Mining Association Rules

CS 431 – Advanced Topics in AI

These slides are adapted from J. Han and M. Kamber’s book slides (http://www.cs.sfu.ca/~han)

What Is Association Mining?

- Association rule mining:
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
- Applications:
  - Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.
- Examples.
  - Rule form: “Body → Head [support, confidence]”.
  - buys(x, “diapers”) → buys(x, “lotion”) [0.5%, 60%]
  - major(x, “CS”) ^ takes(x, “DB”) → grade(x, “A”) [1%, 75%]
Association Rule: Basic Concepts

- Given: (1) database of transactions, (2) each transaction is a list of items (purchased by a customer in a visit)
- Find: all rules that correlate the presence of one set of items with that of another set of items
  - E.g., 98% of people who purchase tires and auto accessories also get automotive services done
- Applications
  - * ⇒ Maintenance Agreement (What the store should do to boost Maintenance Agreement sales)
  - Home Electronics ⇒ * (What other products should the store stocks up?)
  - Attached mailing in direct marketing
  - Detecting “ping-pong”ing of patients, faulty “collisions”

Rule Measures: Support and Confidence

Find all the rules $X \& Y \Rightarrow Z$ with minimum confidence and support
- support, $s$, probability that a transaction contains $\{X \; \& \; Y \; \& \; Z\}$
- confidence, $c$, conditional probability that a transaction having $\{X \; \& \; Y\}$ also contains $Z$

Let minimum support 50%, and minimum confidence 50%, we have
- $A \Rightarrow C$ (50%, 66.6%)
- $C \Rightarrow A$ (50%, 100%)

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Items Bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>A,B,C</td>
</tr>
<tr>
<td>1000</td>
<td>A,C</td>
</tr>
<tr>
<td>4000</td>
<td>A,D</td>
</tr>
<tr>
<td>5000</td>
<td>B,E,F</td>
</tr>
</tbody>
</table>
Association Rule Mining: A Road Map

- **Boolean vs. quantitative associations** (Based on the types of values handled)
  - buys(x, "SQLServer") ^ buys(x, "DMBook") → buys(x, "DBMiner") [0.2%, 60%]
  - age(x, "30..39") ^ income(x, "42..48K") → buys(x, "PC") [1%, 75%]
- Single dimension vs. multiple dimensional associations (see ex. Above)
- Single level vs. multiple-level analysis
- What brands of lotions are associated with what brands of diapers?
- Various extensions
  - Correlation, causality analysis
  - Association does not necessarily imply correlation or causality
  - Maxpatterns and closed itemsets
  - Constraints enforced
    - E.g., small sales (sum < 100) trigger big buys (sum > 1,000)?

Mining Association Rules—An Example

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Items Bought</th>
<th>Frequent Itemset</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>A,B,C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>A,C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000</td>
<td>A,D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>B,E,F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Min. support 50%
Min. confidence 50%

For rule \( A \Rightarrow C \):
- support = \( \text{support}(\{A \underline{\cup} C\}) = 50\%
- confidence = \( \frac{\text{support}(\{A \underline{\cup} C\})}{\text{support}(\{A\})} = 66.6\%\)

The Apriori principle:
- Any subset of a frequent itemset must be frequent
Mining Frequent Itemsets: the Key Step

- Find the frequent itemsets: the sets of items that have minimum support
  - A subset of a frequent itemset must also be a frequent itemset
    - i.e., if \{AB\} is a frequent itemset, both \{A\} and \{B\} should be a frequent itemset
  - Iteratively find frequent itemsets with cardinality from 1 to \(k\) (\(k\)-itemset)
- Use the frequent itemsets to generate association rules.

---

The Apriori Algorithm

- **Join Step:** \(C_i\) is generated by joining \(L_{i-1}\) with itself
- **Prune Step:** Any \((k-1)\)-itemset that is not frequent cannot be a subset of a frequent \(k\)-itemset
- **Pseudo-code:**
  
  \[
  \begin{align*}
  C_i : & \text{Candidate itemset of size } k \\
  L_k : & \text{frequent itemset of size } k \\
  L_j & = \{\text{frequent items}\}; \\
  \text{for } (k = 1; L_j \neq \emptyset; k++) \text{ do begin} \\
  C_{k+1} & = \text{candidates generated from } L_j; \\
  \text{for each transaction } t \text{ in database do} \\
  \quad \text{increment the count of all candidates in } C_{k+1} \\
  \quad \text{that are contained in } t \\
  L_{k+1} & = \text{candidates in } C_{k+1} \text{ with min_support} \\
  \text{end} \\
  \text{return } \bigcup_k L_k.
  \end{align*}
  \]
The Apriori Algorithm — Example

### Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
<th>C₁ itemset</th>
<th>sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
<td>{4}</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

### Itemset sup.

<table>
<thead>
<tr>
<th>Items</th>
<th>sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{4}</td>
<td>1</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

**Scan D**

**C₁**

### L₁

<table>
<thead>
<tr>
<th>itemset</th>
<th>sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{4}</td>
<td>1</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

### How to Generate Candidates?

- Suppose the items in \( L_{k-1} \) are listed in an order
- **Step 1**: self-joining \( L_{k-1} \)
  - insert into \( C_k \)
  - select \( p.item_1, p.item_2, ..., p.item_{k-2}, q.item_{k-1} \)
  - from \( L_{k-1} = p, L_{k-1} = q \)
  - where \( p.item_1=q.item_1, ..., p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1} \)
- **Step 2**: pruning
  - forall **itemsets** \( c \) **in** \( C_k \) do
    - forall (\( k-1 \))-subsets \( s \) **of** \( c \) do
      - if (\( s \) is not in \( L_{k-1} \)) then **delete** \( c \) **from** \( C_k \)
How to Count Supports of Candidates?

- Why counting supports of candidates a problem?
  - The total number of candidates can be very huge
  - One transaction may contain many candidates

Method:
- Candidate itemsets are stored in a hash-tree
- Leaf node of hash-tree contains a list of itemsets and counts
- Interior node contains a hash table
- Subset function: finds all the candidates contained in a transaction

Example of Generating Candidates

- \( L_3 = \{abc, abd, acd, ace, bcd\} \)
- Self-joining: \( L_3 \times L_3 \)
  - \( abcd \) from \( abc \) and \( abd \)
  - \( acde \) from \( acd \) and \( ace \)
- Pruning:
  - \( acde \) is removed because \( ade \) is not in \( L_3 \)
- \( C_4 = \{abcd\} \)
Methods to Improve Apriori’s Efficiency

- Hash-based itemset counting: A $k$-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- Transaction reduction: A transaction that does not contain any frequent $k$-itemset is useless in subsequent scans
- Partitioning: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
- Sampling: mining on a subset of given data, lower support threshold + a method to determine the completeness
- Dynamic itemset counting: add new candidate itemsets only when all of their subsets are estimated to be frequent

Is Apriori Fast Enough? — Performance Bottlenecks

- The core of the Apriori algorithm:
  - Use frequent $(k - 1)$-itemsets to generate candidate frequent $k$-itemsets
  - Use database scan and pattern matching to collect counts for the candidate itemsets
- The bottleneck of Apriori: candidate generation
  - Huge candidate sets:
    - $10^4$ frequent 1-itemset will generate $10^7$ candidate 2-itemsets
    - To discover a frequent pattern of size 100, e.g., $\{a_1, a_2, ..., a_{100}\}$, one needs to generate $2^{100} = 10^{30}$ candidates.
  - Multiple scans of database:
    - Needs $(n + 1)$ scans, $n$ is the length of the longest pattern
Mining Frequent Patterns Without Candidate Generation

- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure
  - highly condensed, but complete for frequent pattern mining
  - avoid costly database scans
- Develop an efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only!

Construct FP-tree from a Transaction DB

<table>
<thead>
<tr>
<th>TID</th>
<th>Items bought (ordered) frequent items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>{f, a, c, d, g, i, m, p} {f, c, a, m, p}</td>
</tr>
<tr>
<td>200</td>
<td>{a, b, c, f, l, m, o} {f, c, a, b, m}</td>
</tr>
<tr>
<td>300</td>
<td>{b, f, h, j, o} {f, b}</td>
</tr>
<tr>
<td>400</td>
<td>{b, c, k, s, p} {c, b, p}</td>
</tr>
<tr>
<td>500</td>
<td>{a, f, c, e, l, p, m, n} {f, c, a, m, p}</td>
</tr>
</tbody>
</table>

Steps:
1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Order frequent items in frequency descending order
3. Scan DB again, construct FP-tree

Header Table:

<table>
<thead>
<tr>
<th>Item frequency head</th>
</tr>
</thead>
<tbody>
<tr>
<td>f: 4</td>
</tr>
<tr>
<td>c: 4</td>
</tr>
<tr>
<td>a: 3</td>
</tr>
<tr>
<td>b: 3</td>
</tr>
<tr>
<td>m: 3</td>
</tr>
<tr>
<td>p: 3</td>
</tr>
</tbody>
</table>

min_support = 0.5
Benefits of the FP-tree Structure

- Completeness:
  - never breaks a long pattern of any transaction
  - preserves complete information for frequent pattern mining
- Compactness
  - reduce irrelevant information—infrequent items are gone
  - frequency descending ordering: more frequent items are more likely to be shared
  - never be larger than the original database (if not count node-links and counts)
- Example: For Connect-4 DB, compression ratio could be over 100

Iceberg Queries

- Iceberg query: Compute aggregates over one or a set of attributes only for those whose aggregate values is above certain threshold
- Example:
  
  ```sql
  select P.custID, P.itemID, sum(P.qty)
  from purchase P
  group by P.custID, P.itemID
  having sum(P.qty) >= 10
  ```
- Compute iceberg queries efficiently by Apriori:
  - First compute lower dimensions
  - Then compute higher dimensions only when all the lower ones are above the threshold
Multiple-Level Association Rules

- Items often form hierarchy.
- Items at the lower level are expected to have lower support.
- Rules regarding itemsets at appropriate levels could be quite useful.
- Transaction database can be encoded based on dimensions and levels
- We can explore shared multi-level mining

Mining Multi-Level Associations

- A top_down, progressive deepening approach:
  - First find high-level strong rules:
    \[ \text{milk} \rightarrow \text{bread} \quad [20\%, \ 60\%]. \]
  - Then find their lower-level "weaker" rules:
    \[ 2\% \text{milk} \rightarrow \text{wheat bread} \quad [6\%, \ 50\%]. \]
- Variations at mining multiple-level association rules.
- Level-crossed association rules:
  \[ 2\% \text{ milk} \rightarrow \text{Wonder wheat bread} \]
- Association rules with multiple, alternative hierarchies:
  \[ 2\% \text{ milk} \rightarrow \text{Wonder bread} \]
Multi-level Association: Uniform Support vs. Reduced Support

- Uniform Support: the same minimum support for all levels
  - + One minimum support threshold. No need to examine itemsets containing any item whose ancestors do not have minimum support.
  - – Lower level items do not occur as frequently. If support threshold
    - too high ⇒ miss low level associations
    - too low ⇒ generate too many high level associations
- Reduced Support: reduced minimum support at lower levels
  - There are 4 search strategies:
    - Level-by-level independent
    - Level-cross filtering by k-itemset
    - Level-cross filtering by single item
    - Controlled level-cross filtering by single item

Uniform Support

Multi-level mining with uniform support

Level 1
\[ \text{min\_sup = 5\%} \]

\[ \text{Milk} \quad [\text{support = 10\%}] \]

Level 2
\[ \text{min\_sup = 5\%} \]

\[ \text{2\% Milk} \quad [\text{support = 6\%}] \]

\[ \text{Skim Milk} \quad [\text{support = 4\%}] \]
Reduced Support

Multi-level mining with reduced support

Level 1
min_sup = 5%

Level 2
min_sup = 3%

Milk
[support = 10%]

2% Milk
[support = 6%]

Skim Milk
[support = 4%]

Multi-level Association: Redundancy Filtering

- Some rules may be redundant due to “ancestor” relationships between items.
- Example
  - milk ⇒ wheat bread [support = 8%, confidence = 70%]
  - 2% milk ⇒ wheat bread [support = 2%, confidence = 72%]
- We say the first rule is an ancestor of the second rule.
- A rule is redundant if its support is close to the “expected” value, based on the rule’s ancestor.
Multi-Level Mining: Progressive Deepening

- A top-down, progressive deepening approach:
  - First mine high-level frequent items:
    milk (15%), bread (10%)
  - Then mine their lower-level “weaker” frequent itemsets:
    2% milk (5%), wheat bread (4%)
- Different min_support threshold across multi-levels lead to different algorithms:
  - If adopting the same min_support across multi-levels
    then toss t if any of t’s ancestors is infrequent.
  - If adopting reduced min_support at lower levels
    then examine only those descendents whose ancestor’s support is frequent/non-negligible.

Progressive Refinement of Data Mining Quality

- Why progressive refinement?
  - Mining operator can be expensive or cheap, fine or rough
- Superset coverage property:
  - Preserve all the positive answers—allow a positive false test but not a false negative test.
- Two- or multi-step mining:
  - First apply rough/cheap operator (superset coverage)
  - Then apply expensive algorithm on a substantially reduced candidate set (Koperski & Han, SSD’95).
Multi-Dimensional Association: Concepts

- Single-dimensional rules:
  \[ \text{buys}(X, \text{“milk”}) \Rightarrow \text{buys}(X, \text{“bread”}) \]

- Multi-dimensional rules: \( \diamond \) 2 dimensions or predicates
  - Inter-dimension association rules (\textit{no repeated predicates})
    \[ \text{age}(X, \text{“19-25”}) \land \text{occupation}(X, \text{“student”}) \Rightarrow \text{buys}(X, \text{“coke”}) \]
  - hybrid-dimension association rules (\textit{repeated predicates})
    \[ \text{age}(X, \text{“19-25”}) \land \text{buys}(X, \text{“popcorn”}) \Rightarrow \text{buys}(X, \text{“coke”}) \]

Categorical Attributes
- finite number of possible values, no ordering among values

Quantitative Attributes
- numeric, implicit ordering among values

Techniques for Mining MD Associations

- Search for frequent \( k \)-predicate set:
  - Example: \{\textit{age}, occupation, buys\} is a 3-predicate set.
  - Techniques can be categorized by how age are treated.

1. Using static discretization of quantitative attributes
   - Quantitative attributes are statically discretized by using predefined concept hierarchies.

2. Quantitative association rules
   - Quantitative attributes are dynamically discretized into “bins” based on the distribution of the data.

3. Distance-based association rules
   - This is a dynamic discretization process that considers the distance between data points.
Static Discretization of Quantitative Attributes

- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges.
- In relational database, finding all frequent k-predicate sets will require k or k+1 table scans.
- Data cube is well suited for mining.
- The cells of an n-dimensional cuboid correspond to the predicate sets.
- Mining from data cubes can be much faster.

Quantitative Association Rules

- Numeric attributes are *dynamically* discretized
  - Such that the confidence or compactness of the rules mined is maximized.
- 2-D quantitative association rules: $A_{\text{quan1}} \land A_{\text{quan2}} \Rightarrow A_{\text{cat}}$
- Cluster “adjacent” association rules to form general rules using a 2-D grid.
- Example:

  $\text{age}(X,"30-34") \land \text{income}(X,"24K - 48K")$
  $\Rightarrow \text{buys}(X,"\text{high resolution TV"})$
ARCS (Association Rule Clustering System)

How does ARCS work?

1. Binning
2. Find frequent predicateset
3. Clustering
4. Optimize

Limitations of ARCS

- Only quantitative attributes on LHS of rules.
- Only 2 attributes on LHS. (2D limitation)
- An alternative to ARCS
  - Non-grid-based
  - equi-depth binning
  - clustering based on a measure of partial completeness.

"Mining Quantitative Association Rules in Large Relational Tables" by R. Srikant and R. Agrawal.
Mining Distance-based Association Rules

- Binning methods do not capture the semantics of interval data

<table>
<thead>
<tr>
<th>Price($)</th>
<th>Equi-width (width $10)</th>
<th>Equi-depth (depth 2)</th>
<th>Distance-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>[0,10]</td>
<td>[7,20]</td>
<td>[7,7]</td>
</tr>
<tr>
<td>20</td>
<td>[11,20]</td>
<td>[22,50]</td>
<td>[20,22]</td>
</tr>
<tr>
<td>22</td>
<td>[21,30]</td>
<td>[51,53]</td>
<td>[50,53]</td>
</tr>
<tr>
<td>50</td>
<td>[31,40]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>[41,50]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>[51,60]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Distance-based partitioning, more meaningful discretization considering:
  - density/number of points in an interval
  - “closeness” of points in an interval

Clusters and Distance Measurements

- $S[X]$ is a set of $N$ tuples $t_1, t_2, \ldots, t_N$, projected on the attribute set $X$
- The diameter of $S[X]$: 
  \[
  d(S[X]) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} dist_x(t_i[X], t_j[X])}{N(N-1)}
  \]
- $dist_x$: distance metric, e.g. Euclidean distance or Manhattan
The diameter, $d$, assesses the density of a cluster $C_x$, where

$$d(C_x) \leq d_0^x$$

$$|C_x| \geq s_0$$

Finding clusters and distance-based rules
- the density threshold, $d_0$, replaces the notion of support
- modified version of the BIRCH clustering algorithm

Interestingness Measurements
- Objective measures
  - Two popular measurements:
    - support; and
    - confidence
- Subjective measures (Silberschatz & Tuzhilin, KDD95)
  - A rule (pattern) is interesting if
    - it is unexpected (surprising to the user); and/or
    - actionable (the user can do something with it)
Criticism to Support and Confidence

Example 1: (Aggarwal & Yu, PODS98)
- Among 5000 students
  - 3000 play basketball
  - 3750 eat cereal
  - 2000 both play basketball and eat cereal
- play basketball ⇒ eat cereal [40%, 66.7%] is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%.
- play basketball ⇒ not eat cereal [20%, 33.3%] is far more accurate, although with lower support and confidence

<table>
<thead>
<tr>
<th></th>
<th>basketball</th>
<th>not basketball</th>
<th>sum(row)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cereal</td>
<td>2000</td>
<td>1750</td>
<td>3750</td>
</tr>
<tr>
<td>not cereal</td>
<td>1000</td>
<td>250</td>
<td>1250</td>
</tr>
<tr>
<td>sum(col.)</td>
<td>3000</td>
<td>2000</td>
<td>5000</td>
</tr>
</tbody>
</table>

Criticism to Support and Confidence (Cont.)

Example 2:
- X and Y: positively correlated,
- X and Z, negatively related
- support and confidence of X=>Z dominates
- We need a measure of dependent or correlated events

\[
corr_{A,B} = \frac{P(A \cup B)}{P(A)P(B)}
\]

<table>
<thead>
<tr>
<th>Rule</th>
<th>Support</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>X=&gt;Y</td>
<td>25%</td>
<td>50%</td>
</tr>
<tr>
<td>X=&gt;Z</td>
<td>37.50%</td>
<td>75%</td>
</tr>
</tbody>
</table>

P(B|A)/P(B) is also called the lift of rule A => B
Other Interestingness Measures: Interest

- Interest (correlation, lift) 
  \[ \frac{P(A \land B)}{P(A)P(B)} \]
  - taking both P(A) and P(B) in consideration
  - P(A \land B) = P(B) \times P(A), if A and B are independent events
  - A and B negatively correlated, if the value is less than 1; otherwise A and B positively correlated

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Support</th>
<th>Interest</th>
</tr>
</thead>
<tbody>
<tr>
<td>X.Y</td>
<td>25%</td>
<td>2</td>
</tr>
<tr>
<td>X.Z</td>
<td>37.50%</td>
<td>0.9</td>
</tr>
<tr>
<td>Y.Z</td>
<td>12.50%</td>
<td>0.57</td>
</tr>
</tbody>
</table>

Constraint-Based Mining

- Interactive, exploratory mining giga-bytes of data?
  - Could it be real? — Making good use of constraints!
- What kinds of constraints can be used in mining?
  - Knowledge type constraint: classification, association, etc.
  - Data constraint: SQL-like queries
    - Find product pairs sold together in Vancouver in Dec.'98.
  - Dimension/level constraints:
    - in relevance to region, price, brand, customer category.
  - Rule constraints
    - small sales (price $< 10) triggers big sales (sum $> 200).
  - Interestingness constraints:
    - strong rules (min_support \geq 3\%, min_confidence \geq 60\%).
Rule Constraints in Association Mining

- Two kind of rule constraints:
  - Rule form constraints: meta-rule guided mining.
    - $P(x, y) \land Q(x, w) \rightarrow \text{takes}(x, \text{"database systems"}).$
  - Rule (content) constraint: constraint-based query optimization (Ng, et al., SIGMOD'98).
    - $\text{sum}(\text{LHS}) < 100 \land \text{min}(\text{LHS}) > 20 \land \text{count}(\text{LHS}) > 3 \land \text{sum}(\text{RHS}) > 1000$
- 1-variable vs. 2-variable constraints (Lakshmanan, et al. SIGMOD'99):
  - 1-var: A constraint confining only one side (L/R) of the rule, e.g., as shown above.
  - 2-var: A constraint confining both sides (L and R).
    - $\text{sum}(\text{LHS}) < \text{min}(\text{RHS}) \land \text{max}(\text{RHS}) < 5 \times \text{sum}(\text{LHS})$

Constrain-Based Association Query

- Database: (1) trans (TID, Itemset), (2) itemInfo (Item, Type, Price)
- A constrained asso. query (CAQ) is in the form of $\{(S_1, S_2) \mid C\}$,
  - where C is a set of constraints on $S_1, S_2$ including frequency constraint
- A classification of (single-variable) constraints:
  - Class constraint: $S \subseteq A$.  \textit{e.g.} $S \subseteq \text{Item}$
  - Domain constraint:
    - $S \theta V, \theta \in \{=, \neq, \leq, >, \geq\}$. \textit{e.g.} $S.\text{Price} < 100$
    - $v \theta S, \theta \in \in \text{or } \not\in$. \textit{e.g.} snacks $\in S.\text{Type}$
    - $V \theta S$ or $S \theta V, \theta \in \{\subseteq, \subset, \not\subseteq, =, \neq\}$
    - \textit{e.g.} \{snacks, sodas\} $\subseteq S.\text{Type}$
  - Aggregation constraint: $agg(S) \theta V$, where $agg$ is in $\{\text{min, max, sum, count, avg}\}$, and $\theta \in \{=, \neq, \leq, >, \geq\}$.
    - \textit{e.g.} $\text{count}(S_1.\text{Type}) = 1, \text{avg}(S_2.\text{Price}) < 100$
Constrained Association Query Optimization Problem

- Given a CAQ = \{ (S_1, S_2) / C \}, the algorithm should be:
  - sound: It only finds frequent sets that satisfy the given constraints C
  - complete: All frequent sets satisfy the given constraints C are found
- A naïve solution:
  - Apply Apriori for finding all frequent sets, and then to test them for constraint satisfaction one by one.
- Our approach:
  - Comprehensive analysis of the properties of constraints and try to push them as deeply as possible inside the frequent set computation.

Anti-monotone and Monotone Constraints

- A constraint \( C_a \) is anti-monotone iff. for any pattern \( S \) not satisfying \( C_a \), none of the super-patterns of \( S \) can satisfy \( C_a \)
- A constraint \( C_m \) is monotone iff. for any pattern \( S \) satisfying \( C_m \), every super-pattern of \( S \) also satisfies it.
Succinct Constraint

- A subset of item \( I_s \) is a succinct set, if it can be expressed as \( \sigma_p(I) \) for some selection predicate \( p \), where \( \sigma \) is a selection operator.
- \( SP \subseteq 2^I \) is a succinct power set, if there is a fixed number of succinct set \( I_1, ..., I_k \subseteq I \), s.t. \( SP \) can be expressed in terms of the strict power sets of \( I_1, ..., I_k \) using union and minus.
- A constraint \( C_s \) is succinct provided \( SAT_{C_s}(I) \) is a succinct power set.

Convertible Constraint

- Suppose all items in patterns are listed in a total order \( R \).
- A constraint \( C \) is convertible anti-monotone iff a pattern \( S \) satisfying the constraint implies that each suffix of \( S \) w.r.t. \( R \) also satisfies \( C \).
- A constraint \( C \) is convertible monotone iff a pattern \( S \) satisfying the constraint implies that each pattern of which \( S \) is a suffix w.r.t. \( R \) also satisfies \( C \).
Relationships Among Categories of Constraints

- Succinctness
- Anti-monotonicity
- Monotonicity
- Convertible constraints
- Inconvertible constraints

Property of Constraints: Anti-Monotone

- Anti-monotonicity: If a set $S$ violates the constraint, any superset of $S$ violates the constraint.

- Examples:
  - $\text{sum}(S.Price) \leq v$ is anti-monotone
  - $\text{sum}(S.Price) \geq v$ is not anti-monotone
  - $\text{sum}(S.Price) = v$ is partly anti-monotone

- Application:
  - Push "$\text{sum}(S.price) \leq 1000$" deeply into iterative frequent set computation.
Characterization of Anti-Monotonicity Constraints

<table>
<thead>
<tr>
<th>$S \emptyset v, \theta \in {=, \leq, \geq}$</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v \in S$</td>
<td>no</td>
</tr>
<tr>
<td>$S \supseteq V$</td>
<td>no</td>
</tr>
<tr>
<td>$S \subseteq V$</td>
<td>yes</td>
</tr>
<tr>
<td>$S = V$</td>
<td>partly</td>
</tr>
<tr>
<td>$\min(S) \leq v$</td>
<td>no</td>
</tr>
<tr>
<td>$\min(S) \geq v$</td>
<td>yes</td>
</tr>
<tr>
<td>$\min(S) = v$</td>
<td>partly</td>
</tr>
<tr>
<td>$\max(S) \leq v$</td>
<td>yes</td>
</tr>
<tr>
<td>$\max(S) \geq v$</td>
<td>no</td>
</tr>
<tr>
<td>$\max(S) = v$</td>
<td>partly</td>
</tr>
<tr>
<td>$\text{count}(S) \leq v$</td>
<td>yes</td>
</tr>
<tr>
<td>$\text{count}(S) \geq v$</td>
<td>no</td>
</tr>
<tr>
<td>$\text{count}(S) = v$</td>
<td>partly</td>
</tr>
<tr>
<td>$\text{sum}(S) \leq v$</td>
<td>yes</td>
</tr>
<tr>
<td>$\text{sum}(S) \geq v$</td>
<td>no</td>
</tr>
<tr>
<td>$\text{sum}(S) = v$</td>
<td>partly</td>
</tr>
<tr>
<td>$\text{avg}(S) \emptyset v, \theta \in {=, \leq, \geq}$</td>
<td>convertible</td>
</tr>
<tr>
<td>(frequent constraint)</td>
<td>(yes)</td>
</tr>
</tbody>
</table>

Example of Convertible Constraints: Avg(S) $\emptyset$ V

- Let R be the value descending order over the set of items
  - E.g. I={9, 8, 6, 4, 3, 1}
- Avg(S) $\geq v$ is convertible monotone w.r.t. R
  - If S is a suffix of $S_1$, avg($S_1$) $\geq$ avg(S)
    - $\{8, 4, 3\}$ is a suffix of $\{9, 8, 4, 3\}$
    - avg($\{9, 8, 4, 3\}$)=6 $\geq$ avg($\{8, 4, 3\}$)=5
  - If S satisfies avg(S) $\geq v$, so does $S_1$
    - $\{8, 4, 3\}$ satisfies constraint avg(S) $\geq 4$, so does $\{9, 8, 4, 3\}$
Property of Constraints: Succinctness

- **Succinctness:**
  - For any set $S_1$ and $S_2$ satisfying $C$, $S_1 \cup S_2$ satisfies $C$.
  - Given $A_1$ is the sets of size 1 satisfying $C$, then any set $S$ satisfying $C$ are based on $A_1$, i.e., it contains a subset belongs to $A_1$.
  - Example:
    - $\text{sum}(S.\text{Price }) \geq v$ is not succinct
    - $\text{min}(S.\text{Price }) \leq v$ is succinct
  - Optimization:
    - If $C$ is succinct, then $C$ is pre-counting prunable. The satisfaction of the constraint alone is not affected by the iterative support counting.

---

Characterization of Constraints by Succinctness

<table>
<thead>
<tr>
<th>$S \theta v, \theta \in {\leq, \geq}$</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v \in S$</td>
<td>yes</td>
</tr>
<tr>
<td>$S \subseteq V$</td>
<td>yes</td>
</tr>
<tr>
<td>$S \subseteq V$</td>
<td>yes</td>
</tr>
<tr>
<td>$S = V$</td>
<td>yes</td>
</tr>
<tr>
<td>$\text{min}(S) \leq v$</td>
<td>yes</td>
</tr>
<tr>
<td>$\text{min}(S) \geq v$</td>
<td>yes</td>
</tr>
<tr>
<td>$\text{min}(S) = v$</td>
<td>yes</td>
</tr>
<tr>
<td>$\text{max}(S) \leq v$</td>
<td>yes</td>
</tr>
<tr>
<td>$\text{max}(S) \geq v$</td>
<td>yes</td>
</tr>
<tr>
<td>$\text{max}(S) = v$</td>
<td>yes</td>
</tr>
<tr>
<td>$\text{count}(S) \leq v$</td>
<td>weakly</td>
</tr>
<tr>
<td>$\text{count}(S) \geq v$</td>
<td>weakly</td>
</tr>
<tr>
<td>$\text{count}(S) = v$</td>
<td>weakly</td>
</tr>
<tr>
<td>$\text{sum}(S) \leq v$</td>
<td>no</td>
</tr>
<tr>
<td>$\text{sum}(S) \geq v$</td>
<td>no</td>
</tr>
<tr>
<td>$\text{sum}(S) = v$</td>
<td>no</td>
</tr>
<tr>
<td>$\text{avg}(S) \theta v, \theta \in {\leq, \geq}$</td>
<td>no</td>
</tr>
<tr>
<td>(frequent constraint)</td>
<td>(no)</td>
</tr>
</tbody>
</table>
Why Is the Big Pie Still There?

- More on constraint-based mining of associations
  - Boolean vs. quantitative associations
    - Association on discrete vs. continuous data
  - From association to correlation and causal structure analysis
    - Association does not necessarily imply correlation or causal relationships
  - From intra-transaction association to inter-transaction associations
    - E.g., break the barriers of transactions (Lu, et al. TOIS’99).
  - From association analysis to classification and clustering analysis
    - E.g., clustering association rules

Summary

- Association rule mining
  - probably the most significant contribution from the database community in KDD
  - A large number of papers have been published
- Many interesting issues have been explored
- An interesting research direction
  - Association analysis in other types of data: spatial data, multimedia data, time series data, etc.
References

- R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95, 3-14, Taipei, Taiwan.
- D.W. Cheung, J. Han, V. Ng, and C.Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. ICDE'96, 106-114, New Orleans, LA.

References (2)

- J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. ICDE'99, Sydney, Australia.
- J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. VLDB'95, 420-431, Zurich, Switzerland.
- J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD'00, 1-12, Dallas, TX, May 2000.
- M. Kamber, J. Han, and J. Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. KDD'97, 207-210, Newport Beach, California.
References (3)

- H. Lu, J. Han, and L. Feng. Stock movement and n-dimensional inter-transaction association rules. SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'98), 12:1-12:7, Seattle, Washington.
- R. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. SIGMOD'98, 13-24, Seattle, Washington.

References (4)

- J. Pei, J. Han, and R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. DMKD'00, Dallas, TX, 11-20, May 2000.
- J. Pei and J. Han. Can We Push More Constraints into Frequent Pattern Mining? KDD'00, Boston, MA, Aug. 2000.
- B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. ICDE'98, 412-421, Orlando, FL.
References (5)

- R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. SIGMOD’96, 1-12, Montreal, Canada.