Chapter 6: Mining Frequent Patterns, Association and Correlations

- **Basic concepts**
- Frequent itemset mining methods
- Constraint-based frequent pattern mining (ch7)
- Association rules
What Is Frequent Pattern Analysis?

- **Frequent pattern**: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set

- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining

- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?

- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.
Why Is Freq. Pattern Mining Important?

- Freq. pattern: intrinsic and important property of data sets
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: associative classification
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression: fascicles
- Broad applications
Basic Concepts: Frequent Patterns

- **itemset**: A set of items
- **k-itemset** \( X = \{x_1, \ldots, x_k\} \)
- **(absolute) support**, or, **support count** of \( X \): Frequency or occurrence of an itemset \( X \)
- **(relative) support**, \( s \), is the fraction of transactions that contains \( X \) (i.e., the probability that a transaction contains \( X \))
- An itemset \( X \) is **frequent** if \( X \)'s support is no less than a **\textit{minsup}** threshold

<table>
<thead>
<tr>
<th>Tid</th>
<th>Items bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Beer, Nuts, Diaper</td>
</tr>
<tr>
<td>20</td>
<td>Beer, Coffee, Diaper</td>
</tr>
<tr>
<td>30</td>
<td>Beer, Diaper, Eggs</td>
</tr>
<tr>
<td>40</td>
<td>Nuts, Eggs, Milk</td>
</tr>
<tr>
<td>50</td>
<td>Nuts, Coffee, Diaper, Eggs, Milk</td>
</tr>
</tbody>
</table>
Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., \{a_1, ..., a_{100}\} contains $2^{100} - 1 = 1.27 \times 10^{30}$ sub-patterns!
- Solution: *Mine closed patterns and max-patterns instead*

An itemset $X$ is a **closed pattern** if $X$ is frequent and there exist *no super-patterns with the same support*
  - all super-patterns must have smaller support

An itemset $X$ is a **max-pattern** if $X$ is frequent and there exist no super-patterns that are frequent

Relationship between the two?

Closed patterns are a lossless compression of freq. patterns, whereas max-patterns are a lossy compression
  - Lossless: can derive all frequent patterns as well as their support
  - Lossy: can derive all frequent patterns
Closed Patterns and Max-Patterns

- DB = \{<a_1, ..., a_{100}>, <a_1, ..., a_{50}>\}
  - min_sup = 1
- What is the set of closed patterns?
  - \(<a_1, ..., a_{100}>\): 1
  - \(<a_1, ..., a_{50}>\): 2
  - How to derive frequent patterns and their support values?
- What is the set of max-patterns?
  - \(<a_1, ..., a_{100}>\): 1
  - How to derive frequent patterns?
- What is the set of all patterns?
  - \(\{a_1\}: 2, ..., \{a_1, a_2\}: 2, ..., \{a_1, a_{51}\}: 1, ..., \{a_1, a_2, ..., a_{100}\}: 1\)
  - A big number: \(2^{100} - 1\)
Closed Patterns and Max-Patterns

For a given dataset with itemset I = \{a,b,c,d\} and min_sup = 8, the closed patterns are \{a,b,c,d\} with support of 10, \{a,b,c\} with support of 12, and \{a,b,d\} with support of 14. Derive the frequent 2-itemsets together with their support values

\{a,b\}: 14 \quad \{a,c\}: 12 \quad \{a,d\}: 14
\{b,c\}: 12 \quad \{b,d\}: 14 \quad \{c,d\}: 10
Chapter 6: Mining Frequent Patterns, Association and Correlations

- Basic concepts
- **Frequent itemset mining methods**
- Constraint-based frequent pattern mining (ch7)
- Association rules
Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach

- Improving the Efficiency of Apriori

- FP-Growth: A Frequent Pattern-Growth Approach

- ECLAT: Frequent Pattern Mining with Vertical Data Format
Scalable Methods for Mining Frequent Patterns

- The **downward closure** (anti-monotonic) property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If \{beer, diaper, nuts\} is frequent, so is \{beer, diaper\}
  - i.e., every transaction having \{beer, diaper, nuts\} also contains \{beer, diaper\}

- Scalable mining methods: Three major approaches
  - **Apriori** (Agrawal & Srikant@VLDB’94)
    - Freq. pattern growth (Fpgrowth: Han, Pei & Yin @SIGMOD’00)
    - Vertical data format (Charm—Zaki & Hsiao @SDM’02)
Apriori: A Candidate Generation-and-Test Approach

- **Apriori pruning principle**: If there is any itemset that is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @ VLDB’ 94, Mannila, et al. @ KDD’ 94)

- **Method**:

  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated
The Apriori Algorithm—An Example

DB

<table>
<thead>
<tr>
<th>Tid</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, c, d</td>
</tr>
<tr>
<td>20</td>
<td>b, c, e</td>
</tr>
<tr>
<td>30</td>
<td>a, b, c, e</td>
</tr>
<tr>
<td>40</td>
<td>b, e</td>
</tr>
</tbody>
</table>

1st scan

$C_1$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a}</td>
<td>2</td>
</tr>
<tr>
<td>{b}</td>
<td>3</td>
</tr>
<tr>
<td>{c}</td>
<td>3</td>
</tr>
<tr>
<td>{d}</td>
<td>1</td>
</tr>
<tr>
<td>{e}</td>
<td>3</td>
</tr>
</tbody>
</table>

$L_1$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a}</td>
<td>2</td>
</tr>
<tr>
<td>{b}</td>
<td>3</td>
</tr>
<tr>
<td>{c}</td>
<td>3</td>
</tr>
<tr>
<td>{e}</td>
<td>3</td>
</tr>
</tbody>
</table>

2nd scan

$L_2$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a, b}</td>
<td>1</td>
</tr>
<tr>
<td>{a, c}</td>
<td>2</td>
</tr>
<tr>
<td>{a, e}</td>
<td>1</td>
</tr>
<tr>
<td>{b, c}</td>
<td>2</td>
</tr>
<tr>
<td>{b, e}</td>
<td>3</td>
</tr>
<tr>
<td>{c, e}</td>
<td>2</td>
</tr>
</tbody>
</table>

$L_3$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{b, c, e}</td>
<td>2</td>
</tr>
</tbody>
</table>

3rd scan

$L_3$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{b, c, e}</td>
<td>2</td>
</tr>
</tbody>
</table>

min_sup = 2
The Apriori Algorithm (Pseudo-code)

$C_k$: Candidate itemset of size $k$
$L_k$: frequent itemset of size $k$

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

\begin{align*}
\text{for } (k = 1; & \ L_k \neq \emptyset; \ k++) \text{ do begin} \\
\quad & C_{k+1} = \text{candidates generated from } L_k; \\
\quad & \text{for each transaction } t \text{ in database do} \\
\quad & \quad \text{increment the count of all candidates in } C_{k+1} \\
\quad & \quad \text{that are contained in } t \\
\quad & L_{k+1} = \text{candidates in } C_{k+1} \text{ with min\_support} \\
\text{end} \\
\text{return } \bigcup_k L_k;
\end{align*}
Implementation of Apriori

- Generate candidates, then count support for the generated candidates
- How to generate candidates?
  - Step 1: self-joining $L_k$
  - Step 2: pruning
- Example:
  - $L_3=\{abc, abd, acd, ace, bcd\}$
  - 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 $L_3$
  - $C_4=\{abcd\}$
- The above procedures do not miss any legitimate candidates. Thus Apriori mines a complete set of frequent patterns.
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: Counting Supports of Candidates

Subset function

1,4,7
2,5,8
3,6,9

Transaction: 1 2 3 5 6
Further Improvement of the Apriori Method

- Major computational challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates
Apriori applications beyond freq. pattern mining

- Given a set $S$ of students, we want to find each subset of $S$ such that the age range of the subset is less than 5.
  - Apriori algorithm, level-wise search using the downward closure property for pruning to gain efficiency

- Can be used to search for any subsets with the downward closure property (i.e., anti-monotone constraint)

- CLIQUE for subspace clustering used the same Apriori principle, where the one-dimensional cells are the items
Chapter 6: Mining Frequent Patterns, Association and Correlations

- Basic concepts
- Frequent itemset mining methods
- Constraint-based frequent pattern mining (ch7)
- Association rules
Constraint-based (Query-Directed) Mining

- Finding all the patterns in a database **autonomously**? — unrealistic!
  - The patterns could be too many but not focused!
- Data mining should be an **interactive** process
  - User directs what to be mined using a **data mining query language** (or a graphical user interface)
- Constraint-based mining
  - User flexibility: provides **constraints** on what to be mined
  - Optimization: explores such constraints for efficient mining — **constraint-based mining**: constraint-pushing, similar to push selection first in DB query processing
  - Note: still find all the answers satisfying constraints, not finding some answers in “heuristic search”
Constrained Mining vs. Constraint-Based Search

- Constrained mining vs. constraint-based search/reasoning
  - Both are aimed at reducing search space
  - Finding all patterns satisfying constraints vs. finding some (or one) answer in constraint-based search in AI
  - Constraint-pushing vs. heuristic search
  - It is an interesting research problem on how to integrate them

- Constrained mining vs. query processing in DBMS
  - Database query processing requires to find all
  - Constrained pattern mining shares a similar philosophy as pushing selections deeply in query processing
Constraint-Based Frequent Pattern Mining

- Pattern space pruning constraints
  - **Anti-monotonic**: If constraint $c$ is violated, its further mining can be terminated
  - **Monotonic**: If $c$ is satisfied, no need to check $c$ again

- **Succinct**: $c$ must be satisfied, so one can start with the data sets satisfying $c$
- **Convertible**: $c$ is not monotonic nor anti-monotonic, but it can be converted into it if items in the transaction can be properly ordered

- Data space pruning constraint
  - **Data succinct**: Data space can be pruned at the initial pattern mining process
  - **Data anti-monotonic**: If a transaction $t$ does not satisfy $c$, $t$ can be pruned from its further mining
Anti-Monotonicity in Constraint Pushing

- Anti-monotonicity
  - When an itemset $S$ violates the constraint, so does any of its superset
  - $\text{sum}(S.\text{Price}) \leq v$ is anti-monotonic
  - $\text{sum}(S.\text{Price}) \geq v$ is not anti-monotonic
  - C: range($S.\text{profit}$) $\leq 15$ is anti-monotonic
    - Itemset $ab$ violates C
    - So does every superset of $ab$
  - support count $\geq \text{min\_sup}$ is anti-monotonic
    - core property used in Apriori

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, b, c, d, f</td>
</tr>
<tr>
<td>20</td>
<td>b, c, d, f, g, h</td>
</tr>
<tr>
<td>30</td>
<td>a, c, d, e, f</td>
</tr>
<tr>
<td>40</td>
<td>c, e, f, g</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>-10</td>
</tr>
</tbody>
</table>
Monotonicity for Constraint Pushing

- Monotonicity
  - *When an itemset S satisfies the constraint, so does any of its superset*
  - \( \text{sum}(S.Price) \geq v \) is monotonic
  - \( \text{min}(S.Price) \leq v \) is monotonic
- C: range(S.profit) \( \geq 15 \)
  - Itemset \( ab \) satisfies C
  - So does every superset of \( ab \)

<table>
<thead>
<tr>
<th>Item</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>-10</td>
</tr>
</tbody>
</table>
Succinctness

- Given $A_1$, the set of items satisfying a succinctness constraint $C$, then any set $S$ satisfying $C$ is based on $A_1$, i.e., $S$ contains a subset belonging to $A_1$.

- Idea: Without looking at the transaction database, whether an itemset $S$ satisfies constraint $C$ can be determined based on the selection of items.

- If a constraint is succinct, we can directly generate precisely the sets that satisfy it, even before support counting begins.

- Avoids substantial overhead of generate-and-test, i.e., such constraint is pre-counting pushable.

- $\text{min}(S.Price) \leq \nu$ is succinct

- $\text{sum}(S.Price) \geq \nu$ is not succinct.
## Constraint-Based Mining—A General Picture

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Antimonotone</th>
<th>Monotone</th>
<th>Succinct</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v \in S$</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>$S \supseteq V$</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>$S \subseteq V$</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>$\min(S) \leq v$</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>$\min(S) \geq v$</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>$\max(S) \leq v$</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>$\max(S) \geq v$</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>$\text{count}(S) \leq v$</td>
<td>yes</td>
<td>no</td>
<td>weakly</td>
</tr>
<tr>
<td>$\text{count}(S) \geq v$</td>
<td>no</td>
<td>yes</td>
<td>weakly</td>
</tr>
<tr>
<td>$\text{sum}(S) \leq v$ ( $a \in S, a \geq 0$ )</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>$\text{sum}(S) \geq v$ ( $a \in S, a \geq 0$ )</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>$\text{range}(S) \leq v$</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>$\text{range}(S) \geq v$</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>$\text{avg}(S) \theta v, \theta \in { =, \leq, \geq }$ convertible</td>
<td>convertible</td>
<td>convertible</td>
<td>no</td>
</tr>
<tr>
<td>$\text{support}(S) \geq \xi$</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>$\text{support}(S) \leq \xi$</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
Chapter 6: Mining Frequent Patterns, Association and Correlations

- Basic concepts
- Frequent itemset mining methods
- Constraint-based frequent pattern mining (ch7)
- Association rules
Basic Concepts: Association Rules

- An association rule is of the form $X \rightarrow Y$, where $X, Y \subseteq I$, $X \cap Y = \emptyset$
  - A rule is **strong** if it satisfies both support and confidence thresholds.
- **support**($X \rightarrow Y$): probability that a transaction contains $X \cup Y$, i.e.,
  $$\text{support}(X \rightarrow Y) = P(X \cup Y)$$
  Can be estimated by the percentage of transactions in DB that contain $X \cup Y$. Not to be confused with $P(X \text{ or } Y)$
- **confidence**($X \rightarrow Y$): conditional probability that a transaction having $X$ also contains $Y$, i.e.
  $$\text{confidence}(X \rightarrow Y) = P(Y|X)$$
  $$\text{confidence}(X \rightarrow Y) = P(Y|X) = \frac{\text{support}(X \cup Y)}{\text{support}(X)} = \frac{\text{support}_\text{count}(X \cup Y)}{\text{support}_\text{count}(X)}$$
- confidence($X \rightarrow Y$) can be easily derived from the support count of $X$ and the support count of $X \cup Y$. Thus association rule mining can be reduced to frequent pattern mining
Basic Concepts: Association rules

<table>
<thead>
<tr>
<th>Tid</th>
<th>Items bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Beer, Nuts, Diaper</td>
</tr>
<tr>
<td>20</td>
<td>Beer, Coffee, Diaper</td>
</tr>
<tr>
<td>30</td>
<td>Beer, Diaper, Eggs</td>
</tr>
<tr>
<td>40</td>
<td>Nuts, Eggs, Milk</td>
</tr>
<tr>
<td>50</td>
<td>Nuts, Coffee, Diaper, Eggs, Milk</td>
</tr>
</tbody>
</table>

Let \( \text{minsup} = 50\% \), \( \text{minconf} = 50\% \)

Freq. Pat.: Beer:3, Nuts:3, Diaper:4, Eggs:3, \{Beer, Diaper\}:3

- Association rules: (many more!)
  - \( \text{Beer} \rightarrow \text{Diaper} \) (60\%, 100\%)
  - \( \text{Diaper} \rightarrow \text{Beer} \) (60\%, 75\%)

If \{a\} \rightarrow \{b\} is an association rule, then \{b\} \rightarrow \{a\} is also an association rule?
- Same support, different confidence

If \{a,b\} \rightarrow \{c\} is an association rule, then \{b\} \rightarrow \{c\} is also an association rule?

If \{b\} \rightarrow \{c\} is an association rule then \{a,b\} \rightarrow \{c\} is also an association rule?
Interestingness Measure: Correlations (Lift)

- *play basketball* ⇒ *eat cereal* [40%, 66.7%] is misleading
  - The overall % of students eating cereal is 75% > 66.7%.
- *play basketball* ⇒ *not eat cereal* [20%, 33.3%] is more accurate, although with lower support and confidence
- Support and confidence are not good to indicate correlations

Measure of dependent/correlated events: *lift*

$$\text{lift} = \frac{P(A \cup B)}{P(A)P(B)}$$

$$\text{lift}(B,C) = \frac{2000 / 5000}{3000 / 5000 \times 3750 / 5000} = 0.89$$

$$\text{lift}(B, \neg C) = \frac{1000 / 5000}{3000 / 5000 \times 1250 / 5000} = 1.33$$