \(\emptyset \neq Z \subseteq Z' \subseteq I\). If \(\text{sup}(Z) = \text{sup}(Z')\) then no rule of the form \((X \Rightarrow Z \setminus X) \in AR(s, c)\) with \(\emptyset \neq X \subseteq Z\) does not belong to \(RR(s, c)\).

   Consequently if the frequent itemset \(Z\) holds that \(\text{maxSup} = \text{sup}(Z)\) then this itemset can not be produced representative rules.
Overall Representative Rules

✓ The approach of representative association rules is aimed at addressing the problem of producing many rules

✓ The set of representative association rules is a minimal set of association rules from which arise all the other rules. By a rule derived by the other rules apply to initial coverage of an operator.