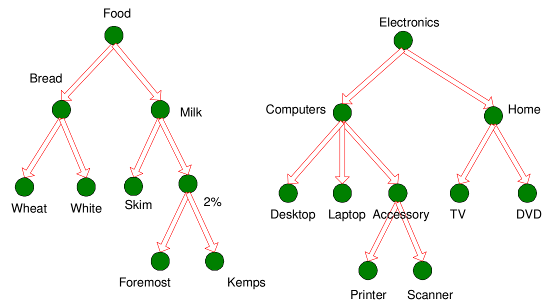
Handling a Concept Hierarchy

Multi-level Association Rules



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 an 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 X1 and X2, then $\sigma(X) \le \sigma(X1) + \sigma(X2)$
  - If $\sigma(X1 \cup Y1) \ge$ minsup,
    and X is parent of X1, Y is parent of Y1
    then $\sigma(X \cup Y1) \ge$ minsup, $\sigma(X1 \cup Y) \ge$ minsup
    $\sigma(X \cup Y) \ge$ minsup
  - If conf$(X1 \Rightarrow Y1) \ge$ minconf,
    then conf$(X1 \Rightarrow Y) \ge$ 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, there are too many frequent patterns involving items from the higher levels
  - Increased dimensionality of the data

Multi-level Association Rules

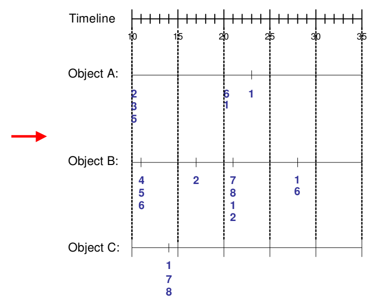
- Approach 2:
  - Generate frequent patterns at highest level first
  - Then, generate frequent patterns at the next highest level, and so on...
- Issues:
  - I/O requirements increase dramatically because we need to perform more passes over the data
  - May miss some potentially interesting cross-level association patterns

Mining Sequential Patterns

Sequence Data

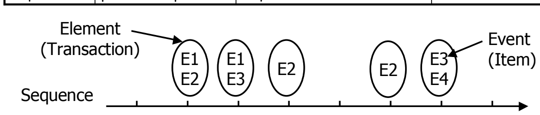
Sequence Database:

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7



Examples of Sequence Data

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



Formal Definition of a Sequence

- A sequence is an ordered list of elements (transactions)

  $s = \langle e_1 e_2 e_3 \ldots \rangle$

  – 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 Sequences

- Web sequence:

  < {Homepage} {Electronics} {Digital Cameras} {Canon Digital Camera} {Shopping Cart} {Order Confirmation} {Return to Shopping} >

- Sequence of books checked out at a library (or films rented at a video store):

  < {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 $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, …, $a_n \subseteq b_{in}$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {8} >	< {2} {3,5} >	Yes
< {1,2} {3,4} >	< {1} {2} >	No
< {2,4} {2,4} {2,5} >	< {2} {4} >	Yes

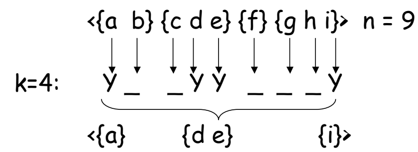
- The support of a subsequence w is defined as the fraction of data sequences that contain w
- A sequential pattern is a frequent subsequence (i.e., a subsequence whose support is ≥ *minsup*)

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?



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

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=60%
< {1} {2,3} >	s=60%
< {2} {2,3} >	s=60%
< {1,2} {2,3} >	s=60%

Extracting Sequential Patterns: Brute-force

- Given n events: $i_1, i_2, i_3, \ldots, i_n$
- Candidate 1-subsequences:
  $<\{i_1\}>, <\{i_2\}>, <\{i_3\}>, \ldots, <\{i_n\}>$
- Candidate 2-subsequences:
  $<\{i_1, i_2\}>, <\{i_1, i_3\}>, \ldots, <\{i_1\} \{i_1\}>, <\{i_1\} \{i_2\}>, \ldots, <\{i_{n-1}\} \{i_n\}>$
- Candidate 3-subsequences:
  $<\{i_1, i_2, i_3\}>, <\{i_1, i_2, i_4\}>, \ldots, <\{i_1, i_2\} \{i_1\}>, <\{i_1, i_2\} \{i_2\}>, \ldots,$
  $<\{i_1\} \{i_1, i_2\}>, <\{i_1\} \{i_1, i_3\}>, \ldots, <\{i_1\} \{i_1\} \{i_1\}>, <\{i_1\} \{i_1\} \{i_2\}>, \ldots$

Candidate Generation Algorithm

- Base case (k=2):
  - Merging two frequent 1-sequences <{i1}> and <{i2}> will produce two candidate 2-sequences: <{i1} {i2}> and <{i1 i2}>
- General case (k>2):
  - Two frequent (k-1)-sequences w1 and w2 are merged together to produce a candidate k-sequence if the subsequence obtained by removing the first event in w1 is the same as the subsequence obtained by removing the last event in w2
    - The resulting candidate after merging is given by the sequence w1 extended with the last event of w2.
      - If the last two events in w2 belong to the same element, then the last event in w2 becomes part of the last element in w1
      - Otherwise, the last event in w2 becomes a separate element appended to the end of w1

Candidate Generation Examples

- Merging the sequences
  w_1 = <{1} {2 3} {4}> and w_2 = <{2 3} {4 5}>
  will produce the candidate sequence <{1} {2 3} {4 5}> because the last two events in w_2 (4 and 5) belong to the same element
- Merging the sequences
  w_1 = <{1} {2 3} {4}> and w_2 = <{2 3} {4} {5}>
  will produce the candidate sequence <{1} {2 3} {4} {5}> 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 = <{1} {2 6} {4}> and w_2 = <{1} {2} {4 5}>
  to produce the candidate <{1} {2 6} {4 5}> because if the latter is a viable candidate, then it can be obtained by merging w_1 with <{1} {2 6} {5}>

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*

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} >

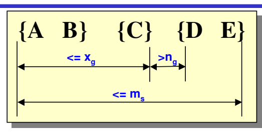
Candidate Generation

< {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)



x_g: max-gap

n_g: min-gap

m_s: maximum span

$x_g = 2$, $n_g = 0$, $m_s = 4$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {4,7} {4,5} {8} >	< {6} {5} >	Yes
< {1} {2} {3} {4} {5} >	< {1} {4} >	No
< {1} {2,3} {3,4} {4,5} >	< {2} {3} {5} >	Yes
< {1,2} {3} {2,3} {3,4} {2,4} {4,5} >	< {1,2} {5} >	No

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 the Apriori principle still hold?

Apriori Principle for Sequence Data

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

Suppose:

$x_g = 1$ (max-gap)

$n_g = 0$ (min-gap)

$m_s = 5$ (maximum span)

minsup = 60%

<{2} {5}> support = 40%

but

<{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:

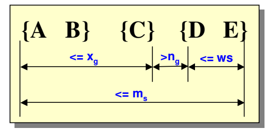
1. s is obtained from w by deleting an item from either e_1 or e_k
2. s is obtained from w by deleting an item from any element e_i that contains 2 or more items
3. s is a contiguous subsequence of s' and s' is a contiguous subsequence of w (recursive definition)

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)



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

$x_g = 2$, $n_g = 0$, ws = 1, $m_s = 5$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {4,7} {4,6} {8} >	< {3} {5} >	No
< {1,2,3,4} {5} {6} >	< {1,4} {5} >	No
< {1,2} {2,3} {3,4} {4,5} >	< {1,2} {3,4} >	Yes

Modified Support Counting Step

Given a candidate pattern: <{a, c}>

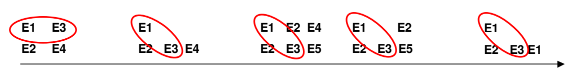
Any data sequences that contain

<… {a c} … >,
<… {a} … {c}…> (where time({c}) – time({a}) ≤ ws)
<…{c} … {a} …> (where time({a}) – time({c}) ≤ ws)

will contribute to the support count of the 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>