DATA MINING - 1DL105, 1DL111

Fall 2007

An introductory class in data mining

http://user.it.uu.se/~udbl/dut-ht2007/
alt. http://www.it.uu.se/edu/course/homepage/infoutv/ht07

Kjell Orsborn
Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden

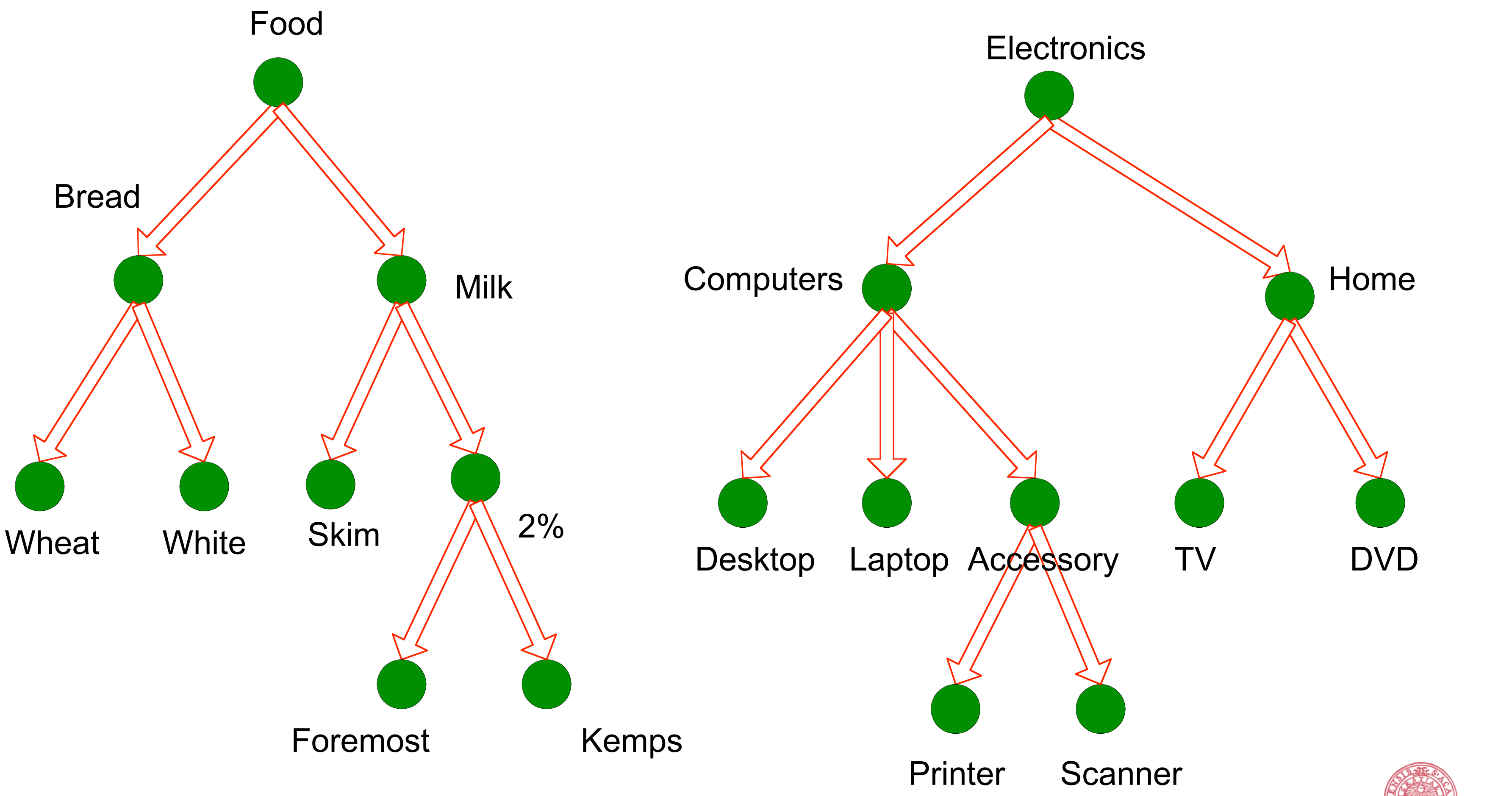
Data Mining
Association Rules: Advanced Concepts and Algorithms

(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 X1 and X2, then
    $\sigma(X) \geq \sigma(X1) + \sigma(X2)$
  - If $\sigma(X1 \cup Y1) \geq minsup$,
    and X is parent of X1, Y is parent of Y1
    then $\sigma(X \cup Y1) \geq minsup$, $\sigma(X1 \cup Y) \geq minsup$
    $\sigma(X \cup Y) \geq minsup$
  - If $conf(X1 \Rightarrow Y1) \geq minconf$,
    then $conf(X1 \Rightarrow Y) \geq minconf$

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

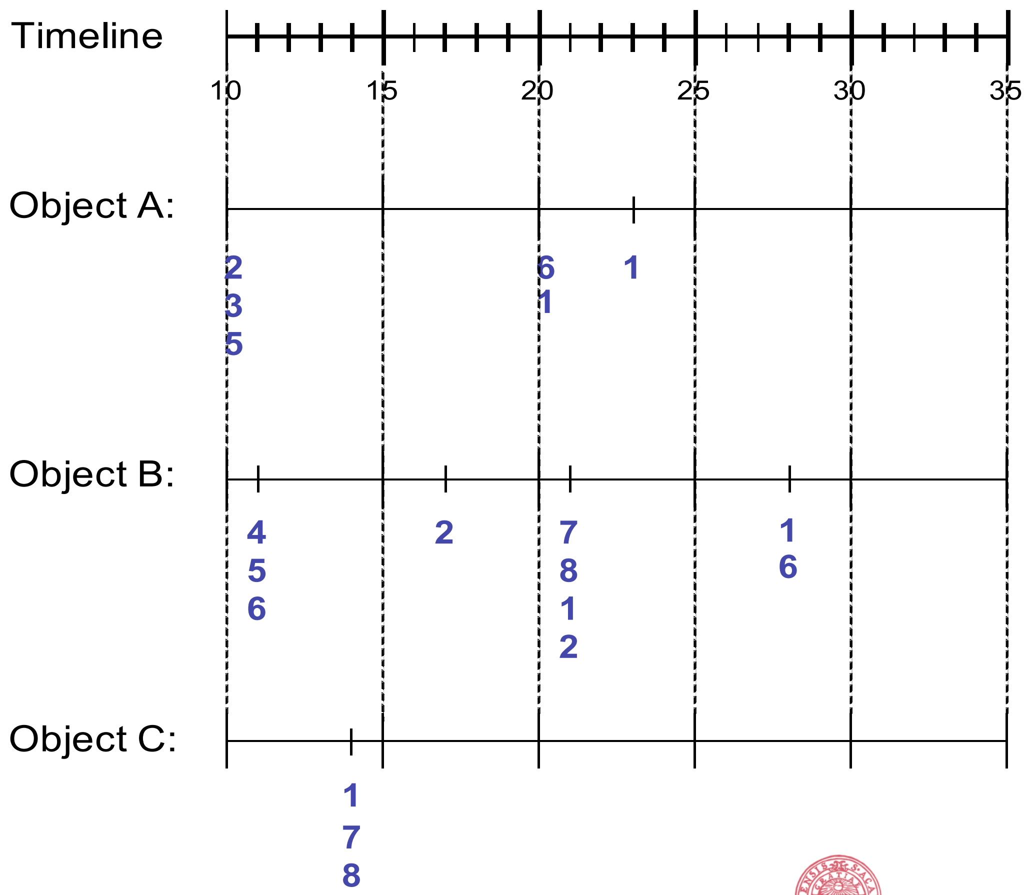
Multi-level association rules

- 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:

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

Sequence 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)
  - $$s = < e_1\ e_2\ e_3\ \ldots >$$
  - 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 $i_1 < i_2 < \ldots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i1}$, …, $a_n \subseteq b_{in}$

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?

```
        <{a  b} {c d  e} {f} {g h  i}>  n = 9

k=4:      Y  _   _ Y  Y   _   _ _  Y

        <{a}       {d  e}          {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

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

- 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 w1 is merged with another frequent (k-1)-sequence w2 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
  w1=<{1} {2 3} {4}> and w2 =<{2 3} {4 5}>
  will produce the candidate sequence < {1} {2 3} {4 5}> because the last two events in w2 (4 and 5) belong to the same element

- Merging the sequences
  w1=<{1} {2 3} {4}> and w2 =<{2 3} {4} {5}>
  will produce the candidate sequence < {1} {2 3} {4} {5}> because the last two events in w2 (4 and 5) do not belong to the same element

- We do not have to merge the sequences
  w1 =<{1} {2 6} {4}> and w2 =<{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 w1 with
  < {2 6} {4 5}>

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} >

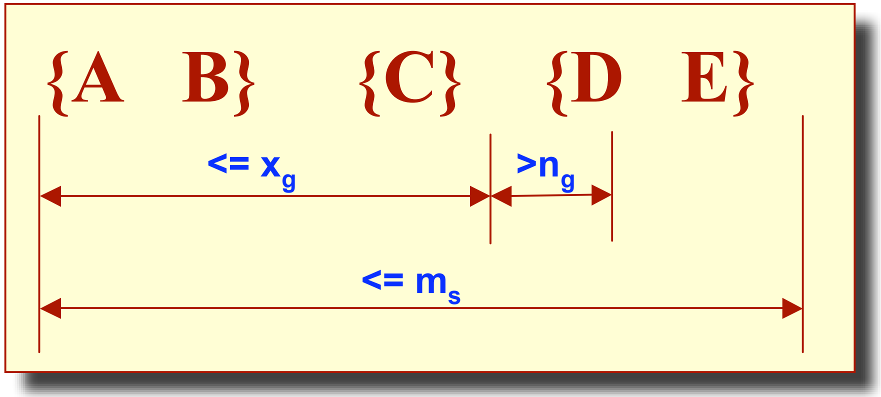
Candidate Generation

< {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 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

  $w = <e_1>< e_2>\ldots< e_k>$

  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

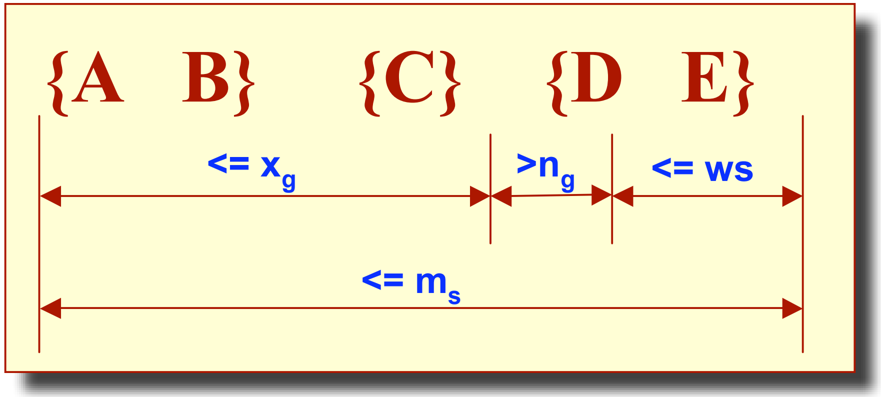
    < {1} {2 3}>, < {1 2} {2} {3}>, and < {3 4} {1 2} {2 3} {4} >
  - is not a contiguous subsequence of

    < {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}>	< {1,2} {3} >	Yes
< {1,2} {2,3} {3,4} {4,5}>	< {1,2} {3,4} >	Yes

Modified support counting step

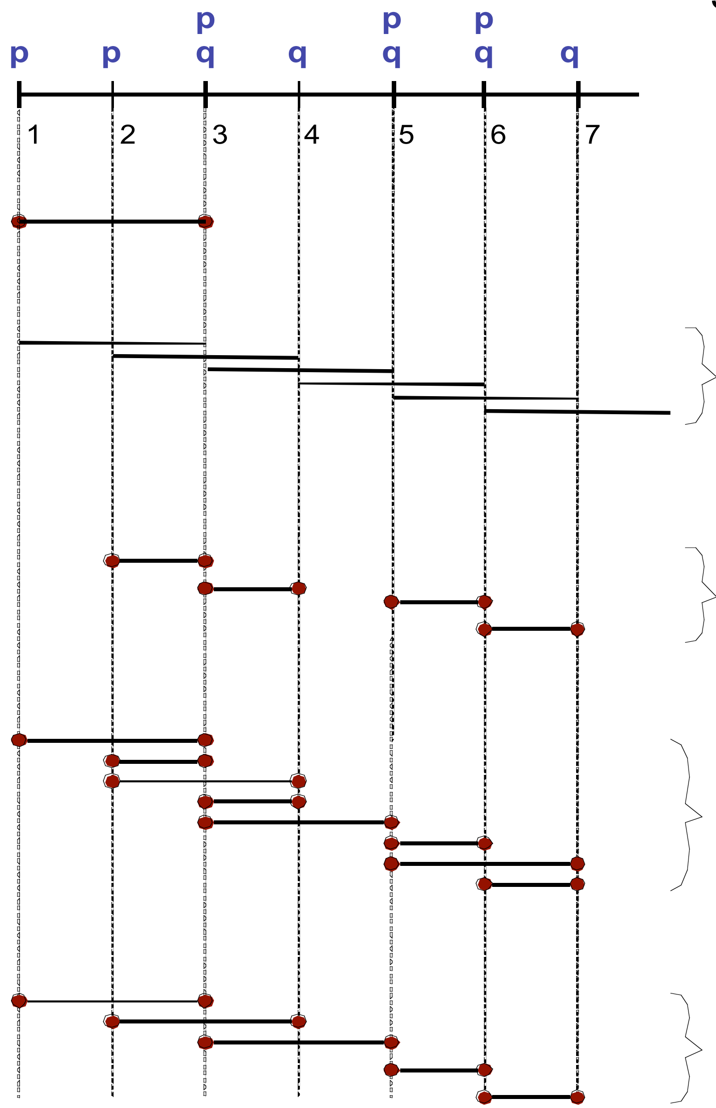
- Given a candidate pattern: <{a, c}>
  - Any data sequences that contain

    <… {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 candidate pattern

General support counting schemes

Object's Timeline



Sequence: (p) (q)

Method	Support Count
COBJ	1
CWIN	6
CMINWIN	4
CDIST_O	8
CDIST	5

Assume:

x_g = 2 (max-gap)

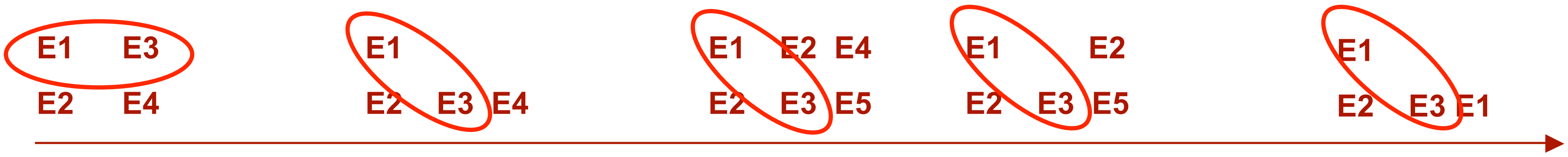
n_g = 0 (min-gap)

ws = 0 (window size)

m_s = 2 (maximum span)

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>