Data Mining2 – Advanced Aspects and Applications

Fosca Giannotti and Mirco Nanni

Pisa KDD Lab, ISTI-CNR & Univ. Pisa

http://www-kdd.isti.cnr.it/

DIPARTIMENTO DI INFORMATICA - Università di Pisa

anno accademico 2013/2014

Sequential Pattern Mining (revisited)

Sequential Patterns- module outline

- What are Sequential Patterns(SP) and what are they used for
- From Itemset to sequences
- Formal Definiton
- Computing Sequential Patterns
- Timing Constraints

From Itemset to sequences

Given: A Transaction Database

{ cid, tid, date, item }

Find: inter-transaction patterns among customers

Example: customers typically rent " Star Wars", then "Empire Strikes Back" and then "Return of the Jedi"

Sequential Patterns

cid	*tid*	*date*	*item*
1	1	01/01/2000	30
1	2	01/02/2000	90
2	3	01/01/2000	40,70
2	4	01/02/2000	30
2	5	01/03/2000	40,60,70
3	6	01/01/2000	30,50,70
4	7	01/01/2000	30
4	8	01/02/2000	40,70
4	9	01/03/2000	90
5	10	01/01/2000	90

Sequential Patterns

Itemset : is a non-empty set of items,

e.g., {30} , {40, 70}.

Sequence: is an ordered list of itemsets,

e.g. <{30} {40,70}> , <{40,70} {30} >.

Size of sequence is the number of itemsets in that sequence.

Sequential Patterns

cid	*tid*	*date*	*item*
1	1	01/01/2000	30
1	2	01/02/2000	90
2	3	01/01/2000	40,70
2	4	01/02/2000	30
2	5	01/03/2000	40,60,70
3	6	01/01/2000	30,50,70
4	7	01/01/2000	30
4	8	01/02/2000	40,70
4	9	01/03/2000	90
5	10	01/01/2000	90

Each transaction of a customer can be viewed as an itemset

A customer's sequences contains the customer's ordered itemsets

Sequential Patterns

cid	*customer sequence*
1	**<{30} {90} >**
2	**<{40,70} {30} {40,60,70}>**
3	**<{30,50,70}>**
4	**<{30} {40,70} {90}>**
5	**<{90}>**

Sequential Patterns

Sequence $<a_1\ a_2\ \ldots .a_n>$ is contained in sequence $<b_1\ b_2\ \ldots .b_m>$ if there exist indexes $i1<i2\ldots .<in$ such that

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

E.g., $<\{3\}\ \{4,5\}\ \{8\}>$ is contained in $<\{3,8\}\ \{4,5,6\}\ \{8\}>$

Is $<\{3\}\ \{4,5\}\ \{8\}>$ contained in $<\{7\}\ \{3,8\}\ \{9\}\ \{4,5,6\}\ \{8\}>$?

Is $<\{3\}\ \{4,5\}\ \{8\}>$ contained in $<\{7\}\ \{9\}\ \{4,5,6\}\ \{3,8\}\ \{8\}>$?

Is $<\{3\}\ \{4,5\}\ \{8\}>$ contained in $<\{7\}\ \{9\}\ \{3,8\}\ \{4,5,6\}>$?

Sequential Patterns

cid	*customer sequence*
1	<{30} {90} >
2	<{40,70} {30} {40,60,70}>
3	<{30,50,70}>
4	<{30} {40,70} {90}>
5	<{90}>

A customer supports sequence s if s is contained in the sequence for this customer.

E.g., customers 1 and 4 support sequence <{30} {90}>

Sequential Patterns

cid	*customer sequence*
1	<{30} {90} >
2	<{40,70} {30} {40,60,70}>
3	<{30,50,70}>
4	<{30} {40,70} {90}>
5	<{90}>

The support for a sequence s is defined as the fraction of total customers who support s .

E.g., customers 1 and 4 support sequence <{30} {90}>

Supp(<{30} {90}>) = 2/5 = 40%

Sequential Patterns

cid	*customer sequence*
1	**<{30} {90} >**
2	**<{40,70} {30} {40,60,70}>**
3	**<{30,50,70}>**
4	**<{30} {40,70} {90}>**
5	**<{90}>**

Supp(<{40,70}>) = 2/5 = 40%

Supp({40,70}) = 3/10 = 30%

Sequences & Supports (intuition)

$\langle I_1, I_2, \ldots, I_n \rangle$ is contained in $\langle J_1, J_2, \ldots, J_m \rangle$
If there exist $h_1 < \ldots < h_n$ such that

$$I_1 \subseteq J_{h1}, \ldots, I_n \subseteq J_{hn}$$

< {30}, {90} > is contained in < {30}, {40,70}, {90} >
< {30}, {40,70} > is contained in < {10,20}, {30}, {40,50,60,70} >
and in < {30}, {40,70}, {90} >

$$\text{Support}(s) = \frac{|\{ c \mid s \text{ contained in } seq(c) \}|}{\text{number of clients}}$$

Support(< {20}, {70} >) = 40%
Supporto(< {90} >) = 60%

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 $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i1}, \ldots, a_n \subseteq b_{in}$

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: Definition

- Given:
  - a database of sequences
  - a user-specified minimum support threshold, *minsup*
- Task:
  - Find all subsequences with support ≥ *minsup*
- **Apriori property: if a sequence is large then all sequences contained in that sequence should be large**.

Sequential Pattern Mining: Challenge

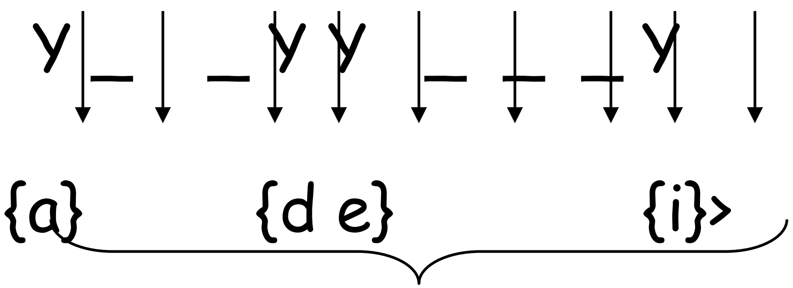
- Given a sequence: <{a b} {c d e} {f} {g h i}>
  - Examples of subsequences:

    <{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:



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: $i_1, i_2, i_3, \ldots, i_n$
- 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:

  $\langle\{i_1, i_2, i_3\}\rangle, \langle\{i_1, i_2, i_4\}\rangle, \ldots, \langle\{i_1, i_2\}\{i_1\}\rangle, \langle\{i_1, i_2\}\{i_2\}\rangle, \ldots,$

  $\langle\{i_1\}\{i_1, i_2\}\rangle, \langle\{i_1\}\{i_1, i_3\}\rangle, \ldots, \langle\{i_1\}\{i_1\}\{i_1\}\rangle, \langle\{i_1\}\{i_1\}\{i_2\}\rangle, \ldots$

Sequential Pattern Mining Algorithms

- Concept introduction and an initial Apriori-like algorithm
  - Agrawal & Srikant. Mining sequential patterns, ICDE' 95
- Apriori-based method: GSP (Generalized Sequential Patterns: Srikant & Agrawal @ EDBT' 96)
- Pattern-growth methods: FreeSpan & PrefixSpan (Han et al.@KDD' 00; Pei, et al.@ICDE' 01)
- Vertical format-based mining: SPADE (Zaki@Machine Leanining' 00)
- Constraint-based sequential pattern mining (SPIRIT: Garofalakis, Rastogi, Shim@VLDB' 99; Pei, Han, Wang @ CIKM' 02)
- Mining closed sequential patterns: CloSpan (Yan, Han & Afshar @SDM' 03)

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=<{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}>

Time constraints (1)

- Time Constraints (limite di tempo tra due transazioni)

$\langle I_1, I_2, \ldots, I_n \rangle$ è contenuta in $\langle J_1, J_2, \ldots, J_m \rangle$

se esistono $h_1 < \ldots < h_n$ per cui

$I_1 \subseteq J_{h1}, \ldots, I_n \subseteq J_{hn}$

mingap < transaction-time(J_{hi}) - transaction-time(J_{hi-1}) < maxgap

per i = 2..n

Sequential Pattern Mining:
Cases and Parameters

- Time interval, int, between events in the discovered pattern
  - int = 0: no interval gap is allowed, i.e., only strictly consecutive sequences are found
    - Ex. "Find frequent patterns occurring in consecutive weeks"
  - min_int ≤ int ≤ max_int: find patterns that are separated by at least min_int but at most max_int
    - Ex. "If a person rents movie A, it is likely she will rent movie B within 30 days" (int ≤ 30)
  - int = c ≠ 0: find patterns carrying an exact interval
    - Ex. "Every time when Dow Jones drops more than 5%, what will happen exactly two days later?" (int = 2)

Time constraints (2)

- Maximum Span

$\langle I_1, I_2, \ldots, I_n \rangle$ è contenuta in $\langle J_1, J_2, \ldots, J_m \rangle$

se esistono $h_1 < u_1 < \ldots < h_n < u_n$ per cui

$$I_1 \subseteq J_{h1}, \ldots, I_n \subseteq J_{hn}$$

$$\text{transaction-time}(J_{hn}) - \text{transaction-time}(J_{h1}) < \text{maxspan}$$

$\langle \{30\}, \{70\} \rangle$ è contenuta in $\langle \{30\}, \{40\}, \{70\}, \{20,50\} \rangle$

se transaction-time({70}) - transaction-time({30}) < maxspan

Time constraints (3)

- Sliding Windows (transazione contenuta in più transazioni)

$\langle I_1, I_2, \ldots, I_n \rangle$ è contenuta in $\langle J_1, J_2, \ldots, J_m \rangle$

se esistono $h_1 < u_1 < \ldots < h_n < u_n$ per cui

$I_1 \subseteq \bigcup_{k = h1..u1} J_k, \ldots, I_n \subseteq \bigcup_{k = hn..un} J_k$

$\text{transaction-time}(J_{ui}) - \text{transaction-time}(J_{hi}) < \text{window-size}$ per $i = 1..n$

< {30}, {40,70} > è contenuta in < {30}, {40}, {70} >

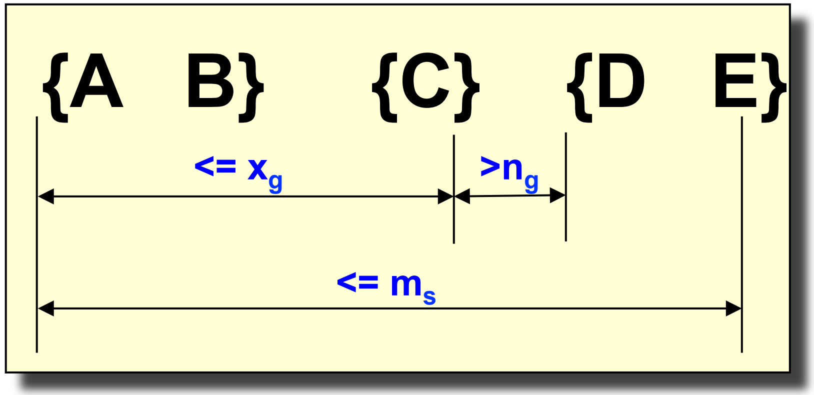
se transaction-time({70}) - transaction-time({40}) < window-size

Sequential Pattern Mining:
Cases and Parameters

- Duration of a time sequence *T*
  - Sequential pattern mining can then be confined to the data within a specified duration
  - Ex. Subsequence corresponding to the year of 1999
  - Ex. Partitioned sequences, such as every year, or every week after stock crashes, or every two weeks before and after a volcano eruption

- Event folding window *w*
  - If $w = T$, time-insensitive frequent patterns are found
  - If $w = 0$ (no event sequence folding), sequential patterns are found where each event occurs at a distinct time instant
  - If $0 < w < T$, sequences occurring within the same period *w* are folded in the analysis

Timing Constraints (I)



x_g: max-gap

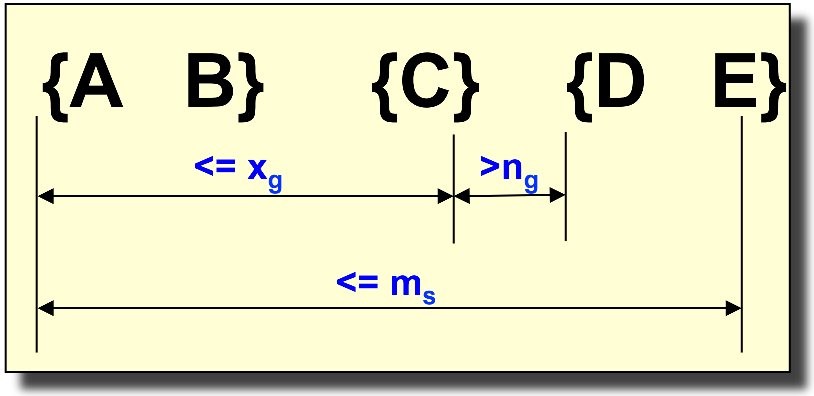
n_g: min-gap

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

Exercise(I)



x_g: max-gap

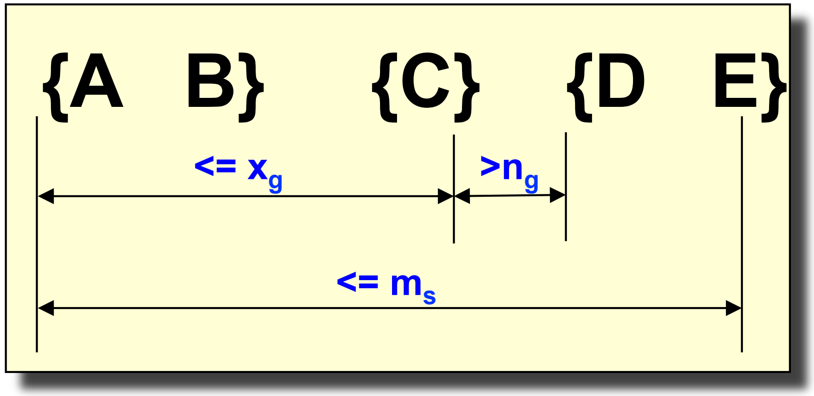
n_g: min-gap

m_s: maximum span

$x_g = 3$, $n_g = 0$, $m_s = 5$

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

Exercise(II)



x_g: max-gap

n_g: min-gap

m_s: maximum span

$x_g = 2$, $n_g = 0$, $m_s = 6$, Ws=1

Data sequence	Subsequence	Contain?
< {A,B} {C,D} {A,B} {C,D} {A,B} {C,D} >	< {A,B,C,D} >	yes
< {A,B} {C,D} {A,B} {C,D} {A,B} {C,D} >	< {A} {B,C,D} {A} >	No
< {A,B} {C,D} {A,B} {C,D} {A,B} {C,D} >	< {B,C} {A,D} {B,C} }>	No
< {A,B} {C,D} {A,B} {C,D} {A,B} {C,D} }>	< {A,B,C,D} {A,B,C,D >	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:

  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

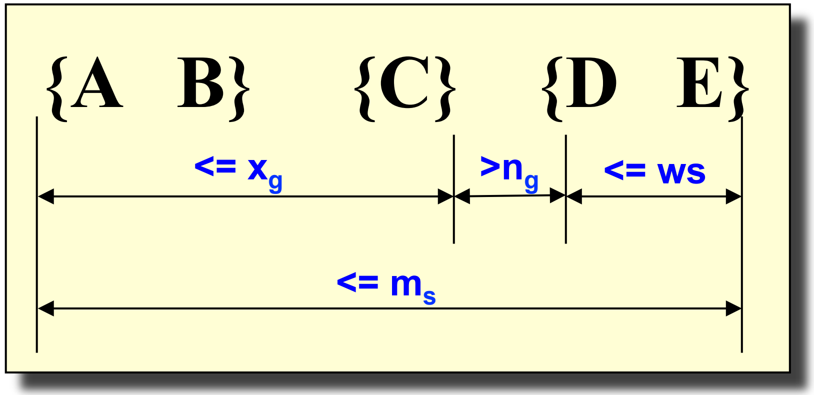
    < {1} {3} {2}> and < {2} {1} {3} {2}>

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)



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} {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

Ref: Mining Sequential Patterns

- R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. EDBT'96.
- H. Mannila, H Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. DAMI:97.
- M. Zaki. SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Learning, 2001.
- J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. ICDE'01 (TKDE'04).
- J. Pei, J. Han and W. Wang, Constraint-Based Sequential Pattern Mining in Large Databases, CIKM'02.
- X. Yan, J. Han, and R. Afshar. CloSpan: Mining Closed Sequential Patterns in Large Datasets. SDM'03.
- J. Wang and J. Han, BIDE: Efficient Mining of Frequent Closed Sequences, ICDE'04.
- H. Cheng, X. Yan, and J. Han, IncSpan: Incremental Mining of Sequential Patterns in Large Database, KDD'04.
- J. Han, G. Dong and Y. Yin, Efficient Mining of Partial Periodic Patterns in Time Series Database, ICDE'99.
- J. Yang, W. Wang, and P. S. Yu, Mining asynchronous periodic patterns in time series data, KDD'00.