DATA MINING - 1DL105, 1DI111

Fall 2006

An introductory class in data mining

http://user.it.uu.se/~udbl/dut-ht2006/
alt. http://www.it.uu.se/edu/course/homepage/infoutv/ht06

Kjell Orsborn
Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden
Data Mining
Association Rules: Advanced Concepts and Algorithms

(Tan, Steinbach, Kumar ch. 7)

Kjell Orsborn

Department of Information Technology
Uppsala University, Uppsala, Sweden
Multi-level association rules (ch 7.3,7.4)
Multi-level association rules

• Why should we incorporate concept hierarchy?
  – Rules at lower levels may not have enough support to appear in any frequent itemsets
  – Rules at lower levels of the hierarchy are overly specific
    • e.g., skim milk → white bread, 2% milk → wheat bread, skim milk → wheat bread, etc.
    are indicative of association between milk and bread
Multi-level association rules

• How do support and confidence vary as we traverse the concept hierarchy?
  – If $X$ is the parent item for both $X_1$ and $X_2$, then
    $\sigma(X) \leq \sigma(X_1) + \sigma(X_2)$

  – If $\sigma(X_1 \cup Y_1) \geq \text{minsup}$,
    and $X$ is parent of $X_1$, $Y$ is parent of $Y_1$
    then $\sigma(X \cup Y_1) \geq \text{minsup}$, $\sigma(X_1 \cup Y) \geq \text{minsup}$
    $\sigma(X \cup Y) \geq \text{minsup}$

  – If $\text{conf}(X_1 \Rightarrow Y_1) \geq \text{minconf}$,
    then $\text{conf}(X_1 \Rightarrow Y) \geq \text{minconf}$
Multi-level association rules

• **Approach 1:**
  – Extend current association rule formulation by augmenting each transaction with higher level items

  Original Transaction: \{skim milk, wheat bread\}
  Augmented Transaction:
  \{skim milk, wheat bread, milk, bread, food\}

• **Issues:**
  – Items that reside at higher levels have much higher support counts
    • if support threshold is low, too many frequent patterns involving items from the higher levels
  – Increased dimensionality of the data
Multi-level association rules

- **Approach 2:**
  - Generate frequent patterns at highest level first
  - Then, generate frequent patterns at the next highest level, and so on

- **Issues:**
  - I/O requirements will increase dramatically because we need to perform more passes over the data
  - May miss some potentially interesting cross-level association patterns
Sequence data

Sequence Database:

<table>
<thead>
<tr>
<th>Object</th>
<th>Timestamp</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>2, 3, 5</td>
</tr>
<tr>
<td>A</td>
<td>20</td>
<td>6, 1</td>
</tr>
<tr>
<td>A</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>11</td>
<td>4, 5, 6</td>
</tr>
<tr>
<td>B</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
<td>7, 8, 1, 2</td>
</tr>
<tr>
<td>B</td>
<td>28</td>
<td>1, 6</td>
</tr>
<tr>
<td>C</td>
<td>14</td>
<td>1, 8, 7</td>
</tr>
</tbody>
</table>

Timeline

Object A:

Object B:

Object C:
## Examples of sequence data

<table>
<thead>
<tr>
<th>Sequence Database</th>
<th>Sequence</th>
<th>Element (Transaction)</th>
<th>Event (Item)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>Purchase history of a given customer</td>
<td>A set of items bought by a customer at time ( t )</td>
<td>Books, diary products, CDs, etc</td>
</tr>
<tr>
<td>Web Data</td>
<td>Browsing activity of a particular Web visitor</td>
<td>A collection of files viewed by a Web visitor after a single mouse click</td>
<td>Home page, index page, contact info, etc</td>
</tr>
<tr>
<td>Event data</td>
<td>History of events generated by a given sensor</td>
<td>Events triggered by a sensor at time ( t )</td>
<td>Types of alarms generated by sensors</td>
</tr>
<tr>
<td>Genome sequences</td>
<td>DNA sequence of a particular species</td>
<td>An element of the DNA sequence</td>
<td>Bases A,T,G,C</td>
</tr>
</tbody>
</table>

### Diagram

```
Element (Transaction)

Sequence

E1 E2 E3 E4
```

---

**Kjell Orsborn** 12/7/06
Formal definition of a sequence

• A sequence is an ordered list of elements (transactions)
  – \( s = < e_1 \ e_2 \ e_3 \ldots > \)
  – Each element contains a collection of events (items)
  – \( e_i = \{i_1, \ i_2, \ldots, \ i_k\} \)
  – Each element is attributed to a specific time or location

• Length of a sequence, \( |s| \), is given by the number of elements of the sequence

• A k-sequence is a sequence that contains k events (items)
Examples of Sequence

• Web sequence:
  – < {Homepage} {Electronics} {Digital Cameras} {Canon Digital Camera} {Shopping Cart} {Order Confirmation} {Return to Shopping} >

• Sequence of initiating events causing the nuclear accident at 3-mile Island:
  (http://stellar-one.com/nuclear/staff_reports/summary_SOE_the_initiating_event.htm)
  – < {clogged resin} {outlet valve closure} {loss of feedwater} {condenser polisher outlet valve shut} {booster pumps trip} {main waterpump trips} {main turbine trips} {reactor pressure increases} >

• Sequence of books checked out at a library:
  – < {Fellowship of the Ring} {The Two Towers} {Return of the King} >
Formal definition of a subsequence

- A sequence \(<a_1 a_2 \ldots a_n>\) is contained in another sequence \(<b_1 b_2 \ldots b_m>\)
  \((m \geq n)\) if there exist integers
  \(i_1 < i_2 < \ldots < i_n\) in such that \(a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_1}, \ldots, a_n \subseteq b_{i_n}\)

<table>
<thead>
<tr>
<th>Data sequence</th>
<th>Subsequence</th>
<th>Contain?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;{2,4} {3,5,6} {8}&gt;)</td>
<td>(&lt;{2} {3,5}&gt;)</td>
<td>Yes</td>
</tr>
<tr>
<td>(&lt;{1,2} {3,4}&gt;)</td>
<td>(&lt;{1} {2}&gt;)</td>
<td>No</td>
</tr>
<tr>
<td>(&lt;{2,4} {2,4} {2,5}&gt;)</td>
<td>(&lt;{2} {4}&gt;)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- The support of a subsequence \(w\) is defined as the fraction of data sequences that contain \(w\)
- A sequential pattern is a frequent subsequence (i.e., a subsequence whose support is \(\geq \text{minsup}\))
Sequential pattern mining: definition

• **Given:**
  – a database of sequences
  – a user-specified minimum support threshold, minsup

• **Task:**
  – Find all subsequences with support ≥ minsup
Sequential pattern mining: challenge

• Given a sequence:  
  \(<\{a\ b\} \{c\ d\ e\} \{f\} \{g\ h\ i\}\>
  
  – Examples of subsequences:
    \(<\{a\} \{c\ d\} \{f\} \{g\}\>, \(<\{c\ d\ e\}\>, \(<\{b\} \{g\}\>\), etc.

• How many k-subsequences can be extracted from a given n-sequence?
  
  \(<\{a\ b\} \{c\ d\ e\} \{f\} \{g\ h\ i\}\> \ n = 9

  \(k=4:\)

  \(<\{a\}\> \quad \{d\ e\}\> \quad \{i\}>\)

  Answer:

  \(\binom{n}{k} = \binom{9}{4} = 126\)
Sequential pattern mining: example

Object | Timestamp | Events
--- | --- | ---
A | 1 | 1,2,4
A | 2 | 2,3
A | 3 | 5
B | 1 | 1,2
B | 2 | 2,3,4
C | 1 | 1,2
C | 2 | 2,3,4
C | 3 | 2,4,5
D | 1 | 2
D | 2 | 3,4
D | 3 | 4,5
E | 1 | 1,3
E | 2 | 2,4,5

**Min sup = 50%**

**Examples of Frequent Subsequences:**

- < {1,2} > \( s=60\% \)
- < {2,3} > \( s=60\% \)
- < {2,4} > \( s=80\% \)
- < {3,5} > \( s=80\% \)
- < {1} {2} > \( s=80\% \)
- < {2} {2} > \( s=60\% \)
- < {1} {2,3} > \( s=60\% \)
- < {2} {2,3} > \( s=60\% \)
- < {1,2} {2,3} > \( s=60\% \)
Extracting sequential patterns

• Given n events: i₁, i₂, i₃, …, in

• Candidate 1-subsequences:
  • <{i₁}>, <{i₂}>, <{i₃}>, …, <{iₙ}>

• Candidate 2-subsequences:
  • <{i₁, i₂}>, <{i₁, i₃}>, …, <{i₁} {i₁}>, <{i₁} {i₂}>, …, <{iₙ₋₁} {iₙ}>

• Candidate 3-subsequences:
  • <{i₁, i₂, i₃}>, <{i₁, i₂, i₄}>, …, <{i₁, i₂} {i₁}>, <{i₁, i₂} {i₂}>, …,
  • <{i₁} {i₁, i₂}>, <{i₁} {i₁, i₃}>, …, <{i₁} {i₁} {i₁}>, <{i₁} {i₁} {i₂}>, …
Generalized sequential pattern (GSP)

- Step 1:
  - Make the first pass over the sequence database D to yield all the 1-element frequent sequences

- Step 2:

  Repeat until no new frequent sequences are found
  - Candidate Generation:
    - Merge pairs of frequent subsequences found in the (k-1)th pass to generate candidate sequences that contain k items
  - Candidate Pruning:
    - Prune candidate k-sequences that contain infrequent (k-1)-subsequences
  - Support Counting:
    - Make a new pass over the sequence database D to find the support for these candidate sequences
  - Candidate Elimination:
    - Eliminate candidate k-sequences whose actual support is less than minsup
Candidate generation

- **Base case (k=2):**
  - Merging two frequent 1-sequences \(<\{i_1\}>\) and \(<\{i_2\}>\) will produce two candidate 2-sequences: \(<\{i_1\} \{i_2\}>\) and \(<\{i_1 \ i_2\}>\)

- **General case (k>2):**
  - A frequent (k-1)-sequence \(w_1\) is merged with another frequent (k-1)-sequence \(w_2\) to produce a candidate k-sequence if the subsequence obtained by removing the first event in \(w_1\) is the same as the subsequence obtained by removing the last event in \(w_2\)
  - The resulting candidate after merging is given by the sequence \(w_1\) extended with the last event of \(w_2\).
    - If the last two events in \(w_2\) belong to the same element, then the last event in \(w_2\) becomes part of the last element in \(w_1\)
    - Otherwise, the last event in \(w_2\) becomes a separate element appended to the end of \(w_1\)
Candidate generation examples

• Merging the sequences
  \( w_1 = \langle \{1\} \{2, 3\} \{4\} \rangle \) and \( w_2 = \langle \{2, 3\} \{4, 5\} \rangle \)
  will produce the candidate sequence \( \langle \{1\} \{2, 3\} \{4, 5\} \rangle \) because the last two events in \( w_2 \) (4 and 5) belong to the same element

• Merging the sequences
  \( w_1 = \langle \{1\} \{2, 3\} \{4\} \rangle \) and \( w_2 = \langle \{2, 3\} \{4\} \{5\} \rangle \)
  will produce the candidate sequence \( \langle \{1\} \{2, 3\} \{4\} \{5\} \rangle \) because the last two events in \( w_2 \) (4 and 5) do not belong to the same element

• We do not have to merge the sequences
  \( w_1 = \langle \{1\} \{2, 6\} \{4\} \rangle \) and \( w_2 = \langle \{1\} \{2\} \{4, 5\} \rangle \)
  to produce the candidate \( \langle \{1\} \{2, 6\} \{4, 5\} \rangle \) because if the latter is a viable candidate, then it can be obtained by merging \( w_1 \) with \( \langle \{1\} \{2, 6\} \{5\} \rangle \)
GSP example

Frequent 3-sequences

Candidate Generation

Candidate Pruning
Timing constraints (I)

\[
\begin{align*}
\{A, B\} & \quad \{C\} & \quad \{D, E\} \\
\leq x_g & \quad > n_g & \leq m_s
\end{align*}
\]

\(x_g\): max-gap  
\(n_g\): min-gap  
\(m_s\): maximum span

\(x_g = 2, n_g = 0, m_s = 4\)

<table>
<thead>
<tr>
<th>Data sequence</th>
<th>Subsequence</th>
<th>Contain?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;{2,4} {3,5,6} {4,7} {4,5} {8}&gt;)</td>
<td>(&lt;{6} {5}&gt;)</td>
<td>Yes</td>
</tr>
<tr>
<td>(&lt;{1} {2} {3} {4} {5}&gt;)</td>
<td>(&lt;{1} {4}&gt;)</td>
<td>No</td>
</tr>
<tr>
<td>(&lt;{1} {2,3} {3,4} {4,5}&gt;)</td>
<td>(&lt;{2} {3} {5}&gt;)</td>
<td>Yes</td>
</tr>
<tr>
<td>(&lt;{1,2} {3} {2,3} {3,4} {2,4} {4,5}&gt;)</td>
<td>(&lt;{1,2} {5}&gt;)</td>
<td>No</td>
</tr>
</tbody>
</table>
Mining sequential patterns with timing constraints

• **Approach 1:**
  – Mine sequential patterns without timing constraints
  – Postprocess the discovered patterns

• **Approach 2:**
  – Modify GSP to directly prune candidates that violate timing constraints
  – Question:
    • Does Apriori principle still hold?
**Apriori principle for sequence data**

<table>
<thead>
<tr>
<th>Object</th>
<th>Timestamp</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1,2,4</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
<td>2,3</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1,2</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2,3,4</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1, 2</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2,3,4</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>2,4,5</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>3, 4</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>4, 5</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>1, 3</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>2, 4, 5</td>
</tr>
</tbody>
</table>

Suppose:

- \( x_g = 1 \) (max-gap)
- \( n_g = 0 \) (min-gap)
- \( m_s = 5 \) (maximum span)
- \( \text{minsup} = 60\% \)

- \(<\{2\} \{5\}>\) support = 40%
- \(\{2\} \{3\} \{5\}>\) support = 60%

**Problem exists because of max-gap constraint**

**No such problem if max-gap is infinite**
Contiguous subsequences

• s is a contiguous subsequence of
  \( w = <e_1><e_2>...<e_k> \)
if any of the following conditions hold:
  – s is obtained from \( w \) by deleting an item from either \( e_1 \) or \( e_k \)
  – s is obtained from \( w \) by deleting an item from any element \( e_i \) that contains more than 2 items
  – s is a contiguous subsequence of \( s' \) and \( s' \) is a contiguous subsequence of \( w \)
    (recursive definition)

• Examples:  \( s = <\{1\} \{2\}> \)
  – is a contiguous subsequence of
    \( <\{1\} \{2\} \{3\}>, <\{1\} \{2\} \{3\}> \)
  – is not a contiguous subsequence of
    \( <\{1\} \{3\} \{2\}> \) and \( <\{2\} \{1\} \{3\} \{2\}> \)
Modified candidate pruning step

• **Without maxgap constraint:**
  – A candidate k-sequence is pruned if at least one of its (k-1)-subsequences is infrequent

• **With maxgap constraint:**
  – A candidate k-sequence is pruned if at least one of its contiguous (k-1)-subsequences is infrequent
Timing constraints (II)

\[ \{A \ B\} \quad \{C\} \quad \{D \ E\} \]

\[ \leq x_g \quad > n_g \quad \leq ws \]

\( x_g: \) max-gap
\( n_g: \) min-gap
\( ws: \) window size
\( m_s: \) maximum span

\[ x_g = 2, \ n_g = 0, \ ws = 1, \ m_s = 5 \]

<table>
<thead>
<tr>
<th>Data sequence</th>
<th>Subsequence</th>
<th>Contain?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt; {2,4} \ {3,5,6} \ {4,7} \ {4,6} \ {8}&gt;)</td>
<td>(&lt; {3} \ {5}&gt;)</td>
<td>No</td>
</tr>
<tr>
<td>(&lt; {1} \ {2} \ {3} \ {4} \ {5}&gt;)</td>
<td>(&lt; {1,2} \ {3}&gt;)</td>
<td>Yes</td>
</tr>
<tr>
<td>(&lt; {1,2} \ {2,3} \ {3,4} \ {4,5}&gt;)</td>
<td>(&lt; {1,2} \ {3,4}&gt;)</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Modified support counting step

• Given a candidate pattern: \(<\{a, c\}>\)
  
  – Any data sequences that contain

    \(<\ldots \{a \text{ c} \ldots \}>,
    \)
    \(<\ldots \{a\} \ldots \{c\} \ldots \text{> ( where time(\{c\}) – time(\{a\}) ≤ ws)}\)
    \(<\ldots \{c\} \ldots \{a\} \ldots \text{> (where time(\{a\}) – time(\{c\}) ≤ ws)}\)

  will contribute to the support count of candidate pattern
Other formulation

• In some domains, we may have only one very long time series
  – Example:
    • monitoring network traffic events for attacks
    • monitoring telecommunication alarm signals

• Goal is to find frequent sequences of events in the time series
  – This problem is also known as frequent episode mining

Pattern: <E1> <E3>
Assume:
\[ x_g = 2 \text{ (max-gap)} \]
\[ n_g = 0 \text{ (min-gap)} \]
\[ ws = 0 \text{ (window size)} \]
\[ m_s = 2 \text{ (maximum span)} \]