Frequent set mining

Karl Potisepp
Tartu 2009
## Item sets

### Data set

<table>
<thead>
<tr>
<th>ID1</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### Transactions

- \{a, b, d\}
- \{a, c\}
- \{b, d, e\}
- \{d, e\}
- \{a, b, d\}
- \{a, c, e\}
Cover

The *cover* of item T is a set containing the identifiers of item sets (rows) in the data set to which item T belongs.

From the example data set, the cover of a is \{1,2,5,6\}.

Similarly, the cover of an item set is a set containing the ID-s of item sets where that item set is present.
Support

The *support* of an item set is the number of times it appears in the data set.

In the example data set, \(\text{supp}\{a,b,d\} = 2\).

The support of an item set is the number of elements in its cover.

Above, support is defined as an absolute value, but it can also be relative. In that case, the support of an item set is the percentage of how often the item set appears in the data set.
Apriori algorithm

Apriori was one of the first algorithms for finding frequent item sets from data sets.

The principle of the algorithm is to first find all frequent items, then start assembling itemsets from frequent items, ignoring nonfrequent items.
Apriori algorithm cont-d

I. Find all frequent items \((F_1)\).

II. Generate a list of candidate itemsets \((G)\), 2 items each.

III. While \(G\) contains at least one itemset:
   I. For each itemset \(I\) in \(G\): if \(I\) is frequent, add \(I\) to \(F_K\), where \(K\) is the number of items in \(I\).
   II. Generate a list of candidate itemsets, \(K+1\) items each.

IV. All frequent itemsets are now contained in variables \(F_1, ..., F_N\), where \(N\) is the number of items in the largest frequent itemset found.
The principle of the Eclat algorithm is to perform a recursive depth-first search, starting from 1 item sets and gradually increasing in size until no more frequent sets are found.
FP-tree is a tree structure that makes it easier to calculate item set supports.

Finding the support of an item set from an FP-tree only takes as many operations as there are items in the set in question.
FP-tree generation algorithm

1. Scan data set once, find all frequent items and their supports and sort them by their supports in descending order.
2. Create root of the FP-tree, $T$, labeled 'null'.
3. For each row in data set:
   1. Order items just like in the first step. Let the sorted list be $[p|P]$, where $p$ is the first item, and $P$ is the rest of the list.
   2. If $T$ has a child $N$, so that $N.name = p.name$, then add 1 to $N$'s count. Otherwise create a new child node $N$ with count 1 to $T$ and $N.name = p.name$. If $P$ is not empty, recursively repeat this step so that $T:=N$ and $[p|P]:=P$. 
FP tree example

Use blackboard.
Database sampling

Large data sets are necessary for accurate results, however getting these results with conventional association rule mining algorithms takes time. Database sampling reduces the execution time of these algorithms by first picking a random sample, determining association rules within the sample and verifying the results on the whole database.
## Database sampling - accuracy

<table>
<thead>
<tr>
<th>Chance of error</th>
<th>Error</th>
<th>Sample size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.01</td>
<td>27,000</td>
</tr>
<tr>
<td>0.01</td>
<td>0.001</td>
<td>38,000</td>
</tr>
<tr>
<td>0.01</td>
<td>0.0001</td>
<td>50,000</td>
</tr>
<tr>
<td>0.001</td>
<td>0.01</td>
<td>2,700,000</td>
</tr>
<tr>
<td>0.001</td>
<td>0.001</td>
<td>3,800,000</td>
</tr>
<tr>
<td>0.001</td>
<td>0.0001</td>
<td>5,000,000</td>
</tr>
</tbody>
</table>