Chapter 5

Mining Frequent Patterns, Associations, and Correlations

頻繁樣式, 關聯, 相互關係的探勘

Outlines

1. Basic concepts and a road map
2. Efficient and scalable frequent itemset mining methods
3. Mining various kinds of association rules
4. From association mining to correlation analysis
5. Constraint-based association mining
6. Summary
1. **Basic concepts and a road map**

2. Efficient and scalable frequent itemset mining methods

3. Mining various kinds of association rules

4. From association mining to correlation analysis

5. Constraint-based association mining

6. Summary

---

**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?

- Discloses an intrinsic and important property of data sets
- Forms the 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

5.1.1 Market Basket Analysis (購物籃分析):
A Motivating Example
If we think of the universe as the set of items available at the store, then each item has a Boolean variable representing the presence or absence of that item.

Each basket can then be represented by a Boolean vector of values assigned to these variables.

The Boolean vectors can be analyzed for buying patterns that reflect items that are frequently associated or purchased together.

These patterns can be represented in the form of association rules. For example, the information that customers who purchase computers also tend to buy antivirus software at the same time is represented in

Association Rule (5.1) below:

\[
\text{computer } \Rightarrow \text{antivirus software } \quad \text{[support} = 2\% , \text{confidence} = 60\% \text{]} \quad (5.1)
\]

Rule support and confidence are two measures of rule interestingness. They respectively reflect the usefulness and certainty of discovered rules.

A support of 2\% for Association Rule (5.1) means that 2\% of all the transactions under analysis show that computer and antivirus software are purchased together (一起購買).

A confidence of 60\% means that 60\% of the customers who purchased a computer (先決條件是要買電腦) also bought the software.

Typically, association rules are considered interesting if they satisfy both a minimum support threshold (最小支持度) and a minimum confidence threshold (最小信賴度). Such thresholds can be set by users or domain experts.

Additional analysis can be performed to uncover interesting statistical correlations between associated items.
5.1.2 Frequent Itemsets, Closed Itemsets, and Association Rules

Let \( I = \{I_1, I_2, \ldots, I_n\} \) be a set of items. Let \( D \), the task-relevant data, be a set of database transactions where each transaction \( T \) is a set of items such that \( T \subseteq I \). Each transaction is associated with an identifier, called TID. Let \( A \) be a set of items. A transaction \( T \) is said to contain \( A \) if and only if \( A \subseteq T \). An association rule is an implication of the form \( A \Rightarrow B \), where \( A \subseteq I, B \subseteq I \), and \( A \cap B = \emptyset \). The rule \( A \Rightarrow B \) holds in the transaction set \( D \) with support \( s \), where \( s \) is the percentage of transactions in \( D \) that contain \( A \cup B \) (i.e., the union of sets \( A \) and \( B \), or say, both \( A \) and \( B \)). This is taken to be the probability, \( P(A \cup B) \). The rule \( A \Rightarrow B \) has confidence \( c \) in the transaction set \( D \), where \( c \) is the percentage of transactions in \( D \) containing \( A \) that also contain \( B \). This is taken to be the conditional probability, \( P(B|A) \). That is,

\[
\begin{align*}
\text{support}(A \Rightarrow B) &= P(A \cup B) \\
\text{confidence}(A \Rightarrow B) &= P(B|A).
\end{align*}
\]

\[\text{confidence}(A \Rightarrow B) = P(B|A) = \frac{\text{support}(A \cup B)}{\text{support}(A)} = \frac{\text{support}_{\text{count}}(A \cup B)}{\text{support}_{\text{count}}(A)}. \quad (5.4)\]

- A set of items is referred to as an itemset.
- An itemset that contains \( k \) items is a k-itemset.
  - i.e., The set \{computer, antivirus software\} is a 2-itemset.
- The occurrence frequency of an itemset is the number of transactions that contain the itemset. This is also known, simply, as the frequency, support count, or count of the itemset.
- Note that the itemset support defined in Equation (5.2) is sometimes referred to as relative support (相對支持度), whereas the occurrence frequency is called the absolute support (絕對支持度).
- If the relative support of an itemset \( I \) satisfies a prespecified minimum support threshold (i.e., the absolute support of \( I \) satisfies the corresponding minimum support count threshold), then \( I \) is a frequent itemset (頻繁項目集).
- The set of frequent k-itemsets is commonly denoted by \( L_k \).
Example:

<table>
<thead>
<tr>
<th>Transaction-id</th>
<th>Items bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A, B, D</td>
</tr>
<tr>
<td>20</td>
<td>A, C, D</td>
</tr>
<tr>
<td>30</td>
<td>A, D, E</td>
</tr>
<tr>
<td>40</td>
<td>B, E, F</td>
</tr>
<tr>
<td>50</td>
<td>B, C, D, E, F</td>
</tr>
</tbody>
</table>

Itemset $X = \{x_1, \ldots, x_k\}$

Find all the rules $X \rightarrow Y$ with minimum support and confidence

- **support**, $s$, probability that a transaction contains $X \cup Y$
- **confidence**, $c$, conditional probability that a transaction having $X$ also contains $Y$

Let $\text{sup}_{\text{min}} = 50\%$, $\text{conf}_{\text{min}} = 50\%$

Freq. Pat.: \{A:3, B:3, D:4, E:3, AD:3\}
Association rules:
- $A \rightarrow D$ (60%, 100%)
- $D \rightarrow A$ (60%, 75%)

Customer buys diaper
Customer buys both
Customer buys beer

association rule mining can be viewed as a two-step process:

1. **Find all frequent itemsets**
   (找出所有的頻繁項目集):
   By definition, each of these itemsets will occur at least as frequently as a predetermined **minimum support count**, min sup.

2. **Generate strong association rules from the frequent itemsets**
   (從頻繁項目集產生強關聯規則):
   By definition, these rules must satisfy minimum support and minimum confidence.
Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., \{a_1, \ldots, a_{100}\} contains:
  \[
  \binom{100}{1} + \binom{100}{2} + \cdots + \binom{100}{100} - 2^{100} - 1 \approx 1.27 \times 10^{30}
  \]

- Solution: *Mine closed patterns and max-patterns instead*
- An itemset \(X\) is **closed** if \(X\) is frequent and there exists no super-pattern \(Y\) has the same support count as \(X\) in \(S\).
- An itemset \(X\) is a **closed frequent itemset** (緊密頻繁項目集) in set \(S\) if \(X\) is both closed and frequent in \(S\).
- An itemset \(X\) is a **maximal frequent itemset** (最大頻繁項目集) if \(X\) is frequent and there exists no frequent super-pattern \(Y\) s.t. \(X \subset Y\) and \(Y\) is frequent in \(S\).
- Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

Example 5.2 Closed Patterns and Max-Patterns

- 資料庫有二筆交易記錄，
  \[ DB = \{<a_1, \ldots, a_{100}>, <a_1, \ldots, a_{50}>\} \]
  - 假設 \(\text{Min}_\text{sup} = 1\).
- What is the set of closed frequent itemset?
  - \(\text{C} = \{\{a_1, a_2, \ldots, a_{100}\}: 1\} : \{a_1, \ldots, a_{50}\}: 2\}
- What is the set of maximal frequent itemset?
  - \(\text{M} = \{\{a_1, \ldots, a_{100}\}: 1\}
- What is the set of all patterns?
  - !!
5.1.3 Frequent Pattern Mining: A Road Map

1. Based on the *completeness* of patterns to be mined
2. Based on the *levels of abstraction* involved in the rule set
3. Based on the *number of data dimensions* involved in the rule

\[ \text{buys}(X, \text{"computer"}) \Rightarrow \text{buys}(X, \text{"antivirus\_software"}) \]  

(5.8)  

\[ \text{age}(X, \text{"30…39"}) \land \text{income}(X, \text{"42K…48K"}) \Rightarrow \text{buys}(X, \text{"high\_resolution\_TV"}). \]  

(5.9)

4. Based on the *types of values* handled in the rule
5. Based on the kinds of *rules* to be mined
6. Based on the *kinds of patterns* to be mined

---

1. Basic concepts and a road map

2. **Efficient and scalable frequent itemset mining methods**

3. Mining various kinds of association rules
4. From association mining to correlation analysis
5. Constraint-based association mining
6. Summary
5.2.1 The Apriori Algorithm: Finding Frequent Itemsets Using Candidate Generation (利用候選項目集尋找頻繁項目集)

- Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules.

- The name of the algorithm is based on the fact that the algorithm uses *prior knowledge* of frequent itemset properties.

- Concept:
  1. An iterative approach known as a level-wise search (逐層搜尋法)
  2. k-itemsets are used to explore (k+1)-itemsets.

- Method:
  1. Initially, scan DB once to get frequent 1-itemset \((L_1)\) and satisfy minimum support.
  2. Generate length (k+1) candidate itemsets from length k frequent itemsets \(L_1 \rightarrow L_2 \rightarrow L_3 \rightarrow \ldots\)
  3. Test the candidates against DB
  4. Terminate when no frequent or candidate set can be generated
Apriori property

- All nonempty subsets of a frequent itemset must also be frequent.
  - If an itemset \( I \) does not satisfy the minimum support threshold, \( \text{min\_sup} \), then \( I \) is not frequent; that is, \( P(I) < \text{min\_sup} \).
  - If an item \( A \) is added to the itemset \( I \), then the resulting itemset (i.e., \( I \cup A \)) cannot occur more frequently than \( I \).
  - \( I \cup A \) is not frequent and \( P(I \cup A) < \text{min\_sup} \).
- Antimonotone (反一致性):
  If a set cannot pass a test, all of its supersets (超集合) will fail the same test as well.

---

Example 5.3

- The AllElectronics transaction database, \( D \), of Table 5.1.
- There are nine transactions in this database, that is, \( |D| = 9 \).
- We use Figure 5.2 to illustrate the Apriori algorithm for finding frequent itemsets in \( D \).

<table>
<thead>
<tr>
<th>TID</th>
<th>List of item IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>T100</td>
<td>i1, i2, i5</td>
</tr>
<tr>
<td>T200</td>
<td>i2, i4</td>
</tr>
<tr>
<td>T300</td>
<td>i2, i3</td>
</tr>
<tr>
<td>T400</td>
<td>i1, i2, i4</td>
</tr>
<tr>
<td>T500</td>
<td>i1, i3</td>
</tr>
<tr>
<td>T600</td>
<td>i2, i3</td>
</tr>
<tr>
<td>T700</td>
<td>i1, i3</td>
</tr>
<tr>
<td>T800</td>
<td>i1, i2, i3, i5</td>
</tr>
<tr>
<td>T900</td>
<td>i1, i2, i3</td>
</tr>
</tbody>
</table>
In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemsets, $C_1$. The algorithm simply scans all of the transactions in order to count the number of occurrences of each item.

**Step 1**

- **Scan $D$ for count of each candidate**
  - $C_1$ items are scanned to count their occurrences.

**Figure 5.2** Generation of candidate itemsets and frequent itemsets, where the minimum support count is 2.
[步驟2] Suppose that the minimum support count required is 2, that is, \( \text{min}_\text{sup} = 2 \). (Here, we are referring to absolute support because we are using a support count. The corresponding relative support is \( \frac{2}{9} = 22\% \)). The set of frequent 1-itemsets, \( L_1 \), can then be determined. It consists of the candidate 1-itemsets satisfying minimum support.

![Table: \( C_1 \) and \( L_1 \)]

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup. count</th>
</tr>
</thead>
<tbody>
<tr>
<td>[12]</td>
<td>7</td>
</tr>
<tr>
<td>[13]</td>
<td>6</td>
</tr>
<tr>
<td>[14]</td>
<td>2</td>
</tr>
<tr>
<td>[15]</td>
<td>2</td>
</tr>
</tbody>
</table>

利用 \( C_1 \) 找出 \( L_1 \)

[步驟3] To discover the set of frequent 2-itemsets, \( L_2 \), the algorithm uses the join \( L_1 \times L_1 \) to generate a candidate set of 2-itemsets, \( C_2 \).

- \( C_2 \) consists of \( \binom{|L_1|}{2} \) 2-itemsets.

- Note that no candidates are removed from \( C_2 \) during the prune step (修剪) because each subset of the candidates is also frequent.

![Diagram: \( C_2 \) candidates from \( L_1 \)]

\[ \text{Permutations} : P^r_n = \frac{n!}{(n-r)!}, r \leq n \]

\[ \text{Combinations} : C^r_n = \frac{n!}{r!(n-r)!}, r \leq n \]
[步驟4] Next, the transactions in $D$ are scanned and the support count of each candidate itemset in $C$, is accumulated.

<table>
<thead>
<tr>
<th>TID</th>
<th>List of item IDs</th>
<th>Scan $D$ for count of each candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>T100</td>
<td>11, 12, 15</td>
<td></td>
</tr>
<tr>
<td>T200</td>
<td>12, 14</td>
<td></td>
</tr>
<tr>
<td>T300</td>
<td>12, 13</td>
<td></td>
</tr>
<tr>
<td>T400</td>
<td>11, 12, 14</td>
<td></td>
</tr>
<tr>
<td>T500</td>
<td>11, 13</td>
<td></td>
</tr>
<tr>
<td>T600</td>
<td>12, 13</td>
<td></td>
</tr>
<tr>
<td>T700</td>
<td>11, 13</td>
<td></td>
</tr>
<tr>
<td>T800</td>
<td>11, 12, 13, 15</td>
<td></td>
</tr>
<tr>
<td>T900</td>
<td>11, 12, 13</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup. count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{11, 12}</td>
<td>4</td>
</tr>
<tr>
<td>{11, 13}</td>
<td>4</td>
</tr>
<tr>
<td>{11, 14}</td>
<td></td>
</tr>
<tr>
<td>{11, 15}</td>
<td>4</td>
</tr>
<tr>
<td>{12, 13}</td>
<td>4</td>
</tr>
<tr>
<td>{12, 14}</td>
<td>2</td>
</tr>
<tr>
<td>{12, 15}</td>
<td>2</td>
</tr>
<tr>
<td>{13, 14}</td>
<td>0</td>
</tr>
<tr>
<td>{13, 15}</td>
<td>1</td>
</tr>
<tr>
<td>{14, 15}</td>
<td>0</td>
</tr>
</tbody>
</table>

[步驟5] The set of frequent 2-itemsets, $L_2$, is then determined, consisting of those candidate 2-itemsets in $C_2$ having minimum support. (找出 Sup.count $\geq 2$)

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup. count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{11, 12}</td>
<td>4</td>
</tr>
<tr>
<td>{11, 13}</td>
<td>4</td>
</tr>
<tr>
<td>{11, 15}</td>
<td>2</td>
</tr>
<tr>
<td>{12, 13}</td>
<td>4</td>
</tr>
<tr>
<td>{12, 14}</td>
<td>2</td>
</tr>
<tr>
<td>{12, 15}</td>
<td>2</td>
</tr>
</tbody>
</table>

Compare candidate support count with minimum support count
• [步驟6]

The generation of the set of candidate 3-itemsets, $C_3$, is detailed in Figure 5.3. From the join step, we first get $C_3 = L_2 \times L_2 = \{\{11, 12, 13\}, \{11, 12, 15\}, \{11, 13, 15\}, \{12, 13, 14\}, \{12, 13, 15\}, \{12, 14, 15\}\}$. Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that the four latter candidates cannot possibly be frequent. We therefore remove them from $C_3$, thereby saving the effort of unnecessarily obtaining their counts during the subsequent scan of $D$ to determine $L_3$.

\[
\text{Generate } C_3 \text{ candidates from } L_2
\]

\[
\begin{array}{c|c}
\text{Itemset} & \{11, 12, 13\} \\
\hline
\{11, 12, 15\}
\end{array}
\]

(a) Join: $C_3 = L_2 \times L_2 = \{\{11, 12, 13\}, \{11, 12, 15\}, \{11, 13, 15\}, \{12, 13, 14\}, \{12, 13, 15\}, \{12, 14, 15\}\} \times \\
\{\{11, 12, 13\}, \{11, 12, 15\}, \{11, 13, 15\}, \{12, 13, 14\}, \{12, 13, 15\}\} = \\
\{\{11, 12, 13\}, \{11, 12, 15\}, \{11, 13, 15\}, \{12, 13, 14\}, \{12, 13, 15\}, \{12, 14, 15\}\}.$

(b) Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent?

(a) The 2-item subsets of $\{11, 12, 13\}$ are $\{11, 12\}$, $\{11, 13\}$, and $\{12, 13\}$. All 2-item subsets of $\{11, 12, 13\}$ are members of $L_2$. Therefore, keep $\{11, 12, 13\}$ in $C_3$.

(b) The 2-item subsets of $\{11, 12, 15\}$ are $\{11, 12\}$, $\{11, 15\}$, and $\{12, 15\}$, and so it is not frequent. Therefore, remove $\{11, 12, 15\}$ from $C_3$.

(c) Therefore, $C_3 = \{\{11, 12, 13\}, \{11, 12, 15\}\}$ after pruning.

**Figure 5.3** Generation and pruning of candidate 3-itemsets, $C_3$, from $L_2$ using the Apriori property.
[步驟7] The transactions in D are scanned in order to determine \( L_3 \), consisting of those candidate 3-itemsets in \( C_3 \) having minimum support.

\[
\begin{array}{c|c|c}
\text{Itemset} & \text{Scan } D \text{ for count of each candidate} & \text{Itemset} \\
\{11, 12, 13\} & & \{11, 12, 13\} \\
\{11, 12, 15\} & & \{11, 12, 15\}
\end{array}
\]

\[
\begin{array}{c|c}
\text{Sup. count} & \\
2 & 2
\end{array}
\]

[步驟8] The algorithm uses \( L_3 \times L_3 \) to generate a candidate set of 4-itemsets, \( C_4 \).

- Although the join results in \( \{\{I_1, I_2, I_3, I_5\}\} \), this itemset is pruned because its subset \( \{I_2, I_3, I_5\} \) is not frequent.
- Thus, \( C_4 = \emptyset \), and the algorithm terminates, having found all of the frequent itemsets.

p.239 Fig. 5-4 Apriori Algorithm - Pseudo-code
p.239 Apriori Algorithm

Fig. 5-4

主程式

• 產生候選項目集
• JOIN 結合
• PRUNE 修剪

• 利用 prior knowledge

5.2.2 Generating Association Rules from Frequent Items

從頻繁項目集產生關聯規則

- Strong association rules 必須滿足二個條件:
  - minimum support
  - minimum confidence).

- This can be done using Equation (5.4) for confidence

\[ confidence(A \Rightarrow B) = \frac{P(B|A)}{P(A)} = \frac{support(A \cup B)}{support(A)} = \frac{support_count(A \cup B)}{support_count(A)}. \quad (5.4) \]

- support count(A \cup B) is the number of transactions containing the itemsets A \cup B
- support count(A) is the number of transactions containing the itemset A.

For each frequent itemset \( l \), generate all nonempty subsets of \( l \).
For every nonempty subset \( s \) of \( l \), output the rule "\( s \Rightarrow (l - s) \)" if \( \frac{\text{support count}(l)}{\text{support count}(s)} \geq \text{min_conf} \), where \( \text{min_conf} \) is the minimum confidence threshold.
Example 5.4 Generating association rules.

- Suppose the data contain the frequent itemset \( l = \{I_1, I_2, I_5\} \). What are the association rules that can be generated from \( l \)?
- Nonempty subsets of \( l \) are \( \{I_1, I_2\} \), \( \{I_1, I_5\} \), \( \{I_2, I_5\} \), \( \{I_1\} \), \( \{I_2\} \), and \( \{I_5\} \).

<table>
<thead>
<tr>
<th>( t )</th>
<th>List of itemsets</th>
</tr>
</thead>
<tbody>
<tr>
<td>T100</td>
<td>{I_1, I_2, I_5}</td>
</tr>
<tr>
<td>T200</td>
<td>{I_2, I_5}</td>
</tr>
<tr>
<td>T300</td>
<td>{I_2, I_5}</td>
</tr>
<tr>
<td>T400</td>
<td>{I_1, I_2, I_4}</td>
</tr>
<tr>
<td>T500</td>
<td>{I_1, I_3}</td>
</tr>
<tr>
<td>T600</td>
<td>{I_2, I_3}</td>
</tr>
<tr>
<td>T700</td>
<td>{I_1, I_3}</td>
</tr>
<tr>
<td>T800</td>
<td>{I_1, I_2, I_4, I_5}</td>
</tr>
<tr>
<td>T900</td>
<td>{I_1, I_2}</td>
</tr>
</tbody>
</table>

\( I_1 \land I_2 \Rightarrow I_5 \), confidence = 2/4 = 50%

\( I_1 \land I_5 \Rightarrow I_2 \), confidence = 2/2 = 100%

\( I_2 \land I_5 \Rightarrow I_1 \), confidence = 2/2 = 100%

\( I_1 \Rightarrow I_2 \land I_5 \), confidence = 2/6 = 33%

\( I_2 \Rightarrow I_1 \land I_5 \), confidence = 2/7 = 29%

\( I_5 \Rightarrow I_1 \land I_2 \), confidence = 2/2 = 100%

- If the minimum confidence threshold is, say, 70%, then only the second, third, and last rules above are output, because these are the only ones generated that are strong.
- Note that, unlike conventional classification rules, association rules can contain more than one conjunct in the right-hand side of the rule.

5.2.3 Improving the Efficiency of Apriori

- Hash-based technique:
  To reduce the size of the candidate \( k \)-itemsets, \( C_k \), \( k > 1 \).

\[ I = \{I_1, I_2, I_3, I_4, I_5\}, \text{i.e.,} \quad \text{計算} \{I_2, I_4\}: (2 \times 10 + 4) \mod 7 = 3 \]

\[ \text{order} = \{1, 2, 3, 4, 5\} \]

- Figure 5.5 Hash table, \( H_2 \), for candidate 2-itemsets: This hash table was generated by scanning the transactions of Table 5.1 while determining \( L_1 \) from \( C_1 \). If the minimum support count is, say, 3, then the items in buckets 0, 1, 3, and 4 cannot be frequent and so they should not be included in \( C_2 \).
5.2.3 Improving the Efficiency of Apriori (cont.)

2. **Transaction reduction:**
   reducing the number of transactions scanned in future iterations

3. **Partitioning:**
   partitioning the data to find candidate itemsets

4. **Sampling:**
   mining on a subset of the given data

5. **Dynamic itemset counting:**
   adding candidate itemsets at different points during a scan

5.2.4 Mining Frequent Itemsets without Candidate Generation

- the Apriori candidate generate-and-test method **significantly reduces** the size of candidate sets, leading to good performance gain.

  - 問題: 二種非明顯成本 or 隱藏成本 (nontrivial costs):
    - A huge number of candidate sets.
    - It may need to repeatedly scan the database and check a large set of candidates by pattern matching.

- 方法:
  - Frequent-Pattern growth (頻繁樣式成長) or 簡稱 FP-growth (FP成長)
Frequent-Pattern growth

- Divide-and-conquer strategy 分割與克服的策略
  - [步驟1]:
    Compress the database representing frequent items into a frequent-pattern tree (FP-tree) 頻繁樣式樹 and retains the itemset association information.
  - [步驟2]:
    Divides the compressed database into a set of conditional databases (a special kind of projected database), each associated with one frequent item or “pattern fragment,” and mines each such database separately.

Example 5.5 FP-growth
(finding frequent itemsets without candidate generation)

- [步驟1]: Scan of the database and derives the set of frequent items (1-itemsets) and their support counts frequencies
  - Let the minimum support count be 2.
  - The set of frequent items is sorted in the order of descending (遞減排序) support count. This resulting set or list is denoted \( L = \{ \{12: 7\}, \{11: 6\}, \{13: 6\}, \{14: 2\}, \{15: 2\} \} \).
Example 5.5 FP-growth (cont.)

- **[步驟2]: 建立 FP-trees**
  1. First, create the root of the tree, labeled with “null.”
  2. Scan database D a second time. The items in each transaction are processed in $L$ order (i.e., sorted according to descending support count), and a branch is created for each transaction.

$L = \{\{I2: 7\}, \{I1: 6\}, \{I3: 6\}, \{I4: 2\}, \{I5: 2\}\}$

- 第1筆記錄，“T100: I1, I2, I5” which contains three items (I2, I1, I5 in L order), construct of the first branch of the tree with three nodes, $<I2: 1>$, $<I1: 1>$, and $<I5: 1>$, where I2 is linked as a child of the root, I1 is linked to I2, and I5 is linked to I1.
- 第2筆記錄，“T200: I2, I4”, which would result in a branch where I2 is linked to the root and I4 is linked to I2. However, this branch would share a common prefix (前綴), I2, with the existing path for T100. Therefore, we instead increment the count of the I2 node by 1, and create a new node, $<I4: 1>$, which is linked as a child of $<I2: 2>$.
- In general, when considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1, and nodes for the items following the prefix are created and linked accordingly.
[步驟3]: Mining FP-trees

- Start from each frequent length-1 pattern (as an initial suffix pattern).
- Construct its conditional pattern base - consists of the set of prefix paths (前置路徑) in the FP-tree co-occurring with the suffix pattern (後置樣式),
- Construct conditional FP-tree (條件FP樹) and perform mining recursively on such a tree.
- The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree.

**Figure 5.7** An FP-tree registers compressed, frequent pattern information.
Table 5.2, 從 \( L \) 之最後一個, I5 開始

\[
L = \{ \{I2: 7\}, \{I1: 6\}, \{I3: 6\}, \{I4: 2\}, \{I5: 2\} \}
\]

<table>
<thead>
<tr>
<th>Item</th>
<th>Conditional Pattern Base</th>
<th>Conditional FP-tree</th>
<th>Frequent Patterns Generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>I5</td>
<td>{{I2, I1: 1}, {I2, I1, I3: 1}}</td>
<td>{I2: 2, I1: 2}</td>
<td>{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}</td>
</tr>
<tr>
<td>I4</td>
<td>{{I2, I1: 1}, {I2}}</td>
<td>{I2: 2}</td>
<td>{I2, I4: 2}</td>
</tr>
<tr>
<td>I3</td>
<td>{{I2, I1: 2}, {I2: 2}, {I1: 2}}</td>
<td>{I2: 4, I1: 2}, {I1: 2}</td>
<td>{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}</td>
</tr>
<tr>
<td>I1</td>
<td>{{I2: 4}}</td>
<td>{I2: 4}</td>
<td>{I2, I1: 4}</td>
</tr>
</tbody>
</table>

5.2.5 Mining Frequent Itemsets Using Vertical Data Format

如果資料格式記錄項目所在交易 ({{項目：交易編號}})，則此種資料格式被稱為垂直資料格式

<table>
<thead>
<tr>
<th>TID</th>
<th>List of item IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>T100</td>
<td>I1, I2, I5</td>
</tr>
<tr>
<td>T200</td>
<td>I2, I4</td>
</tr>
<tr>
<td>T300</td>
<td>I2, I3</td>
</tr>
<tr>
<td>T400</td>
<td>I1, I2, I4</td>
</tr>
<tr>
<td>T500</td>
<td>I1, I3</td>
</tr>
<tr>
<td>T600</td>
<td>I2, I3</td>
</tr>
<tr>
<td>T700</td>
<td>I1, I3</td>
</tr>
<tr>
<td>T800</td>
<td>I1, I2, I3, I5</td>
</tr>
<tr>
<td>T900</td>
<td>I1, I2, I3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Itemset</th>
<th>TID set</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>{T100, T400, T500, T700, T800, T900}</td>
</tr>
<tr>
<td>12</td>
<td>{T100, T200, T300, T400, T600, T800, T900}</td>
</tr>
<tr>
<td>13</td>
<td>{T500, T500, T600, T700, T800, T900}</td>
</tr>
<tr>
<td>14</td>
<td>{T200, T400}</td>
</tr>
<tr>
<td>15</td>
<td>{T100, T800}</td>
</tr>
</tbody>
</table>
R. arules package

Example - Association rules

```r
# R codes
# Example - Association rules
library(arules)
data("Adult")
# Mine association rules.
rules <- apriori(Adult, parameter = list(supp = 0.8, conf = 0.9, target = "rules", minlen=2))
inspect(Adult)    # display transactions
data(Adult)   # display association rules
# end
```

```
# display association rules
  lhs          rhs                       support confidence lift
1  (race=White) => (capital-loss=None) 0.81366489 0.8516307 0.9982720
2  (native-country=United-States) => (capital-gain=No) 0.8219565 0.9159062 0.9983862
3  (native-country=United-States) => (capital-loss=No) 0.8948380 0.9525461 0.9993323
4  (capital-gain=No) => (capital-loss=No) 0.8706646 0.9490705 0.9955863
5  (capital-loss=No) => (capital-gain=No) 0.8706646 0.9133376 0.9955863
```
1. Basic concepts and a road map
2. Efficient and scalable frequent itemset mining methods
3. **Mining various kinds of association rules**
4. From association mining to correlation analysis
5. Constraint-based association mining
6. Summary

---

5.3 Mining Various Kinds of Association Rules

- Mining multilevel association rules 多層次關聯規則
- Mining multidimensional association rules 多維度關聯規則
  - Single dimensional / Intradimensional association rule
  - Multidimensional association rules
  - Mining quantitative association rules 數量關聯規則
Mining Multiple-Level Association Rules

- Items often form hierarchies
- Flexible support settings
  - Items at the lower level are expected to have lower support
- Exploration of *shared* multi-level mining (Agrawal & Srikant@VLB’95, Han & Fu@VLDB’95)

<table>
<thead>
<tr>
<th>Level 1</th>
<th>min_sup = 5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milk</td>
<td>[support = 10%]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level 2</th>
<th>min_sup = 5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>2% Milk</td>
<td>[support = 6%]</td>
</tr>
<tr>
<td>Skim Milk</td>
<td>[support = 4%]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level 1</th>
<th>min_sup = 5%</th>
</tr>
</thead>
</table>

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.
Mining Multi-Dimensional Association

- Single-dimensional rules:
  \[ \text{buys}(X, \text{“milk”}) \Rightarrow \text{buys}(X, \text{“bread”}) \]
- Multi-dimensional rules: \( \geq 2 \) dimensions or predicates
  - Inter-dimension assoc. rules (*no repeated predicates*)
    \[ \text{age}(X,\text{“19-25”}) \land \text{occupation}(X,\text{“student”}) \Rightarrow \text{buys}(X, \text{“coke”}) \]
  - Hybrid-dimension assoc. rules (*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—data cube approach
- Quantitative Attributes: numeric, implicit ordering among values—discretization, clustering, and gradient approaches

Mining Quantitative Associations

- Techniques can be categorized by how numerical attributes, such as *age* or *salary* are treated
  1. Static discretization based on predefined concept hierarchies (data cube methods)
  2. Dynamic discretization based on data distribution (quantitative rules, e.g., Agrawal & Srikant@SIGMOD96)
  3. Clustering: Distance-based association (e.g., Yang & Miller@SIGMOD97)
     - one dimensional clustering then association
  4. Deviation: (such as Aumann and Lindell@KDD99)
     \[ \text{Sex = female} \Rightarrow \text{Wage: mean} = \$7/hr \text{ (overall mean} = \$9) \]
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

- Proposed by Lent, Swami and Widom ICDE’97
- 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 to form general a 2-D grid
- Example

$\text{age}(X, "34-35") \land \text{income}(X, "30-50K") \Rightarrow \text{buys}(X, "high \ resolution \ TV")$
Mining Other Interesting Patterns

- Flexible support constraints (Wang et al. @ VLDB’02)
  - Some items (e.g., diamond) may occur rarely but are valuable
  - Customized $\text{sup}_{\text{min}}$ specification and application
- Top-K closed frequent patterns (Han, et al. @ ICDM’02)
  - Hard to specify $\text{sup}_{\text{min}}$, but top-k with $\text{length}_{\text{min}}$ is more desirable
  - Dynamically raise $\text{sup}_{\text{min}}$ in FP-tree construction and mining, and select most promising path to mine

Chapter 5: Mining Frequent Patterns, Association and Correlations

1. Basic concepts and a road map
2. Efficient and scalable frequent itemset mining methods
3. Mining various kinds of association rules
4. From association mining to correlation analysis
5. Constraint-based association mining
6. Summary
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
- Measure of dependent/correlated events:

\[
lift = \frac{P(A ∪ B)}{P(A)P(B)}
\]

\[
lift(B, C) = \frac{2000/5000}{3000/5000*3750/5000} = 0.89
\]

\[
lift(B, ¬C) = \frac{1000/5000}{3000/5000*1250/5000} = 1.33
\]

---

Are lift and $\chi^2$ Good Measures of Correlation?

- *“Buy walnuts ⇒ buy milk* [1%, 80%]” is misleading
  - if 85% of customers buy milk
- Support and confidence are not good to represent correlations
- So many interestingness measures? (Tan, Kumar, Sritastava @KDD’02)

\[
lift = \frac{P(A ∪ B)}{P(A)P(B)}
\]

\[
all_{-conf} = \frac{\text{sup}(X)}{\text{max}_\text{item} \_ \text{sup}(X)}
\]

\[
coh = \frac{\text{sup}(X)}{\text{universe}(X)}
\]

<table>
<thead>
<tr>
<th>DB</th>
<th>m, c</th>
<th>¬m, c</th>
<th>m¬c</th>
<th>¬m¬c</th>
<th>lift</th>
<th>all-conf</th>
<th>coh</th>
<th>$\chi^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>1000</td>
<td>100</td>
<td>100</td>
<td>10,000</td>
<td>9.26</td>
<td>0.91</td>
<td>0.83</td>
<td>9055</td>
</tr>
<tr>
<td>A2</td>
<td>100</td>
<td>1000</td>
<td>1000</td>
<td>100,000</td>
<td>8.44</td>
<td>0.09</td>
<td>0.05</td>
<td>670</td>
</tr>
<tr>
<td>A3</td>
<td>1000</td>
<td>100</td>
<td>1000</td>
<td>100,000</td>
<td>9.18</td>
<td>0.09</td>
<td>0.09</td>
<td>8172</td>
</tr>
<tr>
<td>A4</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1</td>
<td>0.5</td>
<td>0.33</td>
<td>0</td>
</tr>
</tbody>
</table>
Which Measures Should Be Used?

- **lift and χ²** are not good measures for correlations in large transactional DBs
- **all-conf or coherence** could be good measures (Omiecinski@TKDE’03)
- Both **all-conf and coherence** have the downward closure property
- Efficient algorithms can be derived for mining (Lee et al. @ICDM’03sub)

### Table of Measures

<table>
<thead>
<tr>
<th>Measure</th>
<th>Range</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>lift</td>
<td>0...1</td>
<td></td>
</tr>
<tr>
<td>χ²</td>
<td>0...∞</td>
<td></td>
</tr>
<tr>
<td>all-conf</td>
<td>0...1</td>
<td></td>
</tr>
<tr>
<td>coherence</td>
<td>0...1</td>
<td></td>
</tr>
</tbody>
</table>

Chapter 5: Mining Frequent Patterns, Association and Correlations

1. Basic concepts and a road map
2. Efficient and scalable frequent itemset mining methods
3. Mining various kinds of association rules
4. From association mining to correlation analysis
5. **Constraint-based association mining**
6. Summary
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
  - System optimization: explores such constraints for efficient mining—constraint-based mining

Constraints in Data Mining

- Knowledge type constraint:
  - classification, association, etc.
- Data constraint — using SQL-like queries
  - find product pairs sold together in stores in Chicago in Dec.’02
- Dimension/level constraint
  - in relevance to region, price, brand, customer category
- Rule (or pattern) constraint
  - small sales (price < $10) triggers big sales (sum > $200)
- Interestingness constraint
  - strong rules: min_support ≥ 3%, min_confidence ≥ 60%
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

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-monotone
  - \( \text{sum}(S.\text{Price}) \geq v \) is not anti-monotone
- Example. C: range(S.profit) \( \leq 15 \) is anti-monotone
  - Itemset \( ab \) violates C
  - So does every superset of \( ab \)

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, b, e, 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
  - $\text{sum}(S.\text{Price}) \geq v$ is monotone
  - $\text{min}(S.\text{Price}) \leq v$ is monotone
- Example. $C$: range($S$.profit) $\geq$ 15
  - Itemset $ab$ satisfies $C$
  - So does every superset of $ab$

<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>

Succinctness

- Succinctness:
  - Given $A_f$, the set of items satisfying a succinctness constraint $C$, then any set $S$ satisfying $C$ is based on $A_f$, i.e., $S$ contains a subset belonging to $A_f$
  - Idea: Without looking at the transaction database, whether an itemset $S$ satisfies constraint $C$ can be determined based on the selection of items
  - $\text{min}(S.\text{Price}) \leq v$ is succinct
  - $\text{sum}(S.\text{Price}) \geq v$ is not succinct
- Optimization: If $C$ is succinct, $C$ is pre-counting pushable
The Apriori Algorithm — Example

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</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>2 5</td>
</tr>
</tbody>
</table>

Scan D

C₁

itemset sup

| {1}  | 2 |
| {2}  | 3 |
| {3}  | 3 |
| {4}  | 1 |
| {5}  | 3 |

L₁

itemset sup

| {1}  | 2 |
| {2}  | 3 |
| {3}  | 3 |
| {5}  | 3 |

Scan D

C₂

itemset sup

| {1 2} | 1 |
| {1 3} | 2 |
| {1 5} | 1 |
| {2 3} | 2 |
| {2 5} | 3 |
| {3 5} | 2 |

L₂

C₂

itemset sup

| {1 2} | 1 |
| {1 3} | 2 |
| {1 5} | 1 |
| {2 3} | 2 |
| {2 5} | 3 |
| {3 5} | 2 |

Scan D

C₃

itemset sup

| {2 3 5} | 2 |

L₃

Naïve Algorithm: Apriori + Constraint

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</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>2 5</td>
</tr>
</tbody>
</table>

Scan D

C₁

itemset sup

| {1}  | 2 |
| {2}  | 3 |
| {3}  | 3 |
| {4}  | 1 |
| {5}  | 3 |

L₁

itemset sup

| {1}  | 2 |
| {2}  | 3 |
| {3}  | 3 |
| {5}  | 3 |

Scan D

C₂ itemset sup

| {1 2} | 1 |
| {1 3} | 2 |
| {1 5} | 1 |
| {2 3} | 2 |
| {2 5} | 3 |
| {3 5} | 2 |

Scan D

C₃ itemset sup

| {2 3 5} | 2 |

Constraint: Sum(S.price) < 5
The Constrained Apriori Algorithm: Push an Anti-monotone Constraint Deep

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</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>2 5</td>
</tr>
</tbody>
</table>

Scan D

\[ \text{L1} \]

\[ \text{itemset sup.} \]

\[ \{1\} \quad 2 \]
\[ \{2\} \quad 3 \]
\[ \{3\} \quad 3 \]
\[ \{4\} \quad 1 \]
\[ \{5\} \quad 3 \]

Constraint: \( \text{Sum(S.price)} < 5 \)

\[ \text{L2} \]

\[ \text{itemset sup} \]

\[ \{1 2\} \quad 1 \]
\[ \{1 3\} \quad 2 \]
\[ \{1 5\} \quad 1 \]
\[ \{2 3\} \quad 2 \]
\[ \{2 5\} \quad 3 \]
\[ \{3 5\} \quad 2 \]

\[ \text{C2} \]

\[ \text{itemset sup} \]

\[ \{1 2\} \quad 1 \]
\[ \{1 3\} \quad 2 \]
\[ \{1 5\} \quad 1 \]
\[ \{2 3\} \quad 2 \]
\[ \{2 5\} \quad 3 \]
\[ \{3 5\} \quad 2 \]

Constraint: \( \text{min(S.price)} \leq 1 \)

\[ \text{L3} \]

\[ \text{itemset sup} \]

\[ \{2 3 5\} \quad 2 \]

The Constrained Apriori Algorithm: Push a Succinct Constraint Deep

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</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>2 5</td>
</tr>
</tbody>
</table>

Scan D

\[ \text{L1} \]

\[ \text{itemset sup.} \]

\[ \{1\} \quad 2 \]
\[ \{2\} \quad 3 \]
\[ \{3\} \quad 3 \]
\[ \{4\} \quad 1 \]
\[ \{5\} \quad 3 \]

Constraint: \( \text{Sum(S.price)} < 5 \)

\[ \text{L2} \]

\[ \text{itemset sup} \]

\[ \{1 2\} \quad 1 \]
\[ \{1 3\} \quad 2 \]
\[ \{1 5\} \quad 1 \]
\[ \{2 3\} \quad 2 \]
\[ \{2 5\} \quad 3 \]
\[ \{3 5\} \quad 2 \]

\[ \text{C2} \]

\[ \text{itemset sup} \]

\[ \{1 2\} \quad 1 \]
\[ \{1 3\} \quad 2 \]
\[ \{1 5\} \quad 1 \]
\[ \{2 3\} \quad 2 \]
\[ \{2 5\} \quad 3 \]
\[ \{3 5\} \quad 2 \]

Constraint: \( \text{min(S.price)} \leq 1 \)

\[ \text{L3} \]

\[ \text{itemset sup} \]

\[ \{2 3 5\} \quad 2 \]
Converting “Tough” Constraints

- Convert tough constraints into anti-monotone or monotone by properly ordering items

- Examine C: \( \text{avg}(S.\text{profit}) \geq 25 \)
  - Order items in value-descending order: \( \langle a, f, g, d, b, h, c, e \rangle \)
  - If an itemset \( abf \) violates C
    - So does \( afbh, afb^* \)
    - It becomes anti-monotone!

<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>

Strongly Convertible Constraints

- \( \text{avg}(X) \geq 25 \) is convertible anti-monotone w.r.t. item value descending order \( R: \langle a, f, g, d, b, h, c, e \rangle \)
  - If an itemset \( af \) violates a constraint C, so does every itemset with \( af \) as prefix, such as \( afd \)

- \( \text{avg}(X) \geq 25 \) is convertible monotone w.r.t. item value ascending order \( R^{-1}: \langle e, c, h, b, d, g, f, a \rangle \)
  - If an itemset \( d \) satisfies a constraint C, so does itemsets \( df \) and \( dfa \), which having \( d \) as a prefix

- Thus, \( \text{avg}(X) \geq 25 \) is strongly convertible
Can Apriori Handle Convertible Constraint?

A convertible, not monotone nor anti-monotone nor succinct constraint cannot be pushed deep into the an Apriori mining algorithm
- Within the level wise framework, no direct pruning based on the constraint can be made
- Itemset df violates constraint C: \( \text{avg}(X) \geq 25 \)
- Since adf satisfies C, Apriori needs df to assemble adf, df cannot be pruned
- But it can be pushed into frequent-pattern growth framework!

Mining With Convertible Constraints

- C: \( \text{avg}(X) \geq 25 \), \( \text{min} \_ \text{sup}=2 \)
- List items in every transaction in value descending order R: \(<a, f, g, d, b, h, c, e>\)
  - C is convertible anti-monotone w.r.t. R
- Scan TDB once
  - remove infrequent items
  - Item h is dropped
  - Itemsets a and f are good, …
- Projection-based mining
  - Imposing an appropriate order on item projection
  - Many tough constraints can be converted into (anti)-monotone
Handling Multiple Constraints

- Different constraints may require different or even conflicting item-ordering
- If there exists an order \( R \) s.t. both \( C_1 \) and \( C_2 \) are convertible w.r.t. \( R \), then there is no conflict between the two convertible constraints
- If there exists conflict on order of items
  - Try to satisfy one constraint first
  - Then using the order for the other constraint to mine frequent itemsets in the corresponding projected database

What Constraints Are Convertible?

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Convertible anti-monotone</th>
<th>Convertible monotone</th>
<th>Strongly convertible</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{avg}(S) \leq v \geq 0 )</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>( \text{median}(S) \leq v \geq 0 )</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>( \text{sum}(S) \leq v \geq 0 ) (items could be of any value, ( v \geq 0 ))</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>( \text{sum}(S) \leq v \leq 0 ) (items could be of any value, ( v \leq 0 ))</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>( \text{sum}(S) \geq v \geq 0 ) (items could be of any value, ( v \geq 0 ))</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>( \text{sum}(S) \geq v \leq 0 ) (items could be of any value, ( v \leq 0 ))</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>......</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Constraint-Based Mining—A General Picture

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Antimotone</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 \geq V )</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>( S \leq V )</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>( \text{min}(S) \leq v )</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>( \text{max}(S) \geq v )</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>( \text{count}(S) \geq v )</td>
<td>yes</td>
<td>no</td>
<td>weakly</td>
</tr>
<tr>
<td>( \text{count}(S) \leq v )</td>
<td>no</td>
<td>yes</td>
<td>weakly</td>
</tr>
<tr>
<td>( \sum(S) \geq (a \in S, a \geq 0) )</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>( \sum(S) \leq (a \in S, a \leq 0) )</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>( \text{range}(S) \geq v )</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>( \text{range}(S) \leq v )</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>( \text{avg}(S) \notin [\infty, \infty] )</td>
<td>convertible</td>
<td>convertible</td>
<td>no</td>
</tr>
<tr>
<td>( \text{support}(S) \notin \mathbb{N} )</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>( \text{support}(S) \notin \mathbb{N} )</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

### A Classification of Constraints

- **Antimotone**
- **Monotone**
- **Succinct**
- **Convertible**
- **Inconvertible**
- **Strongly convertible**

![Classification Diagram](image)
Chapter 5: Mining Frequent Patterns, Association and Correlations

1. Basic concepts and a road map
2. Efficient and scalable frequent itemset mining methods
3. Mining various kinds of association rules
4. From association mining to correlation analysis
5. Constraint-based association mining
6. Summary

Frequent-Pattern Mining: Summary

- Frequent pattern mining—an important task in data mining
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Projection-based (FPgrowth, CLOSET+, ...)
  - Vertical format approach (CHARM, ...)
- Mining a variety of rules and interesting patterns
- Constraint-based mining
- Mining sequential and structured patterns
- Extensions and applications
Frequent-Pattern Mining: Research Problems

- Mining fault-tolerant frequent, sequential and structured patterns
  - Patterns allows limited faults (insertion, deletion, mutation)
- Mining truly interesting patterns
  - Surprising, novel, concise, …
- Application exploration
  - E.g., DNA sequence analysis and bio-pattern classification
  - “Invisible” data mining

Ref: Basic Concepts of Frequent Pattern Mining

- (Max-pattern) R. J. Bayardo. Efficiently mining long patterns from databases. SIGMOD'98.
- (Sequential pattern) R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95