Frequent pattern mining under generalized subsumption

Jan Ramon, Jan Struyf, Luc De Raedt
Overview

- Introduction
- Representation languages
- Mining under generalized subsumption
- Implementation issues
- Condensed representations
- Conclusions and further work
Overview

- Introduction
  - Pattern mining
  - Example
- Representation languages
- Mining under generalized subsumption
- Implementation issues
- Condensed representations
- Conclusions and further work
Pattern mining

- Interesting patterns:
  - Frequent patterns
    - E.g. $q : a_1 \land a_2 \land \ldots \land a_n$
    - Frequency $freq(q)$
  - association rules
    - E.g. $r : \text{if } b \text{ then } h$
    - Support: $supp(r) = freq(b \land h)$
    - Confidence:
      $$conf(r) = \frac{freq(b \land h)}{freq(b)}$$
Main mining phases

1. Hypothesis space definition (bias)
2. Mining: iteration of
   (a) Candidate generation
   (b) Frequency testing $\rightarrow$ frequent patterns
   (c) Patterns $\rightarrow$ association rules
3. Interpretation
algorithm\; \textbf{WARMR}(E, R, \rho, \text{minsup})

\begin{align*}
k &= 0; \; C_0 = \{R\}; \; I = \emptyset \\
\textbf{while} \; C_k \neq \emptyset \\
&\quad \text{compute } \text{count}(c, E), \; \forall c \in C_k \\
&\quad F_k = \{c \in C_k \mid \text{count}(c) \geq \text{minsup} \cdot |D|\} \\
&\quad I = I \cup (C_k - F_k); \; C_{k+1} = \emptyset \\
&\quad \textbf{for each} \; c \in F_k, \; c' \in \rho(c) \\
&\quad \quad \textbf{if not} \; \text{tests}(c', c, C_{k+1}, I) \\
&\quad \quad \quad \textbf{then} \; C_{k+1} = C_{k+1} \cup \{c'\} \\
&\quad k = k + 1
\end{align*}
An example (1)

- Database recording what customers bought
- $\text{Freq(pizza)} = 1$, $\text{freq(beer)} = 0.75$

<table>
<thead>
<tr>
<th>customer</th>
<th>beer</th>
<th>wine</th>
<th>pizza</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Mary</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bart</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Lisa</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
An example (2)

- Find all itemsets with $freq \geq 0.6$
- Itemset = set of items. E.g. \{pizza, beer\}
An example (3)

Level 1 - Candidate generation

<table>
<thead>
<tr>
<th>Pattern</th>
<th>beer</th>
<th>wine</th>
<th>pizza</th>
</tr>
</thead>
<tbody>
<tr>
<td>beer</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>wine</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>pizza</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>customer</th>
<th>beer</th>
<th>wine</th>
<th>pizza</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Mary</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bart</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Lisa</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Frequent pattern mining under generalized subsumption – p.9/57
### Level 1 - Frequency counting

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>beer</td>
<td>0.75</td>
</tr>
<tr>
<td>wine</td>
<td>0.5</td>
</tr>
<tr>
<td>pizza</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>customer</th>
<th>beer</th>
<th>wine</th>
<th>pizza</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Mary</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bart</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Lisa</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
An example (5)

**Level 1 - Pruning infrequent**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Frequency</th>
<th>Frequent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>beer</td>
<td>0.75</td>
<td>yes</td>
</tr>
<tr>
<td>wine</td>
<td>0.5</td>
<td>no</td>
</tr>
<tr>
<td>pizza</td>
<td>1.0</td>
<td>yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>customer</th>
<th>beer</th>
<th>wine</th>
<th>pizza</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Mary</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bart</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Lisa</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
### Level 2 - Candidate generation

<table>
<thead>
<tr>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 \ beer ∨ beer</td>
</tr>
<tr>
<td>2 \ beer ∨ wine</td>
</tr>
<tr>
<td>3 \ beer ∨ pizza</td>
</tr>
<tr>
<td>4 \ pizza ∨ beer</td>
</tr>
<tr>
<td>5 \ pizza ∨ wine</td>
</tr>
<tr>
<td>6 \ pizza ∨ pizza</td>
</tr>
</tbody>
</table>
## Level 2 - Candidate pruning

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>beer∧beer</td>
<td>same as parent</td>
</tr>
<tr>
<td>beer∧wine</td>
<td>wine was infrequent</td>
</tr>
<tr>
<td>beer∧pizza</td>
<td></td>
</tr>
<tr>
<td>pizza∧beer</td>
<td>equivalent to 3</td>
</tr>
<tr>
<td>pizza∧wine</td>
<td>wine was infrequent</td>
</tr>
<tr>
<td>pizza∧pizza</td>
<td>same as parent</td>
</tr>
</tbody>
</table>
### Level 2 - Frequency counting

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>beer ∨ pizza</td>
<td>0.75</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>customer</th>
<th>beer</th>
<th>wine</th>
<th>pizza</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Mary</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bart</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Lisa</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Overview

- Introduction
- Representation languages
- Mining under generalized subsumption
- Implementation issues
- Condensed representations
- Conclusions and further work
Representation languages

- Propositional
- Syntactical relational
- Semantical relational
## Propositional language

- **Itemsets with what customers bought**
- **Freq(pizza)=1, freq(beer)=0.75**

<table>
<thead>
<tr>
<th>customer</th>
<th>beer</th>
<th>wine</th>
<th>pizza</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Mary</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bart</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Lisa</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Relational language

- Relational database
- \( \text{freq(buys(X,wine))=0.5, freq(friend(X,Y),buys(Y,wine))=1} \)

\begin{align*}
\text{buys(john,beer).} & \quad \text{buys(bart,wine).} & \quad \text{friend(bart,mary).} \\
\text{buys(john,wine).} & \quad \text{buys(bart,pizza).} & \quad \text{friend(bart,lisa).} \\
\text{buys(john,pizza).} & \quad \text{buys(lisa,beer).} & \\
\text{buys(mary,beer).} & \quad \text{buys(lisa,pizza).} & \\
\text{buys(mary,pizza).} & & \\
\end{align*}
With background knowledge

- Background knowledge defines general properties
- \( \text{freq}(\text{buys}(X, \text{wine})) = 0.5, \text{freq}(\text{buys}(X, Y), \text{alcohol}(Y)) = 1 \)

\begin{align*}
\text{buys}(\text{john}, \text{beer}). & \quad \text{buys}(\text{bart}, \text{wine}). & \quad \text{friend}(\text{bart}, \text{mary}). \\
\text{buys}(\text{john}, \text{wine}). & \quad \text{buys}(\text{bart}, \text{pizza}). & \quad \text{friend}(\text{bart}, \text{lisa}). \\
\text{buys}(\text{john}, \text{pizza}). & \quad \text{buys}(\text{lisa}, \text{beer}). & \quad \text{alcohol}(\text{beer}). \\
\text{buys}(\text{mary}, \text{beer}). & \quad \text{buys}(\text{lisa}, \text{pizza}). & \quad \text{alcohol}(\text{wine}). \\
\text{buys}(\text{mary}, \text{pizza}). & \phantom{\text{buys}(\text{lisa}, \text{pizza})}. & \phantom{\text{alcohol}(\text{beer}).} \\
\end{align*}
Background knowledge?

- User’s knowledge on the domain
- Including but not limited to
  - Hierarchies: alcohol(X) ← X=beer.
  - Symmetry: bond(X,Y) ← bond(Y,X).
  - Transitivity: X<Y ← X<Z, Z<Y.
  - Structures: has_benzene(M) :- atom(M,A1,c), bond(M,A1,A2,7), atom(M,A2,c), bond(M,A2,A3,7), atom(M,A3,c), bond(M,A3,A4,7), atom(M,A4,c), bond(M,A4,A5,7), atom(M,A5,c), bond(M,A5,A6,7), atom(M,A6,c), bond(M,A6,A1,7).
  - ...
Overview

- Introduction
- Representation languages
- Mining under generalized subsumption
- Implementation issues
- Condensed representations
- Conclusions and further work
Generalized subsumption

- Up to now, only syntactical relational approaches.
  - Only special-purpose solutions in propositional setting
- We use generalized subsumption

\[ G \supseteq S \iff (S \theta \land B \models G) \]

- Why use generalized subsumption?
- What is the effect? (implications)
Why general subsumption?

- To allow the algorithm to be more intelligent
- Use background knowledge
- Smaller set of rules with the same information
- Interpretability
- Efficiency
Implications (1)

- Semantically equivalent rules are treated as equivalent.
  - Syntactic: \( p(X, Y) \land p(X, X) \sim p(X, X) \)
  - Semantic:
    \[
    bond(X, Y) \land bond(Y, Z) \land X \neq Z \\
    \sim \\
    bond(Y, X) \land bond(Y, Z) \land X \neq Z
    \]
Implications: the user

- Eliminate all but one of each class of equivalent rules.
- Smaller set of rules, with same information content
- More interpretable for the user
- If the user provides earlier mined knowledge as background, he discovers (nearly) only new knowledge
Implications: efficiency

- Advantages:
  - More space-efficient (less rules)
  - More time-efficient (fewer candidates to refine, to test, ...)

- Disadvantages:
  - Testing generalized subsumption is more complex than test syntactical subsumption
Overview

- Introduction
- Representation languages
- Mining under generalized subsumption
- Implementation issues
- Condensed representations
- Conclusions and further work
Implementation issues

- What operations are needed?
- Which things can not be solved as usual?
- Implementation choices
Operations

- Main operations useful for algorithm:
  - Operations on Examples:
    - Coverage test
  - Operations on queries:
    - Monotonicity test
    - Equivalence test
    - Closure
Coverage test:

- Relations is more complex than propositional
- Syntactical is roughly the same as semantical
  - background knowledge was already used in many approaches for coverage testing
  - optimizations possible without background never applied
Operations

- Monotonicity test:
  - E.g. if $G$ is not frequent and $G \supseteq S$, then $S$ will not be frequent either.
  - E.g. if \textit{b}u\textit{y}s(X, Y), \textit{a}lch\textit{o}ho\textit{l}(Y) is not frequent, then \textit{b}u\textit{y}s(X, p\textit{i}zz\textit{a}), \textit{b}u\textit{y}s(X, c\textit{o}o\textit{k}i\textit{e}s), \textit{b}u\textit{y}s(X, w) is also infrequent.
  - Useful for large databases, not really indispensable (recompute freq).
Operations

- Equivalence test:
  - Avoid to get two equivalent patterns in solution.
  - E.g. \( \text{atom}(X, c), \text{bond}(X, Y), \text{atom}(Y, h) \) and \( \text{atom}(X, c), \text{bond}(Y, X), \text{atom}(Y, h) \) represent the same pattern.
Operations

- **Closure:**
  - Closure of $P$ is conjunction of literals that can be proved from $P \land B$.
  - $\text{closure}(\text{friend}(X, Y), \text{buys}(Y, \text{pizza}))$ is $\text{friend}(X, Y), \text{friend}(Y, X), \text{buys}(Y, \text{pizza})$.
  - When using condensed representations (e.g. closed sets, free sets)
  - Not really needed for std assoc rule mining
  - Useful in some taxonomy approaches
Implementation issues

- What operations are needed?
- Which things can not be solved as usual?
- Implementation choices
Closure

Possible problem: closure is infinite
  ♦ e.g. with infinite background relations

\[
larger(X, Y) : \neg X \text{ is } Y + 1.
\]
\[
larger(X, Y) : \neg X \text{ is } Z + 1, larger(Z, Y).
\]
  ♦ e.g. when using non-range-restricted clauses
Closure

Possible problem: closure is infinite

- e.g. with infinite background relations
- e.g. when using non-range-restricted clauses

\[
\text{person}(X) : \neg \text{friend}(X, Y).
\]
\[
\% \text{friends are persons}
\]
\[
\text{friend}(X, Y) : \neg \text{friend}(Y, X).
\]
\[
\text{friend}(X, Y) : \neg \text{person}(X).
\]
\[
\% \text{everyone has a friend}
\]
Closure

- Restrictions:
  - Range restricted background knowledge clauses
  - No predicates generating infinitely many constants
- Closures may be large but will be finite.
- Optimisations possible, e.g. for comparing
  - facts generated exclusively by background amy not be added to closure (only implicit)
Equivalence

- \( A \sim B \text{ iff } \text{closure}(A) = \text{closure}(B) \).

  - Restricts language
  - Requires 1 expensive operation (closure) for each pattern + efficient lookup.

  - cfr. syntactical: no lookup needed under a number of logics/settings
    - e.g. when using “object identity”
    - e.g. when using a refinement operator that adds literals in a deterministic order.
Equivalence

- $A \sim B$ if $A \succeq B$ and $B \succeq A$ (backtracking)
  - Requires several (less) expensive operations (subsumption tests) per pattern.
  - Pack-optimization possible.
Monotonicity

- No general solutions “linear” in the number of patterns
- Check for each “infrequent” patterns $I$ and each candidate pattern $C$ whether $I \supseteq C$. (Pack-optimization possible).
- More useful for shorter infrequent patterns
Implementation issues

- What operations are needed?
- Which things can not be solved as usual?
- Implementation choices
Implementation choices

- Two separate pieces of background knowledge (Most restrictions only apply to 2nd part)
  1. For querying data
  2. For defining subsumption

- Allow for different optimizations depending on properties of data set.
Overview

- Introduction
- Representation languages
- Mining under generalized subsumption
- Implementation issues
- Condensed representations
- Conclusions and further work
Condensed representations

- Association rules
- Closing the loop
- Free and closed patterns
Association rules

- Relational association rules:

  \[ \text{alcohol}(X) \leftarrow \text{beer}(X) \]

- Rule has \( \delta \) exceptions if for \( \delta \) examples there is an \( X \) for which \( \text{beer}(X) \) and not \( \text{alcohol}(X) \).
Association rules

■ Easy to find using a frequent pattern discovery algorithm

■ just add negation of head as “final” literal (do not refine patterns already having heads)

■ \((\text{body} \land \neg \text{head}) \iff \neg (\text{head} \leftarrow \text{body})\)

■ include in coverage test pack:

\(\text{prefix}, (\text{head}; \neg \text{head})\)
Closing the loop

- If an association rule

\[ h \leftarrow b_1 \land b_2 \land \ldots \land b_n \]

is discovered and has only few (at most \( \delta \)) exceptions, it can be added to the background theory.

- Allows the inductive knowledgebase to re-use discovered knowledge.
Free patterns

- If $q$ frequent and $p \leftarrow q$ high confidence, then $p \land q$ frequent.

$\Rightarrow \ p \land q$ is redundant if approximation good enough (=high confidence)

- Free pattern: no literal can be proven from other literals
Free patterns

- Rules from
  - Background knowledge (s-free)
  - Mined association rules (δ-free)
    - $\delta = \text{maximum number of allowed exceptions to rule}$

- Applying rules $r_1, r_2 \ldots r_n$ with at most $e_1, e_2 \ldots e_n$ exceptions \(\Rightarrow\) frequency can drop by at most $\sum_{i=1}^{n} e_i$
Closed patterns

- Pattern $p$ is closed wrt set of rules $R$ iff $p\theta$ is least Herbrand model of $R \cup p\theta$.
- Rules from background KB and from mined assoc rules.
- Closing a rule gives a closed rule whose frequency approximates the frequency of the original rule (depending on the number of added literals).
Experiments

- Experiments:
  - Naive implementation
  - No optimizations yet
  - Some simplifications

- Compare timings, # of patterns using different settings
Experiments - Mutagenesis / patterns

![Graph showing the reduction in the number of patterns for different levels. The graph includes lines for s-free, δ-free, δ-s-free, and closed patterns.](image)

Frequent pattern mining under generalized subsumption – p.52/57
Experiments - Mutagenesis

timings

![Graph showing level vs. reduction for s-free, δ-free, δ-s-free, and closed categories.](image)

Frequent pattern mining under generalized subsumption – p.53/57
Experiments

- Significant reduction of \# of patterns, even when approximating closely
- s-freeness is faster because no discovery of rules necessary
- closing comes after mining (in current system), so takes extra time.
Experiments - Carcinogenesis / # patterns

Frequent pattern mining under generalized subsumption – p.55/57
Experiments - Carcinogenesis / timings

Frequent pattern mining under generalized subsumption – p.56/57
Conclusions

- Methods to use domain knowledge from expert and mining
  - Generalization of definitions and algorithms
  - Efficiency issues
  - Preliminary implementations

- Further work:
  - Finish implementation
  - investigate in detail behavior and optimizations in different languages.