Sequential Pattern Mining

Lecture Notes for Chapter 7 – Introduction to Data Mining
Tan, Steinbach, Kumar
From itemsets to sequences

- Frequent itemsets and association rules focus on transactions and the items that appear there.
- Databases of transactions usually have a temporal information.
  - Sequential patterns exploit it.
- Example data:
  - Market basket transactions
  - Web server logs
  - Tweets
  - Workflow production logs
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>
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>

Element (Transaction) Sequence:

- E1
- E2
- E1
- E3
- E2
- E2
- E3
- E4

Event (Item)
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 $\langle a_1, a_2, \ldots, a_n \rangle$ is contained in another sequence $\langle b_1, b_2, \ldots, b_m \rangle$ (m $\geq$ n) if there exist integers $i_1 < i_2 < \ldots < i_n$ 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>$\langle {2,4}, {3,5,6}, {8} \rangle$</td>
<td>$\langle {2}, {3,5} \rangle$</td>
<td>Yes</td>
</tr>
<tr>
<td>$\langle {1,2}, {3,4} \rangle$</td>
<td>$\langle {1}, {2} \rangle$</td>
<td>No</td>
</tr>
<tr>
<td>$\langle {2,4}, {2,4}, {2,5} \rangle$</td>
<td>$\langle {2}, {4} \rangle$</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, $\text{minsup}$

- **Task:**
  - Find all subsequences with support $\geq \text{minsup}$
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}>  \text{n = 9}
```

```
k=4: \quad Y \_ \_ Y Y \_ \_ \_ Y
```

```
<{a} \{d e\} \{i}\>
```

Answer:
```
\binom{n}{k} = \binom{9}{4} = 126
```
# Sequential Pattern Mining: Example

## 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%

---

<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>
Exercises

- find instances/occurrence of the following patterns

\[
\langle \{C\} \{H\} \{C\} \rangle \\
\langle \{A\} \{F\} \rangle \\
\langle \{A\} \{A\} \{D\} \rangle \\
\langle \{A\} \{A,B\} \{F\} \rangle
\]

- in the input sequence below

\[
\langle \{A,C\} \{C,D\} \{F,H\} \{A,B\} \{B,C,D\} \{E\} \{A,B,D\} \{F\} \rangle
\]
Exercises

● find instances/occurrence of the following patterns

< {C} {H} {C} >
< {A} {B} >
< {C} {C} {E} >
< {A} {E} >

● in the input sequence below

< < A,C > < C,D,E > < F > < A,H > < B,C,D > < E > < A,B,D > >

< < A,C > < C,D,E > < F > < A,H > < B,C,D > < E > < A,B,D > >
Extracting Sequential Patterns

- Given n events: \( i_1, i_2, i_3, \ldots, i_n \)

- Candidate 1-subsequences:
  \( \langle \{i_1\} \rangle, \langle \{i_2\} \rangle, \langle \{i_3\} \rangle, \ldots, \langle \{i_n\} \rangle \)

- Candidate 2-subsequences:
  \( \langle \{i_1, i_2\} \rangle, \langle \{i_1, i_3\} \rangle, \ldots, \langle \{i_1\} \{i_1\} \rangle, \langle \{i_1\} \{i_2\} \rangle, \ldots, \langle \{i_{n-1}\} \{i_n\} \rangle \)

- Candidate 3-subsequences:
  \( \langle \{i_1, i_2, i_3\} \rangle, \langle \{i_1, i_2, i_4\} \rangle, \ldots, \langle \{i_1, i_2\} \{i_1\} \rangle, \langle \{i_1, i_2\} \{i_2\} \rangle, \ldots, \langle \{i_1\} \{i_1, i_2\} \rangle, \langle \{i_1\} \{i_1, i_3\} \rangle, \ldots, \langle \{i_1\} \{i_1\} \{i_1\} \rangle, \langle \{i_1\} \{i_1\} \{i_2\} \rangle, \ldots \)
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 \( \text{minsup} \)
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**

- \(<\{1\} \{2\} \{3\}>\)
- \(<\{1\} \{2\ 5\}>\)
- \(<\{1\} \{5\} \{3\}>\)
- \(<\{2\} \{3\} \{4\}>\)
- \(<\{2\ 5\} \{3\}>\)
- \(<\{3\} \{4\} \{5\}>\)
- \(<\{5\} \{3\ 4\}>\)

**Candidate Generation**

- \(<\{1\} \{2\} \{3\} \{4\}>\)
- \(<\{1\} \{2\ 5\} \{3\}>\)
- \(<\{1\} \{5\} \{3\ 4\}>\)
- \(<\{2\} \{3\} \{4\} \{5\}>\)
- \(<\{2\ 5\} \{3\ 4\}>\)

**Candidate Pruning**

- \(<\{1\} \{2\ 5\} \{3\}>\)
GSP Exercise

Given the following dataset of sequences

<table>
<thead>
<tr>
<th>ID</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a b → a → b</td>
</tr>
<tr>
<td>2</td>
<td>b → a → c d</td>
</tr>
<tr>
<td>3</td>
<td>a → b</td>
</tr>
<tr>
<td>4</td>
<td>a → a → b d</td>
</tr>
</tbody>
</table>

Generate sequential patterns if min_sup = 35%
<table>
<thead>
<tr>
<th>Sequential pattern</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>100 %</td>
</tr>
<tr>
<td>b</td>
<td>100 %</td>
</tr>
<tr>
<td>d</td>
<td>50 %</td>
</tr>
<tr>
<td>a → a</td>
<td>50 %</td>
</tr>
<tr>
<td>a → b</td>
<td>75 %</td>
</tr>
<tr>
<td>a → d</td>
<td>50 %</td>
</tr>
<tr>
<td>b → a</td>
<td>50 %</td>
</tr>
<tr>
<td>a → a → b</td>
<td>50 %</td>
</tr>
</tbody>
</table>
Timing Constraints (I)

Data sequence: {A B} {C} {D E}

- \( x_g \): 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

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

\(<\{2\} \{5\}> \text{ support } = 40\% \]
\(<\{2\} \{3\} \{5\}> \text{ 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 $w = <e_1><e_2>\ldots<e_k>$ if any of the following conditions hold:
  1. $s$ is obtained from $w$ by deleting an item from either $e_1$ or $e_k$
  2. $s$ is obtained from $w$ by deleting an item from any element $e_i$ that contains more than 2 items
  3. $s$ 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 $<\{1\} \{2\} \{3\}>$, $<\{1\} \{2\} \{2\} \{3\}>$, and $<\{3\} \{4\} \{1\} \{2\} \{2\} \{3\} \{4\}>$
- 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)

Timing constraints:

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

Data sequence

<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>

\( x_g = 2, n_g = 0, ws = 1, m_s = 5 \)
Modified Support Counting Step

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

\(<\ldots \{a \ c\} \ldots \>, \n\langle \ldots \{a\} \ldots \{c\} \ldots \rangle \ (\text{where} \ \text{time}({\{c\}}) - \text{time}({\{a\}}) \leq \text{ws}) \n\langle \ldots \{c\} \ldots \{a\} \ldots \rangle \ (\text{where} \ \text{time}({\{a\}}) - \text{time}({\{c\}}) \leq \text{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: \( <E_1> <E_3> \)
## General Support Counting Schemes

### Object's Timeline

<table>
<thead>
<tr>
<th>Method</th>
<th>Support Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>COBJ</td>
<td>1</td>
</tr>
<tr>
<td>CWIN</td>
<td>6</td>
</tr>
<tr>
<td>CMINWIN</td>
<td>4</td>
</tr>
<tr>
<td>CDIST_O</td>
<td>8</td>
</tr>
<tr>
<td>CDIST</td>
<td>5</td>
</tr>
</tbody>
</table>

Sequence: (p) (q)

Assume:

- $x_g = 2$ (max-gap)
- $n_g = 0$ (min-gap)
- $ws = 0$ (window size)
- $m_s = 2$ (maximum span)