Mining association rules

- Naïve method for finding association rules:
  - Use separate-and-conquer method
  - Treat every possible combination of attribute values as a separate class
- Two problems:
  - Computational complexity
  - Resulting number of rules (which would have to be pruned on the basis of support and confidence)
- But: we can look for high support rules directly!

Item sets

- Support: number of instances correctly covered by association rule
  - The same as the number of instances covered by all tests in the rule (LHS and RHS!)
- Item: one test/attribute-value pair
- Item set: all items occurring in a rule
- Goal: only rules that exceed pre-defined support
  ⇒ Do it by finding all item sets with the given minimum support and generating rules from them!

Weather data

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
</tbody>
</table>

Item sets for weather data

<table>
<thead>
<tr>
<th>One-item sets</th>
<th>Two-item sets</th>
<th>Three-item sets</th>
<th>Four-item sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outlook = Sunny (5)</td>
<td>Outlook = Sunny Temperature = Hot (2)</td>
<td>Outlook = Sunny Temperature = Hot Humidity = High (2)</td>
<td>Outlook = Sunny Temperature = Hot Humidity = High Play = No (2)</td>
</tr>
<tr>
<td>Temperature = Cool (4)</td>
<td>Outlook = Sunny Humidity = High (3)</td>
<td>Outlook = Sunny Humidity = High Windy = False (2)</td>
<td>Outlook = Rainy Temperature = Mild Windy = False Play = Yes (2)</td>
</tr>
</tbody>
</table>

- In total: 12 one-item sets, 47 two-item sets, 39 three-item sets, 6 four-item sets and 0 five-item sets (with minimum support of two)
Generating rules from an item set

Once all item sets with minimum support have been generated, we can turn them into rules.

Example:
Humidity = Normal, Windy = False, Play = Yes (4)

Seven \((2^N-1)\) potential rules:

- If Humidity = Normal and Windy = False then Play = Yes (4/4)
- If Humidity = Normal and Play = Yes then Windy = False (4/6)
- If Windy = False and Play = Yes then Humidity = Normal (4/6)
- If Humidity = Normal then Windy = False and Play = Yes (4/7)
- If Windy = False then Humidity = Normal and Play = Yes (4/8)
- If Play = Yes then Humidity = Normal and Windy = False (4/9)
- If True then Humidity = Normal and Windy = False and Play = Yes (4/12)

Rules for weather data

Rules with support > 1 and confidence = 100%:

<table>
<thead>
<tr>
<th>Association rule</th>
<th>Sup.</th>
<th>Conf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Humidity=Normal Windy=False =&gt; Play=Yes</td>
<td>4</td>
<td>100%</td>
</tr>
<tr>
<td>2 Temperature=Cool =&gt; Humidity=Normal</td>
<td>4</td>
<td>100%</td>
</tr>
<tr>
<td>3 Outlook=Overcast =&gt; Play=Yes</td>
<td>4</td>
<td>100%</td>
</tr>
<tr>
<td>4 Temperature=Cold Play=Yes =&gt; Humidity=Normal</td>
<td>3</td>
<td>100%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>58 Outlook=Sunny Temperature=Hot =&gt; Humidity=High</td>
<td>2</td>
<td>100%</td>
</tr>
</tbody>
</table>

In total:
- 3 rules with support four
- 5 with support three
- 50 with support two

Example rules from the same set

Item set:
Temperature = Cool, Humidity = Normal, Windy = False, Play = Yes (2)

Resulting rules (all with 100% confidence):

Temperature = Cool, Windy = False => Humidity = Normal, Play = Yes
Temperature = Cool, Windy = False, Humidity = Normal => Play = Yes
Temperature = Cool, Windy = False, Play = Yes => Humidity = Normal
due to the following “frequent” item sets:

Temperature = Cool, Windy = False (2)
Temperature = Cool, Humidity = Normal, Windy = False (2)
Temperature = Cool, Windy = False, Play = Yes (2)

Generating item sets efficiently

How can we efficiently find all frequent item sets?
Finding one-item sets easy

Idea: use one-item sets to generate two-item sets, two-item sets to generate three-item sets, ...

- If \((A\ B)\) is frequent item set, then \((A)\) and \((B)\) have to be frequent item sets as well!
- In general: if \(X\) is frequent \(k\)-item set, then all \((k-1)\)-item subsets of \(X\) are also frequent

\(\Rightarrow\) Compute \(k\)-item set by merging \((k-1)\)-item sets
Example

- Given: five three-item sets
  \((A B C), (A B D), (A C D), (A C E), (B C D)\)
- Lexicographically ordered!
- Candidate four-item sets:
  \((A B C D)\) OK because of \((A C D)\), \((B C D)\)
  \((A C D E)\) Not OK because of \((C D E)\)
- Final check by counting instances in dataset!
- \((k-1)\)-item sets are stored in hash table

Generating rules efficiently

- We are looking for all high-confidence rules
  - Support of antecedent obtained from hash table
    - But: brute-force method is \(2^{N-1}\)
  - Better way: building \((c+1)\)-consequent rules from \(c\)-consequent ones
    - Observation: \((c+1)\)-consequent rule can only hold if all corresponding \(c\)-consequent rules also hold
  - Resulting algorithm similar to procedure for large item sets

Example

- 1-consequent rules:
  - If Outlook = Sunny and Windy = False and Play = No
    then Humidity = High (2/2)
  - If Humidity = High and Windy = False and Play = No
    then Outlook = Sunny (2/2)

  Corresponding 2-consequent rule:
  - If Windy = False and Play = No
    then Outlook = Sunny and Humidity = High (2/2)

  - Final check of antecedent against hash table!

Association rules: discussion

- Above method makes one pass through the data for each different size item set
  - Other possibility: generate \((k+2)\)-item sets just after \((k+1)\)-item sets have been generated
  - Result: more \((k+2)\)-item sets than necessary will be considered but less passes through the data
  - Makes sense if data too large for main memory
- Practical issue: generating a certain number of rules (e.g. by incrementally reducing min. support)
Other issues

- Standard ARFF format very inefficient for typical market basket data
  - Attributes represent items in a basket and most items are usually missing
  - Data should be represented in sparse format
- Instances are also called transactions
- Confidence is not necessarily the best measure
  - Example: milk occurs in almost every supermarket transaction
  - Other measures have been devised (e.g. lift)