MSCIT 5210: Knowledge Discovery and Data Mining

Acknowledgement: Slides modified by Dr. Lei Chen based on the slides provided by Tan, Steinbach, Kumar, and Jiawei Han, Micheline Kamber, and Jian Pei
Continuous and Categorical Attributes

How to apply association analysis formulation to non-asymmetric binary variables?

<table>
<thead>
<tr>
<th>Session Id</th>
<th>Country</th>
<th>Session Length (sec)</th>
<th>Number of Web Pages viewed</th>
<th>Gender</th>
<th>Browser Type</th>
<th>Buy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>USA</td>
<td>982</td>
<td>8</td>
<td>Male</td>
<td>IE</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>China</td>
<td>811</td>
<td>10</td>
<td>Female</td>
<td>Netscape</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>USA</td>
<td>2125</td>
<td>45</td>
<td>Female</td>
<td>Mozilla</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Germany</td>
<td>596</td>
<td>4</td>
<td>Male</td>
<td>IE</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Australia</td>
<td>123</td>
<td>9</td>
<td>Male</td>
<td>Mozilla</td>
<td>No</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Example of Association Rule:

\[ \{\text{Number of Pages} \in [5,10) \land (\text{Browser}=\text{Mozilla})\} \rightarrow \{\text{Buy} = \text{No}\} \]
Handling Categorical Attributes

- Transform categorical attribute into asymmetric binary variables

- Introduce a new “item” for each distinct attribute-value pair
  - Example: replace Browser Type attribute with
    - Browser Type = Internet Explorer
    - Browser Type = Mozilla
    - Browser Type = Mozilla
Handling Categorical Attributes

Potential Issues

- What if attribute has many possible values
  - Example: attribute country has more than 200 possible values
  - Many of the attribute values may have very low support
    - Potential solution: Aggregate the low-support attribute values

- What if distribution of attribute values is highly skewed
  - Example: 95% of the visitors have Buy = No
  - Most of the items will be associated with (Buy=No) item
    - Potential solution: drop the highly frequent items
Handling Continuous Attributes

Different kinds of rules:
- Age $\in [21, 35) \land$ Salary $\in [70k, 120k) \rightarrow$ Buy
- Salary $\in [70k, 120k) \land$ Buy $\rightarrow$ Age: $\mu=28, \sigma=4$

Different methods:
- Discretization-based
- Statistics-based
- Non-discretization based
  - minApriori
Handling Continuous Attributes

- Use discretization
- Unsupervised:
  - Equal-width binning
  - Equal-depth binning
  - Clustering
- Supervised:

<table>
<thead>
<tr>
<th>Class</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_5$</th>
<th>$v_6$</th>
<th>$v_7$</th>
<th>$v_8$</th>
<th>$v_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomalous</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Normal</td>
<td>150</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>100</td>
<td>150</td>
<td>100</td>
</tr>
</tbody>
</table>

Attribute values, $v$
Discretization Issues

- Size of the discretized intervals affect support & confidence
  
  \{\text{Refund} = \text{No}, (\text{Income} = 51,250)\} \rightarrow \{\text{Cheat} = \text{No}\}
  
  \{\text{Refund} = \text{No}, (60K \leq \text{Income} \leq 80K)\} \rightarrow \{\text{Cheat} = \text{No}\}
  
  \{\text{Refund} = \text{No}, (0K \leq \text{Income} \leq 1B)\} \rightarrow \{\text{Cheat} = \text{No}\}

- If intervals too small
  - may not have enough support

- If intervals too large
  - may not have enough confidence

- Potential solution: use all possible intervals
Discretization Issues

- Execution time
  - If intervals contain $n$ values, there are on average $O(n^2)$ possible ranges

- Too many rules

  \{
  \text{Refund} = \text{No}, \ (\text{Income} = \$51,250) \} \rightarrow \{ \text{Cheat} = \text{No} \}

  \{
  \text{Refund} = \text{No}, \ (51K \leq \text{Income} \leq 52K) \} \rightarrow \{ \text{Cheat} = \text{No} \}

  \{
  \text{Refund} = \text{No}, \ (50K \leq \text{Income} \leq 60K) \} \rightarrow \{ \text{Cheat} = \text{No} \\}
Approach by Srikant & Agrawal

- Preprocess the data
  - Discretize attribute using equi-depth partitioning
    - Use \textit{partial completeness measure} to determine number of partitions
    - Merge adjacent intervals as long as support is less than max-support

- Apply existing association rule mining algorithms

- Determine interesting rules in the output
Approach by Srikant & Agrawal

- Discretization will lose information

  ![Approximated X](image)

  ![X](image)

- Use *partial completeness measure* to determine how much information is lost

  C: frequent itemsets obtained by considering all ranges of attribute values
  P: frequent itemsets obtained by considering all ranges over the partitions

  P is *K-complete w.r.t C* if P ⊆ C, and ∀X ∈ C, ∃ X′ ∈ P such that:
  1. X′ is a generalization of X and support (X′) ≤ K × support(X) (K ≥ 1)
  2. ∀Y ⊆ X, ∃ Y′ ⊆ X′ such that support (Y′) ≤ K × support(Y)

Given *K (partial completeness level)*, can determine number of intervals (N)
Interestingness Measure

Given an itemset: $Z = \{z_1, z_2, ..., z_k\}$ and its generalization $Z' = \{z_1', z_2', ..., z_k'\}$

- $P(Z)$: support of $Z$
- $E_{Z'}(Z)$: expected support of $Z$ based on $Z'$

$$E_{Z'}(Z) = \frac{P(z_1)}{P(z_1')} \times \frac{P(z_2)}{P(z_2')} \times \cdots \times \frac{P(z_k)}{P(z_k')} \times P(Z')$$

- $Z$ is R-interesting w.r.t. $Z'$ if $P(Z) \geq R \times E_{Z'}(Z)$

{Refund = No, (Income = $51,250)} → {Cheat = No}
{Refund = No, (51K ≤ Income ≤ 52K)} → {Cheat = No}
{Refund = No, (50K ≤ Income ≤ 60K)} → {Cheat = No}
Interestingness Measure

- For \( S: X \rightarrow Y \), and its generalization \( S': X' \rightarrow Y' \)
  - \( P(Y|X) \): confidence of \( X \rightarrow Y \)
  - \( P(Y'|X') \): confidence of \( X' \rightarrow Y' \)
  - \( E_{S'}(Y|X) \): expected support of \( Z \) based on \( Z' \)

\[
E(Y \mid X) = \frac{P(y_1)}{P(y_1')} \times \frac{P(y_2)}{P(y_2')} \times \cdots \times \frac{P(y_k)}{P(y_k')} \times P(Y' \mid X')
\]

- Rule \( S \) is \( R \)-interesting w.r.t its ancestor rule \( S' \) if
  - Support, \( P(S) \geq R \times E_{S'}(S) \) or
  - Confidence, \( P(Y|X) \geq R \times E_{S'}(Y|X) \)
Statistics-based Methods

- **Example:**
  
  \[ \text{Browser} = \text{Mozilla} \land \text{Buy} = \text{Yes} \rightarrow \text{Age: } \mu = 23 \]

- Rule consequent consists of a continuous variable, characterized by their statistics
  - mean, median, standard deviation, etc.

- **Approach:**
  - Withhold the target variable from the rest of the data
  - Apply existing frequent itemset generation on the rest of the data
  - For each frequent itemset, compute the descriptive statistics for the corresponding target variable
    - Frequent itemset becomes a rule by introducing the target variable as rule consequent
  - Apply statistical test to determine interestingness of the rule
Statistics-based Methods

How to determine whether an association rule interesting?

- Compare the statistics for segment of population covered by the rule vs segment of population not covered by the rule:

  \[ A \Rightarrow B: \mu \text{ versus } A \Rightarrow B: \mu' \]

  \[ Z = \frac{\mu' - \mu - \Delta}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \]

- Statistical hypothesis testing:
  - Null hypothesis: \( H_0: \mu' = \mu + \Delta \)
  - Alternative hypothesis: \( H_1: \mu' > \mu + \Delta \)
  - \( Z \) has zero mean and variance 1 under null hypothesis
Statistics-based Methods

Example:

\[ r: \text{Browser=Mozilla} \land \text{Buy=Yes} \rightarrow \text{Age: } \mu=23 \]

- Rule is interesting if difference between \( \mu \) and \( \mu' \) is greater than 5 years (i.e., \( \Delta = 5 \))
- For \( r \), suppose \( n_1 = 50, s_1 = 3.5 \)
- For \( r' \) (complement): \( n_2 = 250, s_2 = 6.5 \)

\[
Z = \frac{\mu' - \mu - \Delta}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} = \frac{30 - 23 - 5}{\sqrt{\frac{3.5^2}{50} + \frac{6.5^2}{250}}} = 3.11
\]

- For 1-sided test at 95% confidence level, critical Z-value for rejecting null hypothesis is 1.64.
- Since \( Z \) is greater than 1.64, \( r \) is an interesting rule
**Min-Apriori (Han et al)**

Document-term matrix:

<table>
<thead>
<tr>
<th>TID</th>
<th>W1</th>
<th>W2</th>
<th>W3</th>
<th>W4</th>
<th>W5</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>D3</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Example:

W1 and W2 tends to appear together in the same document
Min-Apriori

- Data contains only continuous attributes of the same “type”
  - e.g., frequency of words in a document

- Potential solution:
  - Convert into 0/1 matrix and then apply existing algorithms
    - lose word frequency information
  - Discretization does not apply as users want association among words not ranges of words
Min-Apriori

- How to determine the support of a word?
  - If we simply sum up its frequency, support count will be greater than total number of documents!
    - Normalize the word vectors – e.g., using $L_1$ norm
    - Each word has a support equals to 1.0

<table>
<thead>
<tr>
<th>TID</th>
<th>W1</th>
<th>W2</th>
<th>W3</th>
<th>W4</th>
<th>W5</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>D3</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TID</th>
<th>W1</th>
<th>W2</th>
<th>W3</th>
<th>W4</th>
<th>W5</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>0.40</td>
<td>0.33</td>
<td>0.00</td>
<td>0.00</td>
<td>0.17</td>
</tr>
<tr>
<td>D2</td>
<td>0.00</td>
<td>0.00</td>
<td>0.33</td>
<td>1.00</td>
<td>0.33</td>
</tr>
<tr>
<td>D3</td>
<td>0.40</td>
<td>0.50</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>D4</td>
<td>0.00</td>
<td>0.00</td>
<td>0.33</td>
<td>0.00</td>
<td>0.17</td>
</tr>
<tr>
<td>D5</td>
<td>0.20</td>
<td>0.17</td>
<td>0.33</td>
<td>0.00</td>
<td>0.33</td>
</tr>
</tbody>
</table>
New definition of support:

$$\text{sup}(C) = \sum_{i \in T} \min_{j \in C} D(i, j)$$

Example:

$$\text{Sup}(W_1, W_2, W_3) = 0 + 0 + 0 + 0 + 0.17 = 0.17$$
Anti-monotone property of Support

Example:

Sup(W1) = 0.4 + 0 + 0.4 + 0 + 0.2 = 1
Sup(W1, W2) = 0.33 + 0 + 0.4 + 0 + 0.17 = 0.9
Sup(W1, W2, W3) = 0 + 0 + 0 + 0 + 0.17 = 0.17
Multi-level Association Rules

Food
- Bread
  - Wheat
  - White
  - Skim
  - Skim 2%
  - Foremost
  - Kemps

Electronics
- Computers
  - Desktop
  - Laptop
  - Accessory
  - TV
  - DVD
- Home
  - Printer
  - Scanner

Computers
  - Desktop
  - Laptop
  - Accessory
  - TV
  - DVD

Home
  - Printer
  - Scanner
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 $\rightarrow$ white bread, 2% milk $\rightarrow$ wheat bread,
    - skim milk $\rightarrow$ 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 $\sigma(X_1 \cup Y_1) \geq \text{minsуп}$,
  and $X$ is parent of $X_1$, $Y$ is parent of $Y_1$
  then $\sigma(X \cup Y_1) \geq \text{minsуп}$, $\sigma(X_1 \cup Y) \geq \text{minsуп}$
    $\sigma(X \cup Y) \geq \text{minsуп}$

- If $\text{conf}(X_1 \Rightarrow Y_1) \geq \text{minконф}$,
  then $\text{conf}(X_1 \Rightarrow Y) \geq \text{minконф}$
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:**
  - 2
  - 3
  - 5
  - 6
  - 1

- **Object B:**
  - 4
  - 5
  - 6
  - 8
  - 7
  - 1
  - 2

- **Object C:**
  - 1
  - 7
  - 8
## 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>

![Sequence Diagram](image)

- **Sequence**: E1 E2 E3 E2 E4
- **Element (Transaction)**: E1 E2 E3 E2 E3
- **Event (Item)**: E4
Formal Definition of a Sequence

- A sequence is an ordered list of elements (transactions)
  \[ s = < e_1 e_2 e_3 ... > \]
  - Each element contains a collection of events (items)
    \[ e_i = \{i_1, i_2, ..., i_k\} \]
  - 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 \(i_1 < i_2 < \ldots < i_n\) 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, \( \text{mins}\up{up} \)

Task:
- Find all subsequences with support \( \geq \text{mins}\up{up} \)
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?

\[
\begin{align*}
\{a\ b\} \{c\ d\ e\} \{f\} \{g\ h\ i\} & \quad n = 9 \\
\text{k=4:} \quad & \underbrace{Y \_ \_ Y Y \_ \_ \_ Y}_{\text{Answer :}} \\
{\{a\}} \quad & \{d\ e\} \quad \{i\}
\end{align*}
\]

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

<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>

**Minsup = 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} >` \( s = 80\% \)
- `< {1} {2, 3} >` \( s = 60\% \)
- `< {2} {2, 3} >` \( s = 60\% \)
- `< {1, 2} {2, 3} >` \( s = 60\% \)
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 \( \langle \{1\} \{2\ 6\} \{5\} \rangle \).
**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 $\text{minsup}$
Suppose now we have 3 events: 1, 2, 3, and let min-support be 50%.

*The sequence database is shown in following table:*

<table>
<thead>
<tr>
<th>Object</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1), (2), (3)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2), (3)</td>
</tr>
<tr>
<td>C</td>
<td>(1), (2, 3)</td>
</tr>
<tr>
<td>D</td>
<td>(1, 2, 3)</td>
</tr>
<tr>
<td>E</td>
<td>(1, 2), (2, 3), (1, 3)</td>
</tr>
</tbody>
</table>
Step 1: Make the first pass over the sequence database D to yield all the 1-element frequent sequences

<table>
<thead>
<tr>
<th>Object</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1), (2), (3)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2), (3)</td>
</tr>
<tr>
<td>C</td>
<td>(1), (2, 3)</td>
</tr>
<tr>
<td>D</td>
<td>(1, 2, 3)</td>
</tr>
<tr>
<td>E</td>
<td>(1, 2), (2, 3), (1, 3)</td>
</tr>
</tbody>
</table>

\(<\{1\}\>, \,<\{2\}\>, \,<\{3\}\>
generate candidate sequences that contain k items

<table>
<thead>
<tr>
<th>Object</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1), (2), (3)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2), (3)</td>
</tr>
<tr>
<td>C</td>
<td>(1), (2, 3)</td>
</tr>
<tr>
<td>D</td>
<td>(1, 2, 3)</td>
</tr>
<tr>
<td>E</td>
<td>(1, 2), (2, 3), (1, 3)</td>
</tr>
</tbody>
</table>

Candidate 1-sequences are:

<{1}>, <{2}>, <{3}>

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}>

Candidate 2-sequences are:

<{1, 2}>,  <{1, 3}>,  <{2, 3}>,
<{1}, {1}>,  <{1}, {2}>,  <{1}, {3}>,
<{2}, {1}>,  <{2}, {2}>,  <{2}, {3}>,
<{3}, {1}>,  <{3}, {2}>,  <{3}, {3}>
Step 2: Candidate Pruning:
Prune candidate k-sequences that contain infrequent (k-1)-subsequences

<table>
<thead>
<tr>
<th>Object</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1), (2), (3)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2), (3)</td>
</tr>
<tr>
<td>C</td>
<td>(1), (2, 3)</td>
</tr>
<tr>
<td>D</td>
<td>(1, 2, 3)</td>
</tr>
<tr>
<td>E</td>
<td>(1, 2), (2, 3), (1, 3)</td>
</tr>
</tbody>
</table>

After candidate pruning, the 2-sequences should remain the same:

\[
\{1, 2\}, \{1, 3\}, \{2, 3\}, \{1\}, \{1\}, \{1\}, \{2\}, \{2\}, \{2\}, \{3\}, \{3\}, \{3\}, \{3\}
\]
**Step 2: Support Counting and Candidate Elimination:**

<table>
<thead>
<tr>
<th>Object</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1), (2), (3)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2), (3)</td>
</tr>
<tr>
<td>C</td>
<td>(1), (2, 3)</td>
</tr>
<tr>
<td>D</td>
<td>(1, 2, 3)</td>
</tr>
<tr>
<td>E</td>
<td>(1, 2), (2, 3), (1, 3)</td>
</tr>
</tbody>
</table>

After candidate elimination, the remaining **frequent 2-sequences** are:

- \(<\{1, 2\}>\) (support=0.6),
- \(<\{2, 3\}>\) (support=0.6),
- \(<\{1\}, \{2\}>\) (support=0.6),
- \(<\{1\}, \{3\}>\) (support=0.8),
- \(<\{2\}, \{3\}>\) (support=0.6)
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 \(w_1\).
<table>
<thead>
<tr>
<th>Object</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1), (2), (3)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2), (3)</td>
</tr>
<tr>
<td>C</td>
<td>(1), (2, 3)</td>
</tr>
<tr>
<td>D</td>
<td>(1, 2, 3)</td>
</tr>
<tr>
<td>E</td>
<td>(1, 2), (2, 3), (1, 3)</td>
</tr>
</tbody>
</table>

Generate **3-sequences** from the remaining **2-sequences**:

- \(<\{1, 2\}\>\), \(<\{2, 3\}\>\), \(<\{1\}, \{2\}\>\), \(<\{1\}, \{3\}\>\), \(<\{2\}, \{3\}\>\)

3-sequences are:

- \(<\{1, 2, 3\}\>\) (generated from \(<\{1, 2\}\>\) and \(<\{2, 3\}\>\)),
- \(<\{1, 2\}, \{3\}\>\) (generated from \(<\{1, 2\}\>\) and \(<\{2\}, \{3\}\>\)),
- \(<\{1\}, \{2\}, \{3\}\>\) (generated from \(<\{1\}, \{2\}\>\) and \(<\{2\}, \{3\}\>\))
- \(<\{1\}, \{2\ 3\}\>\) (generated from \(<\{1\}, \{2\}\>\) and \(<\{2, 3\}\>\))
Repeat Step 2: **Candidate Pruning**

<table>
<thead>
<tr>
<th>Object</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1), (2), (3)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2), (3)</td>
</tr>
<tr>
<td>C</td>
<td>(1), (2, 3)</td>
</tr>
<tr>
<td>D</td>
<td>(1, 2, 3)</td>
</tr>
<tr>
<td>E</td>
<td>(1, 2), (2, 3), (1, 3)</td>
</tr>
</tbody>
</table>

3-sequences:
<{1, 2, 3}> should be **pruned** because one 2-subsequences <{1, 3}> is **not** frequent.

<{1, 2}, {3}> should **not** be pruned because all 2-subsequences <{1}, {3}> and <{2}, {3}> are frequent.

<{1}, {2}, {3}> should **not** be pruned because all 2-subsequences <{1}, {2}>, <{2}, {3}> and <{1}, {3}> are frequent.

So after pruning, the remaining 3-sequences are:
<{1, 2}, {3}>, <{1}, {2}, {3}>, and <{1}, {2, 3}>
Repeat Step 2: Support Counting

<table>
<thead>
<tr>
<th>Object</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1), (2), (3)</td>
</tr>
<tr>
<td>B</td>
<td>(1, 2), (3)</td>
</tr>
<tr>
<td>C</td>
<td>(1), (2, 3)</td>
</tr>
<tr>
<td>D</td>
<td>(1, 2, 3)</td>
</tr>
<tr>
<td>E</td>
<td>(1, 2), (2, 3), (1, 3)</td>
</tr>
</tbody>
</table>

Remaining 3-sequences:

- \(<\{1, 2\}, \{3\}\>\), support = 0.4 < 0.5, should be eliminated
- \(<\{1\}, \{2\}, \{3\}\>\), support = 0.4 < 0.5, should be eliminated.
- \(<\{1\}, \{2\, 3\}\>\), support = 0.4 < 0.5, should be eliminated

Thus, there are no 3-sequences left.

So the final frequent sequences are:

\(<\{1\}\>\), \(<\{2\}\>\), \(<\{3\}\>\),
\(<\{1, 2\}\>, \(<\{2, 3\}\>\), \(<\{1\}, \{2\}\>\), \(<\{1\}, \{3\}\>\), \(<\{2\}, \{3\}\>\)
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

- \(<\{1\} \{2\ 5\} \{3\} >\)
Timing Constraints (I)

Let's consider the following notation:

- \( x_g \): 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:

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

\[
\{\{2\} \{5\}\} \text{ support } = 40\%
\]

but

\[
\{\{2\} \{3\} \{5\}\} \text{ support } = 60\%
\]

Problem exists because of max-gap constraint

No such problem if max-gap is infinite
Contiguous Subsequences

- $s$ is a contiguous subsequence of $w = \langle e_1 \rangle \langle e_2 \rangle \ldots \langle e_k \rangle$
  
  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: $s = \langle \{1\} \{2\} \rangle$
  - is a contiguous subsequence of $\langle \{1\} \{2\} \{3\}\rangle$, $\langle \{1\} \{2\} \{3\}\rangle$, and $\langle \{3\} \{4\} \{1\} \{2\} \{2\} \{3\} \{4\} \rangle$
  - is not a contiguous subsequence of $\langle \{1\} \{3\} \{2\}\rangle$ and $\langle \{2\} \{1\} \{3\} \{2\}\rangle$
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)

- $x_g$: 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
    
    \(<\ldots \{a \ c\} \ \ldots \ >,\)
    
    \(<\ldots \{a\} \ \ldots \ \{c\}\ldots\>\)  \(\text{(where } \text{time}\{\{c\}\} - \text{time}\{\{a\}\} \leq \text{ws})\)
    
    \(<\ldots\{c\} \ \ldots \ \{a\} \ \ldots\>\)  \(\text{(where } \text{time}\{\{a\}\} - \text{time}\{\{c\}\} \leq \text{ws})\)

    will contribute to the support count of candidate pattern