Chapter 3: Frequent Itemset Mining

Lecture: Prof. Dr. Thomas Seidl
Tutorials: Julian Busch, Evgeniy Faerman, Florian Richter, Klaus Schmid
Chapter 3: Frequent Itemset Mining

1) Introduction
   – Transaction databases, market basket data analysis

2) Mining Frequent Itemsets
   – Apriori algorithm, hash trees, FP-tree

3) Simple Association Rules
   – Basic notions, rule generation, interestingness measures

4) Further Topics

5) Extensions and Summary
What is Frequent Itemset Mining?

Frequent Itemset Mining:
Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.

- **Given:**
  - A set of items \( I = \{i_1, i_2, \ldots, i_m\} \)
  - A database of transactions \( D \), where a transaction \( T \subseteq I \) is a set of items

- **Task 1:** find all subsets of items that occur together in many transactions.
  - E.g.: 85% of transactions contain the itemset \{milk, bread, butter\}

- **Task 2:** find all rules that correlate the presence of one set of items with that of another set of items in the transaction database.
  - E.g.: 98% of people buying tires and auto accessories also get automotive service done

- **Applications:** Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, recommendation systems, etc.
Example: Basket Data Analysis

- **Transaction database**
  
  \[ D = \{\{\text{butter, bread, milk, sugar}\}; \]
  \[ \{\text{butter, flour, milk, sugar}\}; \]
  \[ \{\text{butter, eggs, milk, salt}\}; \]
  \[ \{\text{eggs}\}; \]
  \[ \{\text{butter, flour, milk, salt, sugar}\}\} \]

- **Question of interest:**
  - Which items are bought together frequently?

- **Applications**
  - Improved store layout
  - Cross marketing
  - Focused attached mailings / add-on sales
  - \* ⇒ *Maintenance Agreement* (What the store should do to boost Maintenance Agreement sales)
  - *Home Electronics* ⇒ \* (What other products should the store stock up?)

<table>
<thead>
<tr>
<th>items</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>{butter}</td>
<td>4</td>
</tr>
<tr>
<td>{milk}</td>
<td>4</td>
</tr>
<tr>
<td>{butter, milk}</td>
<td>4</td>
</tr>
<tr>
<td>{sugar}</td>
<td>3</td>
</tr>
<tr>
<td>{butter, sugar}</td>
<td>3</td>
</tr>
<tr>
<td>{milk, sugar}</td>
<td>3</td>
</tr>
<tr>
<td>{butter, milk, sugar}</td>
<td>3</td>
</tr>
<tr>
<td>{eggs}</td>
<td>2</td>
</tr>
</tbody>
</table>

...
Chapter 3: Frequent Itemset Mining

1) Introduction
   - Transaction databases, market basket data analysis

2) Mining Frequent Itemsets
   - Apriori algorithm, hash trees, FP-tree

3) Simple Association Rules
   - Basic notions, rule generation, interestingness measures

4) Further Topics
   - Hierarchical Association Rules
     • Motivation, notions, algorithms, interestingness
   - Quantitative Association Rules
     • Motivation, basic idea, partitioning numerical attributes, adaptation of apriori algorithm, interestingness

5) Extensions and Summary
**Mining Frequent Itemsets: Basic Notions**

- **Items** $I = \{i_1, i_2, \ldots, i_m\}$: a set of literals (denoting items)
- **Itemset** $X$: Set of items $X \subseteq I$
- **Database** $D$: Set of transactions $T$, each transaction is a set of items $T \subseteq I$
- Transaction $T$ contains an itemset $X$: $X \subseteq T$
- The items in transactions and itemsets are sorted lexicographically:
  - itemset $X = (x_1, x_2, \ldots, x_k)$, where $x_1 \leq x_2 \leq \ldots \leq x_k$
- **Length** of an itemset: number of elements in the itemset
- **$k$-itemset**: itemset of length $k$
- The **support** of an itemset $X$ is defined as: $support(X) = |\{T \in D | X \subseteq T\}|$
- **Frequent itemset**: an itemset $X$ is called frequent for database $D$ iff it is contained in more than $minSup$ many transactions: $support(X) \geq minSup$
- **Goal 1**: Given a database $D$ and a threshold $minSup$, find all frequent itemsets $X \in Pot(I)$. 
Mining Frequent Itemsets: Basic Idea

• Naïve Algorithm
  – count the frequency of all possible subsets of \( I \) in the database
  \( \Rightarrow \) too expensive since there are \( 2^m \) such itemsets for \( |I| = m \) items

• The Apriori principle (anti-monotonicity):
  *Any non-empty subset of a frequent itemset is frequent, too!*
  \( A \subseteq I \) with \( \text{support}(A) \geq \text{minSup} \) \( \Rightarrow \forall A' \subset A \land A' \neq \emptyset : \text{support}(A') \geq \text{minSup} \)
  *Any superset of a non-frequent itemset is non-frequent, too!*
  \( A \subseteq I \) with \( \text{support}(A) < \text{minSup} \) \( \Rightarrow \forall A' \supset A : \text{support}(A') < \text{minSup} \)

• Method based on the apriori principle
  – First count the 1-itemsets, then the 2-itemsets, then the 3-itemsets, and so on
  – When counting \((k+1)\)-itemsets, only consider those \((k+1)\)-itemsets where all subsets of length \( k \) have been determined as frequent in the previous step
The Apriori Algorithm

variable $C_k$: candidate itemsets of size $k$
variable $L_k$: frequent itemsets of size $k$

$L_1 = \{\text{frequent items}\}$

for $(k = 1; L_k \neq \emptyset; k++)$ do begin
// JOIN STEP: join $L_k$ with itself to produce $C_{k+1}$
// PRUNE STEP: discard $(k+1)$-itemsets from $C_{k+1}$ that contain non-frequent $k$-itemsets as subsets
$C_{k+1} = \text{candidates generated from } L_k$

for each transaction $t$ in database do
  Increment the count of all candidates in $C_{k+1}$ that are contained in $t$

$L_{k+1} = \text{candidates in } C_{k+1} \text{ with min\_support}$

return $\bigcup_k L_k$
Generating Candidates (Join Step)

- Requirements for set of all candidate \((k + 1)\)-itemsets \(C_{k+1}\)
  - \textit{Completeness:}
    Must contain all frequent \((k + 1)\)-itemsets (superset property \(C_{k+1} \supseteq L_{k+1}\))
  - \textit{Selectiveness:}
    Significantly smaller than the set of all \((k + 1)\)-subsets
  - Suppose the items are sorted by any order (e.g., lexicograph.)

- Step 1: Joining \((C_{k+1} = L_k \bowtie L_k)\)
  - Consider frequent \(k\)-itemsets \(p\) and \(q\)
  - \(p\) and \(q\) are joined if they share the same first \(k - 1\) items

\[
\text{insert into } C_{k+1} \\
\text{select } p.i_1, p.i_2, \ldots, p.i_{k-1}, p.i_k, q.i_k \\
\text{from } L_k : p, L_k : q \\
\text{where } p.i_1 = q.i_1, \ldots, p.i_{k-1} = q.i_{k-1}, p.i_k < q.i_k
\]
Generating Candidates (Prune Step)

- Step 2: Pruning \( L_{k+1} = \{ X \in C_{k+1} | \text{support}(X) \geq \minSup \} \)
  - Naïve: Check support of every itemset in \( C_{k+1} \) \( \leftarrow \) inefficient for huge \( C_{k+1} \)
  - Instead, apply Apriori principle first: Remove candidate \((k+1)\)-itemsets which contain a non-frequent \(k\)-subset \(s\), i.e., \(s \notin L_k\)

\[
\text{forall itemsets } c \text{ in } C_{k+1} \text{ do} \\
\quad \text{forall } k\text{-subsets } s \text{ of } c \text{ do} \\
\quad \quad \text{if } (s \text{ is not in } L_k) \text{ then delete } c \text{ from } C_{k+1}
\]

- Example 1
  - \( L_3 = \{(ACF), (ACG), (AFG), (AFH), (CFG)\} \)
  - Candidates after the join step: \( \{(ACFG), (AFGH)\} \)
  - In the pruning step: delete \( (AFGH) \) because \( (FGH) \notin L_3 \), i.e., \( (FGH) \) is not a frequent \(3\)-itemset; also \( (AGH) \notin L_3 \)
  - \( C_4 = \{(ACFG)\} \rightarrow \) check the support to generate \( L_4 \)
### Frequent Itemset Mining → Algorithms → Apriori Algorithm

**Apriori Algorithm – Full Example**

<table>
<thead>
<tr>
<th>TID</th>
<th>items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4 6</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
</tr>
<tr>
<td>400</td>
<td>1 5 6</td>
</tr>
</tbody>
</table>

**minSup = 0.5**

**database D**

<table>
<thead>
<tr>
<th>itemset</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>3</td>
</tr>
<tr>
<td>{2}</td>
<td>2</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>
<tr>
<td>{6}</td>
<td>2</td>
</tr>
</tbody>
</table>

**C₁**

- **scan D**
- **prune C₁**

<table>
<thead>
<tr>
<th>itemset</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>3</td>
</tr>
<tr>
<td>{2}</td>
<td>2</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>
<tr>
<td>{6}</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>itemset</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>3</td>
</tr>
<tr>
<td>{2}</td>
<td>2</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
<tr>
<td>{6}</td>
<td>2</td>
</tr>
</tbody>
</table>

**L₁**

- **L₁ ⊖ L₁**

<table>
<thead>
<tr>
<th>itemset</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>3</td>
</tr>
<tr>
<td>{2}</td>
<td>2</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
<tr>
<td>{6}</td>
<td>2</td>
</tr>
</tbody>
</table>

**C₂**

- **scan D**
- **prune C₃**

<table>
<thead>
<tr>
<th>itemset</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 2}</td>
<td></td>
</tr>
<tr>
<td>{1 3}</td>
<td></td>
</tr>
<tr>
<td>{1 5}</td>
<td></td>
</tr>
<tr>
<td>{1 6}</td>
<td></td>
</tr>
<tr>
<td>{2 3}</td>
<td></td>
</tr>
<tr>
<td>{2 5}</td>
<td></td>
</tr>
<tr>
<td>{2 6}</td>
<td></td>
</tr>
<tr>
<td>{3 5}</td>
<td></td>
</tr>
<tr>
<td>{3 6}</td>
<td></td>
</tr>
<tr>
<td>{5 6}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>itemset</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 2}</td>
<td></td>
</tr>
<tr>
<td>{1 3}</td>
<td></td>
</tr>
<tr>
<td>{1 5}</td>
<td></td>
</tr>
<tr>
<td>{1 6}</td>
<td></td>
</tr>
<tr>
<td>{2 3}</td>
<td></td>
</tr>
<tr>
<td>{2 5}</td>
<td></td>
</tr>
<tr>
<td>{2 6}</td>
<td></td>
</tr>
<tr>
<td>{3 5}</td>
<td></td>
</tr>
<tr>
<td>{3 6}</td>
<td></td>
</tr>
<tr>
<td>{5 6}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>itemset</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 2}</td>
<td>1</td>
</tr>
<tr>
<td>{1 3}</td>
<td>2</td>
</tr>
<tr>
<td>{1 5}</td>
<td>2</td>
</tr>
<tr>
<td>{1 6}</td>
<td>2</td>
</tr>
<tr>
<td>{2 3}</td>
<td>2</td>
</tr>
<tr>
<td>{2 5}</td>
<td>2</td>
</tr>
<tr>
<td>{2 6}</td>
<td>0</td>
</tr>
<tr>
<td>{3 5}</td>
<td>2</td>
</tr>
<tr>
<td>{3 6}</td>
<td>1</td>
</tr>
<tr>
<td>{5 6}</td>
<td>1</td>
</tr>
</tbody>
</table>

**C₃**

- **scan D**

<table>
<thead>
<tr>
<th>itemset</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 3 5}</td>
<td></td>
</tr>
<tr>
<td>{1 3 6}</td>
<td></td>
</tr>
<tr>
<td>{1 5 6}</td>
<td></td>
</tr>
<tr>
<td>{2 3 5}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>itemset</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 3 5}</td>
<td>1</td>
</tr>
<tr>
<td>{1 3 6}</td>
<td>2</td>
</tr>
<tr>
<td>{1 5 6}</td>
<td>2</td>
</tr>
<tr>
<td>{2 3 5}</td>
<td>2</td>
</tr>
</tbody>
</table>

**L₃**

- **L₃ ⊖ L₃**

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

**C₄**

- C₄ is empty
How to Count Supports of Candidates?

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

- Method: Hash-Tree
  - Candidate itemsets are stored in a hash-tree
  - Leaf nodes of hash-tree contain lists of itemsets and their support (i.e., counts)
  - Interior nodes contain hash tables
  - Subset function finds all the candidates contained in a transaction

\[ h(K) = K \mod 3 \]

- e.g. for 3-Itemsets

\[
\begin{align*}
(3 6 7) &\quad (3 5 7) &\quad (7 9 12) &\quad (1 4 11) &\quad (7 8 9) &\quad (2 3 8) &\quad (2 5 6) \\
(3 4 15) &\quad (3 7 11) &\quad (1 6 11) &\quad (1 7 9) &\quad (1 11 12) &\quad (5 6 7) &\quad (5 8 11)
\end{align*}
\]
Hash-Tree – Construction

- Searching for an itemset
  - Start at the root (level 1)
  - At level \( d \): apply the hash function \( h \) to the \( d \)-th item in the itemset

- Insertion of an itemset
  - search for the corresponding leaf node, and insert the itemset into that leaf
  - if an overflow occurs:
    - Transform the leaf node into an internal node
    - Distribute the entries to the new leaf nodes according to the hash function

\[ h(K) = K \mod 3 \]

for 3-Itemsets
Hash-Tree – Counting

- Search all candidate itemsets contained in a transaction \( T = (t_1 \, t_2 \ldots \, t_n) \) for a current itemset length of \( k \)
- At the root
  - Determine the hash values for each item \( t_1 \, t_2 \ldots \, t_{n-k+1} \) in \( T \)
  - Continue the search in the resulting child nodes
- At an internal node at level \( d \) (reached after hashing of item \( t_i \))
  - Determine the hash values and continue the search for each item \( t_j \) with \( i < j \leq n - k + d \)
- At a leaf node
  - Check whether the itemsets in the leaf node are contained in transaction \( T \)

in our example \( n=5 \) and \( k=3 \)

\[ h(K) = K \mod 3 \]

Transaction \((1, 3, 7, 9, 12)\)
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-itemsets will generate \(10^7\) candidate 2-itemsets
    • To discover a frequent pattern of size 100, e.g., \(\{a_1, a_2, \ldots, a_{100}\}\), one needs to generate \(2^{100} \approx 10^{30}\) candidates.
  – Multiple scans of database:
    • Needs \(n\) or \(n+1\) scans, \(n\) is the length of the longest pattern

→ Is it possible to mine the complete set of frequent itemsets without candidate generation?
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!

• Idea:
  – Compress database into FP-tree, retaining the itemset association information
  – Divide the compressed database into conditional databases, each associated with one frequent item and mine each such database separately.
Construct FP-tree from a Transaction DB

Steps for compressing the database into a FP-tree:
1. Scan DB once, find frequent 1-itemsets (single items)
2. Order frequent items in frequency descending order

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

header 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>

minSup=0.5

sort items in the order of descending support
Construct FP-tree from a Transaction DB

Steps for compressing the database into a FP-tree:

1. Scan DB once, find frequent 1-itemsets (single items)
2. Order frequent items in frequency descending order
3. Scan DB again, construct FP-tree starting with most frequent item per transaction

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

**header 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>

for each transaction only keep its frequent items sorted in descending order of their frequencies

1&2

for each transaction build a path in the FP-tree:
- If a path with common prefix exists:
  increment frequency of nodes on this path and append suffix
- Otherwise: create a new branch

3a
Construct FP-tree from a Transaction DB

Steps for compressing the database into a FP-tree:
1. Scan DB once, find frequent 1-itemsets (single items)
2. Order frequent items in frequency descending order
3. Scan DB again, construct FP-tree starting with most frequent item per transaction

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

header table references occurrences of the frequent items in the FP-tree
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)
  – Experiments demonstrate compression ratios over 100
Mining Frequent Patterns Using FP-tree

• General idea (divide-and-conquer)
  – Recursively grow frequent pattern path using the FP-tree
• Method
  – For each item, construct its conditional pattern-base (prefix paths), and then its conditional FP-tree
  – Repeat the process on each newly created conditional FP-tree …
  – …until the resulting FP-tree is empty, or it contains only one path (single path will generate all the combinations of its sub-paths, each of which is a frequent pattern)
Major Steps to Mine FP-tree

1) Construct conditional pattern base for each node in the FP-tree
2) Construct conditional FP-tree from each conditional pattern-base
3) Recursively mine conditional FP-trees and grow frequent patterns obtained so far
   - If the conditional FP-tree contains a single path, simply enumerate all the patterns
Major Steps to Mine FP-tree: Conditional Pattern Base

1) Construct conditional pattern base for each node in the FP-tree
   - Starting at the frequent header table in the FP-tree
   - Traverse FP-tree by following the link of each frequent item (dashed lines)
   - Accumulate all of transformed prefix paths of that item to form a conditional pattern base
     • For each item its prefixes are regarded as condition for it being a suffix. These prefixes form the conditional pattern base. The frequency of the prefixes can be read in the node of the item.

header table:

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

conditional pattern base:

<table>
<thead>
<tr>
<th>item</th>
<th>cond. pattern base</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>{}</td>
</tr>
<tr>
<td>c</td>
<td>f:3, {}</td>
</tr>
<tr>
<td>a</td>
<td>fc:3</td>
</tr>
<tr>
<td>m</td>
<td>fca:2, fcab:1</td>
</tr>
<tr>
<td>b</td>
<td>fca:1, f:1, c:1</td>
</tr>
<tr>
<td>p</td>
<td>fcam:2, cb:1</td>
</tr>
</tbody>
</table>

Frequent Itemset Mining ➔ Algorithms ➔ FP-Tree
Properties of FP-tree for Conditional Pattern Bases

• Node-link property
  – For any frequent item $a_i$, all the possible frequent patterns that contain $a_i$ can be obtained by following $a_i$'s node-links, starting from $a_i$'s head in the FP-tree header

• Prefix path property
  – To calculate the frequent patterns for a node $a_i$ in a path $P$, only the prefix sub-path of $a_i$ in $P$ needs to be accumulated, and its frequency count should carry the same count as node $a_i$. 
Major Steps to Mine FP-tree: Conditional FP-tree

1) Construct conditional pattern base for each node in the FP-tree

2) Construct conditional FP-tree from each conditional pattern-base

   - The prefix paths of a suffix represent the conditional basis. They can be regarded as transactions of a database.
   - Those prefix paths whose support $\geq \minSup$, induce a conditional FP-tree
   - For each pattern-base
     - Accumulate the count for each item in the base
     - Construct the FP-tree for the frequent items of the pattern base

---

**conditional pattern base:**

<table>
<thead>
<tr>
<th>item</th>
<th>cond. pattern base</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f$</td>
<td>${}$</td>
</tr>
<tr>
<td>$c$</td>
<td>$f:3$</td>
</tr>
<tr>
<td>$a$</td>
<td>$fc:3$</td>
</tr>
<tr>
<td>$b$</td>
<td>$fca:1$, $f:1$, $c:1$</td>
</tr>
<tr>
<td>$m$</td>
<td>$fca:2$, $fcab:1$</td>
</tr>
<tr>
<td>$p$</td>
<td>$fcam:2$, $cb:1$</td>
</tr>
</tbody>
</table>

**m-conditional FP-tree**

<table>
<thead>
<tr>
<th>item</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f$</td>
<td>3</td>
</tr>
<tr>
<td>$c$</td>
<td>3</td>
</tr>
<tr>
<td>$a$</td>
<td>3</td>
</tr>
<tr>
<td>$b$</td>
<td>1X</td>
</tr>
</tbody>
</table>

Frequent Itemset Mining $\rightarrow$ Algorithms $\rightarrow$ FP-Tree
Major Steps to Mine FP-tree: Conditional FP-tree

1) Construct conditional pattern base for each node in the FP-tree
2) Construct conditional FP-tree from each conditional pattern-base

```
conditional pattern base:

<table>
<thead>
<tr>
<th>item</th>
<th>cond. pattern base</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>{}</td>
</tr>
<tr>
<td>c</td>
<td>f:3</td>
</tr>
<tr>
<td>a</td>
<td>fc:3</td>
</tr>
<tr>
<td>b</td>
<td>fca:1, f:1, c:1</td>
</tr>
<tr>
<td>m</td>
<td>fca:2, fcab:1</td>
</tr>
<tr>
<td>p</td>
<td>fcam:2, cb:1</td>
</tr>
</tbody>
</table>
```

```
{} | f = {}
    |           |
        |           |
    f:3 |           |
    c:3 |           |

{} | c
    |               |
        |               |
    f:3 |               |
    c:3 |               |

{} | a
    |               |
        |               |
    f:3 |               |
    c:3 |               |

{} | b = {}
    |           |
        |           |
    f:3 |           |
    c:3 |           |
    a:3 |           |

{} | m
    |           |
        |           |
    f:3 |           |
    c:3 |           |

{} | p
    |           |
        |           |
    c:3 |           |
```
Major Steps to Mine FP-tree

1) Construct conditional pattern base for each node in the FP-tree
2) Construct conditional FP-tree from each conditional pattern-base
3) Recursively mine conditional FP-trees and grow frequent patterns obtained so far
   - If the conditional FP-tree contains a single path, simply enumerate all the patterns (enumerate all combinations of sub-paths)

**Example:**

```
  m-conditional FP-tree
  {} | m
    |   |
    | f:3 |
    |   |
    | c:3 |
    |   |
    | a:3
```

**All frequent patterns concerning m**

- m, fm, cm, am, fcm, fam, cam, fcam
FP-tree: Full Example

database:

<table>
<thead>
<tr>
<th>TID</th>
<th>items bought</th>
<th>(ordered) frequent items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>{b, c, f}</td>
<td>{f, b, c}</td>
</tr>
<tr>
<td>200</td>
<td>{a, b, c}</td>
<td>{b, c}</td>
</tr>
<tr>
<td>300</td>
<td>{d, f}</td>
<td>{f}</td>
</tr>
<tr>
<td>400</td>
<td>{b, c, e, f}</td>
<td>{f, b, c}</td>
</tr>
<tr>
<td>500</td>
<td>{f, g}</td>
<td>{f}</td>
</tr>
</tbody>
</table>

header table:

<table>
<thead>
<tr>
<th>item</th>
<th>frequency</th>
<th>head</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

minSup=0.4

conditional pattern base:

<table>
<thead>
<tr>
<th>item</th>
<th>cond. pattern base</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>{}</td>
</tr>
<tr>
<td>b</td>
<td>f:2, {}</td>
</tr>
<tr>
<td>c</td>
<td>fb:2, b:1</td>
</tr>
</tbody>
</table>
FP-tree: Full Example

### Conditional Pattern Base 1:

<table>
<thead>
<tr>
<th>Item</th>
<th>Cond. Pattern Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>{}</td>
</tr>
<tr>
<td>b</td>
<td>f:2</td>
</tr>
<tr>
<td>c</td>
<td>fb:2, b:1</td>
</tr>
</tbody>
</table>

### Conditional Pattern Base 2:

<table>
<thead>
<tr>
<th>Item</th>
<th>Cond. Pattern Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>f:2</td>
</tr>
<tr>
<td>f</td>
<td>{}</td>
</tr>
</tbody>
</table>

---

Frequent Itemset Mining → Algorithms → FP-Tree
Principles of Frequent Pattern Growth

• Pattern growth property
  – Let $\alpha$ be a frequent itemset in DB, $B$ be $\alpha$'s conditional pattern base, and $\beta$ be an itemset in $B$. Then $\alpha \cup \beta$ is a frequent itemset in DB iff $\beta$ is frequent in $B$.

• “abcdef” is a frequent pattern, if and only if
  – “abcde” is a frequent pattern, and
  – “f” is frequent in the set of transactions containing “abcde”
Why Is Frequent Pattern Growth Fast?

- Performance study in [Han, Pei&Yin ’00] shows
  - FP-growth is an order of magnitude faster than Apriori, and is also faster than tree-projection

- Reasoning
  - No candidate generation, no candidate test
    - Apriori algorithm has to proceed breadth-first
  - Use compact data structure
  - Eliminate repeated database scan
  - Basic operation is counting and FP-tree building

Data set T25I20D10K:
- T 25 avg. length of transactions
- I 20 avg. length of frequent itemsets
- D 10K database size (#transactions)
Maximal or Closed Frequent Itemsets

• Big challenge: database contains potentially a huge number of frequent itemsets (especially if minSup is set too low).
  – A frequent itemset of length 100 contains $2^{100} - 1$ many frequent subsets

• *Closed frequent itemset:*  
  An itemset $X$ is *closed* in a data set $D$ if there exists no proper super-itemset $Y$ such that $\text{support}(X) = \text{support}(Y)$ in $D$.  
  – The set of closed frequent itemsets contains complete information regarding its corresponding frequent itemsets.

• *Maximal frequent itemset:*  
  An itemset $X$ is *maximal* in a data set $D$ if there exists no proper super-itemset $Y$ such that $\text{support}(Y) \geq \text{minSup}$ in $D$.  
  – The set of maximal itemsets does not contain the complete support information  
  – More compact representation
Chapter 3: Frequent Itemset Mining

1) Introduction
   – Transaction databases, market basket data analysis

2) Mining Frequent Itemsets
   – Apriori algorithm, hash trees, FP-tree

3) Simple Association Rules
   – Basic notions, rule generation, interestingness measures

4) Further Topics
   – Hierarchical Association Rules
     • Motivation, notions, algorithms, interestingness
   – Quantitative Association Rules
     • Motivation, basic idea, partitioning numerical attributes, adaptation of apriori algorithm, interestingness

5) Extensions and Summary
Simple Association Rules: Introduction

- **Transaction database:**
  
  \[ D = \{\{\text{butter, bread, milk, sugar}\}; \]
  
  \{\{\text{butter, flour, milk, sugar}\}; \]
  
  \{\{\text{butter, eggs, milk, salt}\}; \]
  
  \{\{\text{eggs}\}; \]
  
  \{\{\text{butter, flour, milk, salt, sugar}\}\} \]

- **Frequent itemsets:**
  
<table>
<thead>
<tr>
<th>items</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>{\text{butter}}</td>
<td>4</td>
</tr>
<tr>
<td>{\text{milk}}</td>
<td>4</td>
</tr>
<tr>
<td>{\text{butter, milk}}</td>
<td>4</td>
</tr>
<tr>
<td>{\text{sugar}}</td>
<td>3</td>
</tr>
<tr>
<td>{\text{butter, sugar}}</td>
<td>3</td>
</tr>
<tr>
<td>{\text{milk, sugar}}</td>
<td>3</td>
</tr>
<tr>
<td>{\text{butter, milk, sugar}}</td>
<td>3</td>
</tr>
</tbody>
</table>

- **Question of interest:**
  
  - If milk and sugar are bought, will the customer always buy butter as well?  
    \[ \text{milk, sugar} \Rightarrow \text{butter} \] ?
  
  - In this case, what would be the probability of buying butter?
Simple Association Rules: Basic Notions

- **Items** $I = \{i_1, i_2, \ldots, i_m\}$: a set of literals (denoting items)
- **Itemset** $X$: Set of items $X \subseteq I$
- **Database** $D$: Set of transactions $T$, each transaction is a set of items $T \subseteq I$
- Transaction $T$ contains an itemset $X$: $X \subseteq T$
- The items in transactions and itemsets are sorted lexicographically:
  - itemset $X = (x_1, x_2, \ldots, x_k)$, where $x_1 \leq x_2 \leq \ldots \leq x_k$
- **Length** of an itemset: cardinality of the itemset ($k$-itemset: itemset of length $k$)
- The support of an itemset $X$ is defined as: $\text{support}(X) = |\{T \in D | X \subseteq T\}|$
- **Frequent itemset**: an itemset $X$ is called frequent iff $\text{support}(X) \geq \text{minSup}$
- **Association rule**: An association rule is an implication of the form $X \Rightarrow Y$ where $X, Y \subseteq I$ are two itemsets with $X \cap Y = \emptyset$.
- Note: simply enumerating all possible association rules is not reasonable! → What are the interesting association rules w.r.t. $D$?
Interestingness of Association Rules

- **Interestingness of an association rule:** Quantify the interestingness of an association rule with respect to a transaction database D:
  - Support: frequency (probability) of the entire rule with respect to D
    
    \[ \text{support}(X \Rightarrow Y) = P(X \cup Y) = \frac{|\{T \in D | X \cup Y \subseteq T\}|}{|D|} = \text{support}(X \cup Y) \]
    
    “probability that a transaction in D contains the itemset \( X \cup Y \)”
  - Confidence: indicates the strength of implication in the rule
    
    \[ \text{confidence}(X \Rightarrow Y) = P(Y | X) = \frac{|\{T \in D | X \cup Y \subseteq T\}|}{|\{T \in D | X \subseteq T\}|} = \frac{\text{support}(X \cup Y)}{\text{support}(X)} \]
    
    “conditional probability that a transaction in D containing the itemset \( X \) also contains itemset \( Y \)”
  - Rule form: “Body \( \Rightarrow \) Head [support, confidence]”

- **Association rule examples:**
  - buys diapers \( \Rightarrow \) buys beers [0.5%, 60%]
  - major in CS \( \land \) takes DB \( \Rightarrow \) avg. grade A [1%, 75%]
Task of mining association rules:
Given a database \( D \), determine all association rules having a support \( \geq minSup \) and a confidence \( \geq minConf \) (so-called strong association rules).

Key steps of mining association rules:
1) Find frequent itemsets, i.e., itemsets that have at least support = \( minSup \)
2) Use the frequent itemsets to generate association rules
   - For each itemset \( X \) and every nonempty subset \( Y \subset X \) generate rule \( Y \Rightarrow (X - Y) \) if \( minSup \) and \( minConf \) are fulfilled
   - we have \( 2^{|X|} - 2 \) many association rule candidates for each itemset \( X \)

Example
frequent itemsets

<table>
<thead>
<tr>
<th>1-itemset</th>
<th>count</th>
<th>2-itemset</th>
<th>count</th>
<th>3-itemset</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>( {A} )</td>
<td>3</td>
<td>( {A, B} )</td>
<td>3</td>
<td>( {A, B, C} )</td>
<td>2</td>
</tr>
<tr>
<td>( {B} )</td>
<td>4</td>
<td>( {A, C} )</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( {C} )</td>
<td>5</td>
<td>( {B, C} )</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

rule candidates: \( A \Rightarrow B; B \Rightarrow A; A \Rightarrow C; C \Rightarrow A; B \Rightarrow C; C \Rightarrow B; \)
\( A, B \Rightarrow C; A, C \Rightarrow B; C, B \Rightarrow A; A \Rightarrow B, C; B \Rightarrow A, C; C \Rightarrow A, B \)
Generating Rules from Frequent Itemsets

- For each frequent itemset \( X \)
  - For each nonempty subset \( Y \) of \( X \), form a rule \( Y \Rightarrow (X - Y) \)
  - Delete those rules that do not have minimum confidence

  Note: 1) support always exceeds \( \text{minSup} \)
          2) the support values of the frequent itemsets suffice to calculate the confidence

- Example: \( X = \{A, B, C\} \), \( \text{minConf} = 60\% \)
  - \( \text{conf} (A \Rightarrow B) = 3/3; \checkmark \)
  - \( \text{conf} (B \Rightarrow A) = 3/4; \checkmark \)
  - \( \text{conf} (A \Rightarrow C) = 2/3; \checkmark \)
  - \( \text{conf} (C \Rightarrow A) = 2/5; \times \)
  - \( \text{conf} (B \Rightarrow C) = 4/4; \checkmark \)
  - \( \text{conf} (C \Rightarrow B) = 4/5; \checkmark \)
  - \( \text{conf} (A \Rightarrow B, C) = 2/3; \checkmark \)
  - \( \text{conf} (B, C \Rightarrow A) = 1/2 \times \)
  - \( \text{conf} (B \Rightarrow A, C) = 2/4; \times \)
  - \( \text{conf} (A, C \Rightarrow B) = 1 \checkmark \)
  - \( \text{conf} (C \Rightarrow A, B) = 2/5; \times \)
  - \( \text{conf} (A, B \Rightarrow C) = 2/3 \checkmark \)

- Exploit anti-monotonicity for generating candidates for strong association rules!
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)
Example 1 [Aggarwal & Yu, PODS98]

- Among 5000 students
  - 3000 play basketball (=60%)
  - 3750 eat cereal (=75%)
  - 2000 both play basketball and eat cereal (=40%)
- Rule \( \text{play basketball} \Rightarrow \text{eat cereal} \) [40%, 66.7%] is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%
- Rule \( \text{play basketball} \Rightarrow \text{not eat cereal} \) [20%, 33.3%] is far more accurate, although with lower support and confidence
- Observation: \( \text{play basketball} \) and \( \text{eat cereal} \) are negatively correlated

- Not all strong association rules are interesting and some can be misleading.
  -augment the support and confidence values with interestingness measures such as the correlation \( A \Rightarrow B \) \([\text{supp}, \text{conf}, \text{corr}]\)
Other Interestingness Measures: Correlation

- **Lift** is a simple correlation measure between two items $A$ and $B$:

$$corr_{A,B} = \frac{P(A \cup B)}{P(A)P(B)} = \frac{P(B|A)}{P(B)} = \frac{\text{conf}(A \Rightarrow B)}{\text{supp}(B)}$$

*The two rules $A \Rightarrow B$ and $B \Rightarrow A$ have the same correlation coefficient.*

- take both $P(A)$ and $P(B)$ in consideration

- $corr_{A,B} > 1$ the two items $A$ and $B$ are positively correlated
- $corr_{A,B} = 1$ there is no correlation between the two items $A$ and $B$
- $corr_{A,B} < 1$ the two items $A$ and $B$ are negatively correlated
Other Interestingness Measures: Correlation

- Example 2:
  
  \[
  \begin{array}{cccccccc}
  X & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
  Y & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
  Z & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  \end{array}
  \]

- X and Y: positively correlated
- X and Z: negatively related
- support and confidence of X=>Z dominates
- but items X and Z are negatively correlated
- Items X and Y are positively correlated

<table>
<thead>
<tr>
<th>rule</th>
<th>support</th>
<th>confidence</th>
<th>correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>X ⇒ Y</td>
<td>25%</td>
<td>50%</td>
<td>2</td>
</tr>
<tr>
<td>X ⇒ Z</td>
<td>37.5%</td>
<td>75%</td>
<td>0.86</td>
</tr>
<tr>
<td>Y ⇒ Z</td>
<td>12.5%</td>
<td>50%</td>
<td>0.57</td>
</tr>
</tbody>
</table>
Chapter 3: Frequent Itemset Mining

1) Introduction
   - Transaction databases, market basket data analysis

2) Mining Frequent Itemsets
   - Apriori algorithm, hash trees, FP-tree

3) Simple Association Rules
   - Basic notions, rule generation, interestingness measures

4) Further Topics
   - Hierarchical Association Rules
     • Motivation, notions, algorithms, interestingness
   - Quantitative Association Rules
     • Motivation, basic idea, partitioning numerical attributes, adaptation of apriori algorithm, interestingness

5) Extensions and Summary
Hierarchical Association Rules: Motivation

- Problem of association rules in plain itemsets
  - *High minsup*: apriori finds only few rules
  - *Low minsup*: apriori finds unmanageably many rules
- Exploit item taxonomies (generalizations, *is-a* hierarchies) which exist in many applications

![Hierarchy Diagram]

- New task: find all generalized association rules between generalized items → Body and Head of a rule may have items of any level of the hierarchy
- **Generalized association rule**: \( X \Rightarrow Y \)
  with \( X, Y \subseteq I, X \cap Y = \emptyset \) and no item in \( Y \) is an ancestor of any item in \( X \)
  i.e., *jackets* ⇒ *clothes* is essentially true

*Frequent Itemset Mining → Further Topics → Hierarchical Association Rules*
Hierarchical Association Rules: Motivating Example

- Examples
  
  Jeans $\Rightarrow$ boots
  jackets $\Rightarrow$ boots $\big\{\text{Support} < \text{minSup}\}$
  Outerwear $\Rightarrow$ boots $\text{Support} > \text{minSup}$

- Characteristics
  
  - Support(“outerwear $\Rightarrow$ boots”) is not necessarily equal to the sum support(“jackets $\Rightarrow$ boots”) + support( “jeans $\Rightarrow$ boots”)
    e.g. if a transaction with jackets, jeans and boots exists
  
  - Support for sets of generalizations (e.g., product groups) is higher than support for sets of individual items
    If the support of rule “outerwear $\Rightarrow$ boots” exceeds minsup, then the support of rule “clothes $\Rightarrow$ boots” does, too
Mining Multi-Level Associations

- A top-down, progressive deepening approach:
  - First find high-level strong rules:
    - milk \( \Rightarrow \) bread [20%, 60%].
  - Then find their lower-level “weaker” rules:
    - 1.5% milk \( \Rightarrow \) wheat bread [6%, 50%].

- Different min_support threshold across multi-levels lead to different algorithms:
  - adopting the same min_support across multi-levels
  - adopting reduced min_support at lower levels
Minimum Support for Multiple Levels

• Uniform Support
  - milk
    - support = 10%
  - 3.5%
    - support = 6%
  - 1.5%
    - support = 4%
  + the search procedure is simplified (monotonicity)
  + the user is required to specify only one support threshold

• Reduced Support (Variable Support)
  - milk
    - support = 10%
  - 3.5%
    - support = 6%
  - 1.5%
    - support = 4%
  + takes the lower frequency of items in lower levels into consideration
Multilevel Association Mining using Reduced Support

- A *top-down, progressive deepening* approach:
  - First find high-level strong rules:
    - *milk ⇒ bread* [20%, 60%].
  - Then find their lower-level “weaker” rules:
    - 1.5% *milk ⇒ wheat bread* [6%, 50%].

  *level-wise processing (breadth first)*

3 approaches using reduced Support:

- **Level-by-level independent method:**
  - Examine each node in the hierarchy, regardless of whether or not its parent node is found to be frequent

- **Level-cross-filtering by single item:**
  - Examine a node only if its parent node at the preceding level is frequent

- **Level-cross-filtering by k-itemset:**
  - Examine a k-itemset at a given level only if its parent k-itemset at the preceding level is frequent
Multilevel Associations: Variants

• A *top-down, progressive deepening* approach:
  - First find high-level strong rules:
    • *milk* ⇒ *bread* [20%, 60%].
  - Then find their lower-level “weaker” rules:
    • 1.5% *milk* ⇒ *wheat bread* [6%, 50%].

`level-wise processing (breadth first)`

• Variations at mining multiple-level association rules.
  - Level-crossed association rules:
    • 1.5 % *milk* ⇒ *Wonder wheat bread*
  - Association rules with multiple, alternative hierarchies:
    • 1.5 % *milk* ⇒ *Wonder bread*
Some rules may be redundant due to “ancestor” relationships between items.

Example

- $R_1$: milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%]
- $R_2$: 1.5% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%]

We say that rule 1 is an ancestor of rule 2.

Redundancy:
A rule is redundant if its support is close to the “expected” value, based on the rule’s ancestor.
Interestingness of Hierarchical Association Rules: Notions

Let $X, X', Y, Y' \subseteq I$ be itemsets.

- An itemset $X'$ is an ancestor of $X$ iff there exist ancestors $x'_1, \ldots, x'_k$ of $x_1, \ldots, x_k \in X$ and $x_{k+1}, \ldots, x_n$ with $n = |X|$ such that
  \[ X' = \{x'_1, \ldots, x'_k, x_{k+1}, \ldots, x_n\}. \]

- Let $X'$ and $Y'$ be ancestors of $X$ and $Y$. Then we call the rules $X' \Rightarrow Y'$, $X \Rightarrow Y'$, and $X' \Rightarrow Y$ ancestors of the rule $X \Rightarrow Y$.

- The rule $X' \Rightarrow Y'$ is a direct ancestor of rule $X \Rightarrow Y$ in a set of rules if:
  - Rule $X' \Rightarrow Y'$ is an ancestor of rule $X \Rightarrow Y$, and
  - There is no rule $X'' \Rightarrow Y''$ such that $X'' \Rightarrow Y''$ is an ancestor of $X \Rightarrow Y$ and $X' \Rightarrow Y'$ is an ancestor of $X'' \Rightarrow Y''$.

- A hierarchical association rule $X \Rightarrow Y$ is called $R$-interesting if:
  - There are no direct ancestors of $X \Rightarrow Y$ or
  - The actual support is larger than $R$ times the expected support or
  - The actual confidence is larger than $R$ times the expected confidence.
Expected Support and Expected Confidence

• How to compute the expected support? Given the rule for $X \Rightarrow Y$ and its ancestor rule $X' \Rightarrow Y'$ the expected support of $X \Rightarrow Y$ is defined as:

$$E_{Z'}[P(Z)] = \frac{P(z_1)}{P(z'_1)} \times \cdots \times \frac{P(z_j)}{P(z'_j)} \times P(Z')$$

where $Z = X \cup Y = \{z_1, ..., z_n\}$, $Z' = X' \cup Y' = \{z'_1, ..., z'_j, z_{j+1}, ..., z_n\}$ and each $z'_i \in Z'$ is an ancestor of $z_i \in Z$

• How to compute the expected confidence?
  Given the rule for $X \implies Y$ and its ancestor rule $X' \implies Y'$, then the expected confidence of $X \implies Y$ is defined as:

$$E_{X \Rightarrow Y}[P(Y|X)] = \frac{P(y_1)}{P(y'_1)} \times \cdots \times \frac{P(y_j)}{P(y'_j)} \times P(Y'|X')$$

where $Y = \{y_1, \ldots, y_n\}$ and $Y' = \{y'_1, \ldots, y'_j, y_{j+1}, \ldots, y_n\}$ and each $y'_i \in Y'$ is an ancestor of $y_i \in Y$

### Interestingness of Hierarchical Association Rules: Example

- **Example**
  - Let $R = 1.6$

<table>
<thead>
<tr>
<th>Item</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>clothes</td>
<td>20</td>
</tr>
<tr>
<td>outerwear</td>
<td>10</td>
</tr>
<tr>
<td>jackets</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No</th>
<th>rule</th>
<th>support</th>
<th>R-interesting?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>clothes $\Rightarrow$ shoes</td>
<td>10</td>
<td>yes: no ancestors</td>
</tr>
<tr>
<td>2</td>
<td>outerwear $\Rightarrow$ shoes</td>
<td>9</td>
<td>yes: Support $&gt; R \cdot \text{exp. support (wrt. rule 1)} = 8$</td>
</tr>
</tbody>
</table>
| 3  | jackets $\Rightarrow$ shoes   | 4       | Not wrt. support:  
Support $> R \cdot \text{exp. support (wrt. rule 1)} = 3.2$
Support $< R \cdot \text{exp. support (wrt. rule 2)} = 5.75$
$\Rightarrow$ still need to check the confidence! |

- Frequent Itemset Mining ➔ Further Topics ➔ Hierarchical Association Rules
Chapter 3: Frequent Itemset Mining

1) Introduction
   – Transaction databases, market basket data analysis

2) Simple Association Rules
   – Basic notions, rule generation, interestingness measures

3) Mining Frequent Itemsets
   – Apriori algorithm, hash trees, FP-tree

4) Further Topics
   – Hierarchical Association Rules
     • Motivation, notions, algorithms, interestingness
   – Multidimensional and Quantitative Association Rules
     • Motivation, basic idea, partitioning numerical attributes, adaptation of apriori algorithm, interestingness

5) Summary
Multi-Dimensional Association: Concepts

- Single-dimensional rules:
  - buys milk \( \Rightarrow \) buys bread

- Multi-dimensional rules: \( \geq 2 \) dimensions
  - Inter-dimension association rules (*no repeated dimensions*)
    - age between 19-25 \( \land \) status is student \( \Rightarrow \) buys coke
  - hybrid-dimension association rules (*repeated dimensions*)
    - age between 19-25 \( \land \) buys popcorn \( \Rightarrow \) buys coke
Techniques for Mining Multi-Dimensional Associations

- Search for frequent $k$-predicate set:
  - Example: $\{\text{age}, \text{occupation}, \text{buys}\}$ is a 3-predicate set.
  - Techniques can be categorized by how $\text{age}$ is 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.
Quantitative Association Rules

• Up to now: associations of *boolean* attributes only
• Now: *numerical* attributes, too
• Example:
  – Original database
  
<table>
<thead>
<tr>
<th>ID</th>
<th>age</th>
<th>marital status</th>
<th># cars</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23</td>
<td>single</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>38</td>
<td>married</td>
<td>2</td>
</tr>
</tbody>
</table>

  – Boolean database

<table>
<thead>
<tr>
<th>ID</th>
<th>age: 20..29</th>
<th>age: 30..39</th>
<th>m-status: single</th>
<th>m-status: married</th>
<th>. . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>. . .</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>. . .</td>
</tr>
</tbody>
</table>
Quantitative Association Rules: Ideas

• Static discretization
  – Discretization of all attributes *before* mining the association rules
  – E.g. by using a generalization hierarchy for each attribute
  – Substitute numerical attribute values by ranges or intervals

• Dynamic discretization
  – Discretization of the attributes *during* association rule mining
  – Goal (e.g.): maximization of confidence
  – Unification of neighboring association rules to a generalized rule
Partitioning of Numerical Attributes

- Problem: Minimum support
  - Too many intervals → too small support for each individual interval
  - Too few intervals → too small confidence of the rules

- Solution
  - First, partition the domain into many intervals
  - Afterwards, create new intervals by merging adjacent interval

- Numeric attributes are *dynamically* discretized such that the confidence or compactness of the rules mined is maximized.
Quantitative Association Rules

- 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:

\[
\text{age}(X,\text{"30-34"}) \land \text{income}(X,\text{"24K - 48K"}) \Rightarrow \text{buys}(X,\text{"high resolution TV"})
\]
Chapter 3: Frequent Itemset Mining

1) Introduction
   - Transaction databases, market basket data analysis

2) Mining Frequent Itemsets
   - Apriori algorithm, hash trees, FP-tree

3) Simple Association Rules
   - Basic notions, rule generation, interestingness measures

4) Further Topics
   - Hierarchical Association Rules
     • Motivation, notions, algorithms, interestingness
   - Quantitative Association Rules
     • Motivation, basic idea, partitioning numerical attributes, adaptation of apriori algorithm, interestingness

5) Summary
Chapter 3: Summary

- Mining frequent itemsets
  - Apriori algorithm, hash trees, FP-tree
- Simple association rules
  - support, confidence, rule generation, interestingness measures (correlation), ...
- Further topics
  - Hierarchical association rules: algorithms (top-down progressive deepening), multilevel support thresholds, redundancy and R-interestingness
  - Quantitative association rules: partitioning numerical attributes, adaptation of apriori algorithm, interestingness
- Extensions: multi-dimensional association rule mining