BITS Pilani
Hyderabad Campus

BITS Pilani

Dr. Manik Gupta
Assistant Professor
Department of CSIS

innovate achieve lead

BITS Pilani
Hyderabad Campus

Data Mining (CS F415)
Lecture 14 – Sequential Pattern Mining

Thursday, 13th February 2020

Today's Agenda

- Sequential Pattern Mining

Why Sequential Pattern Mining?

- Sequential pattern mining: Finding time-related or event based frequent patterns (frequent subsequences)
- Most data and applications are time-related
  - Customer shopping patterns, telephone calling patterns
    - E.g., first buy computer, then CD-ROMS, software, within 3 mos.
  - Natural disasters (e.g., earthquake, hurricane)
  - Disease and treatment
  - Stock market fluctuation
  - Weblog click stream analysis
  - DNA sequence analysis

Examples of Sequence

- Sequence of different transactions by a customer at an online store:
  - < {Digital Camera,iPad} {memory card} {headphone,iPad cover} >
- Sequence of initiating events causing the nuclear accident at 3-mile Island:
  - < {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}>
- Can you think of other examples?

Sequential Pattern Discovery- Examples

- In telecommunications alarm logs
  - Inverter_Problem

  (Excessive_Line_Current) (Rectifier_Alarm) --> (Fire_Alarm)

- In point-of-sale transaction sequences
  - Computer Bookstore

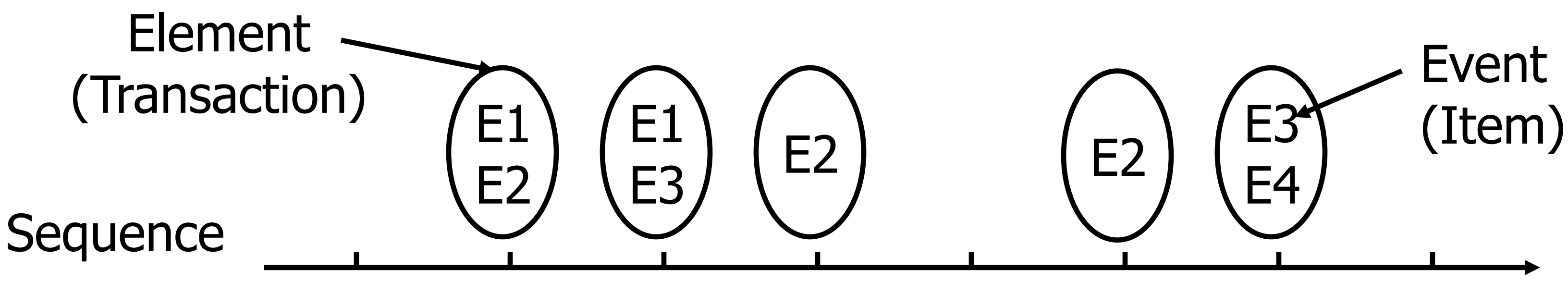
  (Intro_To_Visual_C) (C++_Primer) -->
  (Perl_for_dummies,Tcl_Tk)
  - Athletic Apparel Store

  (Shoes) (Racket, Racketball) --> (Sports_Jacket)

Why is sequential pattern mining important?

- Helps identify recurring features of a dynamic system
- Predict future occurrences of certain events

Sequence Data - Terminology



- A sequence is an ordered list of elements.
- Each element is a collection of one or more events.
- A sequence can be characterized by its length and number of occurring events.
- Length of sequence corresponds to number of elements present in the sequence
- k-sequence is a sequence that contains k events

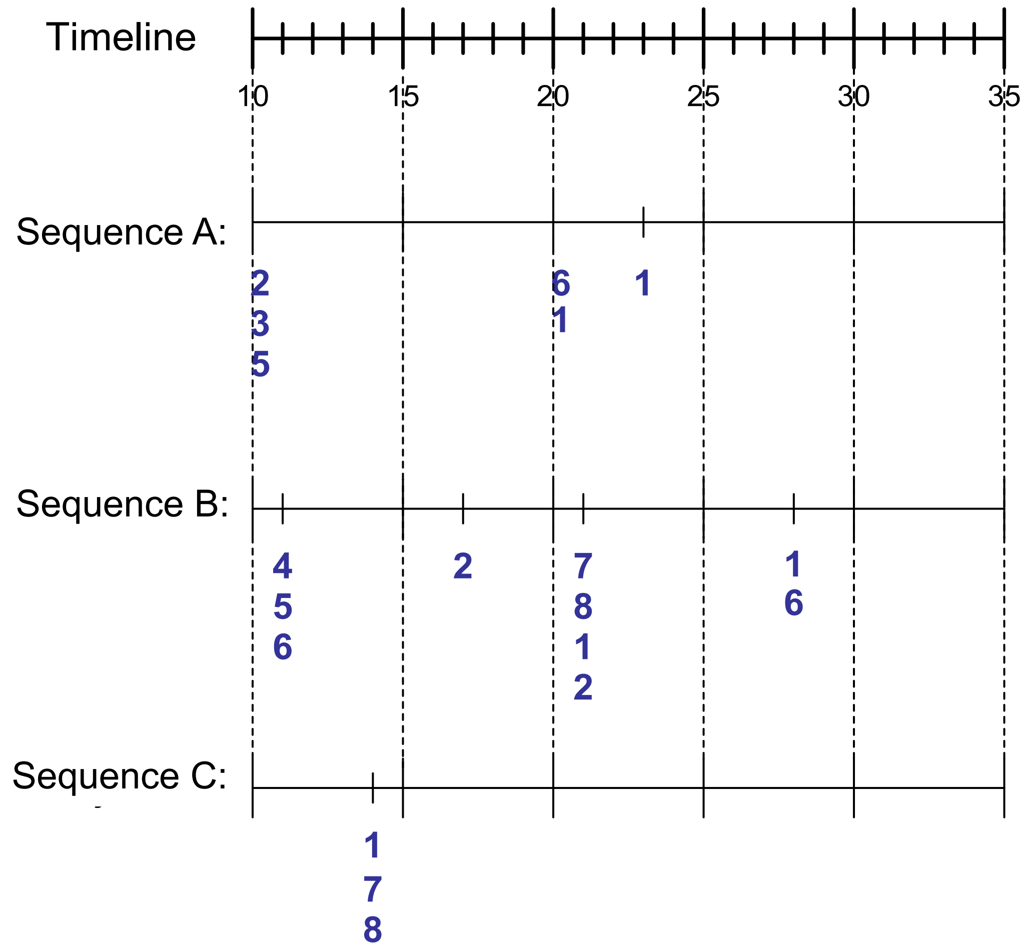
Example of elements and events in sequence data sets

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

Sequence Data

Sequence Database

Sequence ID	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



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 $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, …, $a_n \subseteq b_{in}$
- Illustrative Example:

s:	b_1	b_2	b_3	b_4	b_5
t:		a_1	a_2		a_3

t is a subsequence of s if $a_1 \subseteq b_2$, $a_2 \subseteq b_3$, $a_3 \subseteq b_5$.

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

Sequential Pattern: Definition

- Say D is a dataset that contains one or more data sequences.
  - Data sequence refers to an ordered list of events associated with a single data object
- 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 Discovery: Definition

- Given:
  - a database of sequences
  - a user-specified minimum support threshold, *minsup*
- Task:
  - Find all subsequences with support ≥ *minsup*

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%

Why is sequence pattern discovery challenging?

- Sequence pattern discovery is a computationally challenging task.
- What is the number of distinct sequences for a data sequence with 9 events?

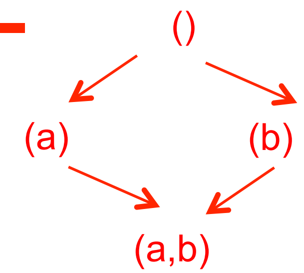
Extracting Sequential Patterns

- 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:
  - $<\{i_1, i_2\}>, <\{i_1, i_3\}>, \ldots,$
  - $<\{i_1\} \{i_1\}>, <\{i_1\} \{i_2\}>, \ldots, <\{i_n\} \{i_n\}>$
- Candidate 3-subsequences:
  - $<\{i_1, i_2, i_3\}>, <\{i_1, i_2, i_4\}>, \ldots,$
  - $<\{i_1, i_2\} \{i_1\}>, <\{i_1, i_2\} \{i_2\}>, \ldots,$
  - $<\{i_1\} \{i_1, i_2\}>, <\{i_1\} \{i_1, i_3\}>, \ldots,$
  - $<\{i_1\} \{i_1\} \{i_1\}>, <\{i_1\} \{i_1\} \{i_2\}>, \ldots$

What do you notice?

- An event can appear more than once in a sequence.
  - There are many candidate 2-sequences such as $<\{i_1,i_2\}>$, $<\{i_1\},\{i_1\}>$, $<\{i_1\},\{i_2\}>$, $<\{i_2\},\{i_1\}\}>$ that can be generated
- Order matters in sequences.
  - $<\{i_1\}, \{i_2\}>$ and $<\{i_2\}, \{i_1\}>$ correspond to different sequences and thus must be generated separately

Extracting Sequential Patterns: Simple example



Item-set patterns

- Given 2 events: a, b
- Candidate 1-subsequences:
  - <{a}>, <{b}>.
- Candidate 2-subsequences:
  - <{a} {a}>, <{a} {b}>, <{b} {a}>, <{b} {b}>, <{a, b}>.
- Candidate 3-subsequences:
  - <{a} {a} {a}>, <{a} {a} {b}>, <{a} {b} {a}>, <{a} {b} {b}>,
  - <{b} {b} {b}>, <{b} {b} {a}>, <{b} {a} {b}>, <{b} {a} {a}>
  - <{a, b} {a}>, <{a, b} {b}>, <{a} {a, b}>, <{b} {a, b}>

Sequence Pattern Mining - Recap

A sequence database

Sequence_ID	*Sequence*
1	$\langle a(abc)(ac)d(cf)\rangle$
2	$\langle (ad)c(bc)(ae)\rangle$
3	$\langle (ef)(ab)(df)cb\rangle$
4	$\langle eg(af)cbc\rangle$

How many sequences are there?

Consider the sequence <a(abc)(ac)d(c f)>. What is the length of the sequence?

Is sequence <a(bc)d f>
a subsequence of sequence 1 ?

Consider subsequence s = <(ab)c>. What is the support of s?

Different Algorithms

- GSP adopts a *candidate generate-and-test* approach using *horizonal data format* (where the data are represented as ⟨*sequence* ID : *sequence of elements*⟩, as usual, where each element is an event).
- SPADE adopts a candidate generate- and-test approach using *vertical data format* (where the data are represented as ⟨*itemset* : (*sequence ID*, *event ID*)⟩). The vertical data format can be obtained by transforming from a horizontally formatted sequence database in just one scan.

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 thaan minsup

Candidate Generation

- Base case (k=2):
  - Merging two frequent 1-sequences <{i_1}> and <{i_2}> will produce the following candidate 2-sequences: <{i_1} {i_1}>, <{i_1} {i_2}>, <{i_2} {i_2}>, <{i_2} {i_1}> 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 an event from the first element in w_1** is the **same as** the **subsequence obtained by removing an event from the last element in w_2**
    - The resulting candidate after merging is given by extending the sequence w_1 as follows-
      - If the **last element of w_2 has only one event, append it to w_1**
      - Otherwise **add the event from the last element of w_2** (which is absent in the last element of w_1) **to the last element of w_1**

GSP Example

- Merging w_1=<{1 2 3} {4 6}> and w_2 =<{2 3} {4 6} {5}> produces the candidate sequence < {1 2 3} {4 6} {5}> because the last element of w_2 has only one event
- Merging w_1=<{1} {2 3} {4}> and w_2 =<{2 3} {4 5}> produces the candidate sequence < {1} {2 3} {4 5}> because the last element in w_2 has more than one event
- Merging w_1=<{1 2 3} > and w_2 =<{2 3 4} > produces the candidate sequence < {1 2 3 4}> because the last element in w_2 has more than one event
- 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
  < {2 6} {4 5}>

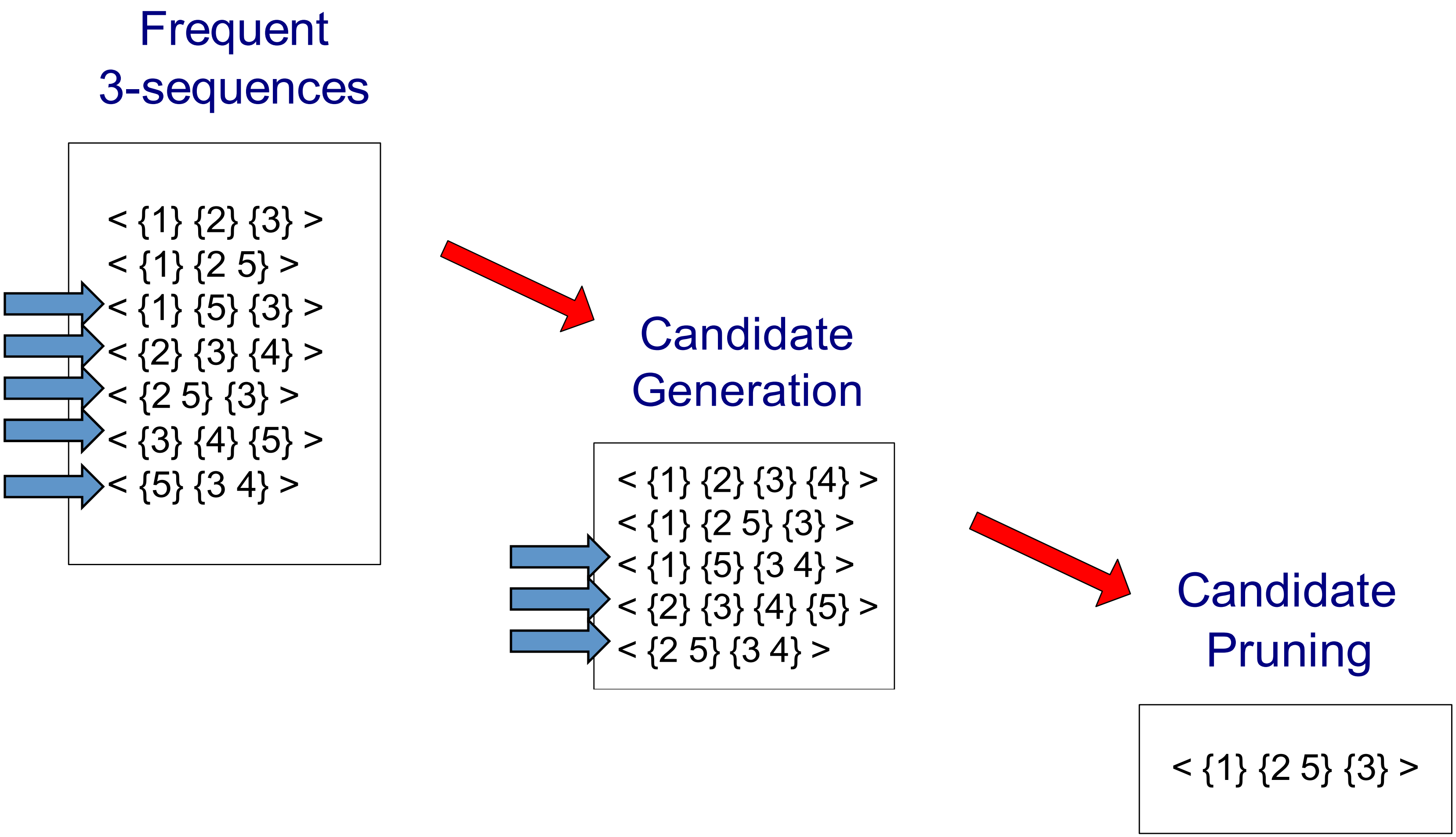
Frequent
3-sequences

< {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} >

https://www.youtube.com/watch?v=zVPg6Q6hitk



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} >

Candidate Generate-and-test - Drawbacks

- A huge set of candidate sequences generated.
  - Especially 2-item candidate sequence.
- Multiple Scans of database needed.
  - The length of each candidate grows by one at each database scan.
- Inefficient for mining long sequential patterns.
  - A long pattern grow up from short patterns
  - The number of short patterns is exponential to the length of mined patterns.

SPADE Algorithm

- SPADE (<u>S</u>equential <u>PA</u>ttern <u>D</u>iscovery using <u>E</u>quivalent Class) that uses vertical format data
- A sequence database is mapped to a set of tuples of the form ⟨*itemset* : (*sequence ID*, *event ID*)⟩
- SPADE requires one scan to find the frequent 1-sequences.
  - To find candidate 2-sequences, all pairs of single items are joined if they are frequent (therein, it applies the Apriori property), **share the same sequence identifier, and their event identifiers follow a sequential ordering.**
    - The first item in the pair must occur as an event before the second item, where both occur in the same sequence.
- The procedure stops when no frequent sequences can be found or no such sequences can be formed by such joins

Vertical format database and fragments of ID lists for 1-, 2-, and 3-sequences

Each itemset (or event) is associated with its ID list, which is the set of *SID* (*sequence ID*) and *EID* (*event ID*) pairs that contain the itemset.

SID	EID	Items
1	1	a
1	2	abc
1	3	ac
1	4	d
1	5	cf
2	1	ad
2	2	c
2	3	bc
2	4	ae
3	1	ef
3	2	ab
3	3	df
3	4	c
3	5	b
4	1	e
4	2	g
4	3	af
4	4	c
4	5	b
4	6	c

a		b		...
SID	EID	SID	EID	...
1	1	1	2	
1	2	2	3	
1	3	3	2	
2	1	3	5	
2	4	4	5	
3	2			
4	3			

2-subsequences – <{a} {a}>, <{a} {b}>, <{b} {a}>, <{b} {b}>, <{a, b}>.

Join the ID lists of a and b by joining on the same *sequence ID* wherever, according to the *event IDs*, a occurs before b. That is, the join must preserve the temporal order of the events involved.

ab			ba			...
SID	EID (a)	EID(b)	SID	EID (b)	EID(a)	...
1	1	2	1	2	3	
2	1	3	2	3	4	
3	2	5				
4	3	5				

aba				...
SID	EID (a)	EID(b)	EID(a)	...
1	1	2	3	
2	1	3	4	

https://www.youtube.com/watch?v=Zl40pWQShVI&t=8s

Advantages of SPADE

- The use of vertical data format, with the creation of ID lists, reduces scans of the sequence database.
- The **ID lists carry the information necessary to find the support of candidates**. As the length of a frequent sequence increases, the size of its ID list decreases, resulting in very fast joins.

To Explore

- PrefixSpan - Prefix-Projected Sequential Pattern Growth

Timing Constraints

Buyer A: < {TV} … {DVD Player} >

Buyer B: < {TV} … {DVD Player} >

…

- The sequential pattern of interest is

  <{TV}{DVD Player}>

which suggests that people who buy TV will also soon buy DVD player.

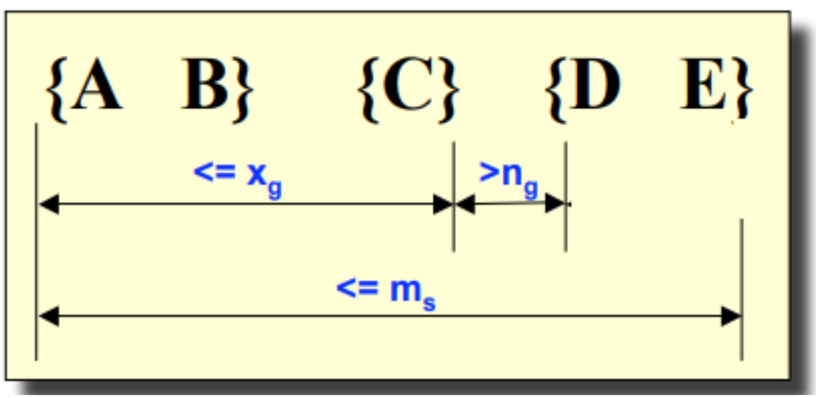
- A person who bought a TV ten years earlier should not be considered as supporting the pattern because the time gap between the purchases is too long.

Timing Constraints

- In some applications, relative timing of the transactions is crucial to define the pattern
- Consider a credit card company wanting to mine unusual patterns in purchasing behavior:
  - A fraudulent user of the card could easily buy similar items as the normal users would do, so the sequence of transactions might not discriminate enough
  - But the fraudulent user would do the purchases in short time interval to make maximum use of the card before it is close
- Constraining the patterns in temporal dimension is required to mine such patterns

Timing Constraints

- Consider two kinds of constraints:
  - Max-span constraint (m_s): maximum allowed time between the first element and the last element in the sequence
  - Max-gap constraint (x_g): maximum length of a gap between two consecutive element

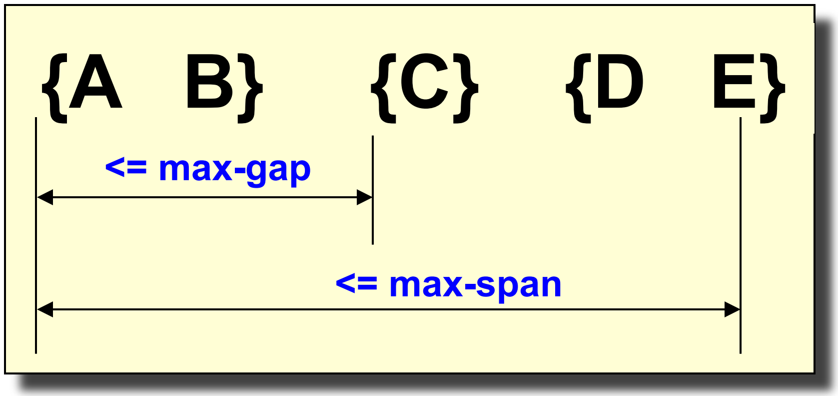


x_g: max-gap

n_g: min-gap

m_s: maximum span

Timing Constraints



Consider the data sequences below with element timestamps 1,2,3,...
max-gap = 2, max-span= 4

Data sequence	Subsequence	Contained?
<{2,4} {3,5,6} {4,7} {4,5} {8}>	< {6} {5} >	Yes
<{1} {2} {3} {4} {5}>	< {1} {4} >	No (max-gap=3)
<{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 (max-span=5)

Mining Sequential Patterns with Timing Constraints

Approach 1:

- Mine sequential patterns without timing constraints
- Postprocess the discovered patterns

Approach 2:

- Modify algorithm 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)

m_s = 5 (maximum span)

minsup = 60%

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

but

<{2} {3} {5}> support = 60%

Problem exists because of max-gap constraint!

Support has increased when the number of events increases in a sequence - contradicts Apriori principle!

Object D does not support <{2},{5}> since time gap between events 2 and 5 is greater than maxgap.

Contiguous Subsequences

- This problem caused by the maxgap constraint can be circumvented by considering contiguous subsequences
- 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}>
- A k-1-sequence t is a contiguous subsequence of k-sequence s if t can be constructed by
  - deleting events from the elements of s
  - while not allowing middle elements to get empty

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)

Data Sequence, s	Sequential Pattern, t	Is t a contiguous subsequence of s?
$<\{1\}\ \{2,3\}>$	$<\{1\}\ \{2\}>$	Yes
$<\{1,2\}\ \{2\}\ \{3\}>$	$<\{1\}\ \{2\}>$	Yes
$<\{3,4\}\ \{1,2\}\ \{2,3\}\ \{4\}>$	$<\{1\}\ \{2\}>$	Yes
$<\{1\}\ \{3\}\ \{2\}>$	$<\{1\}\ \{2\}>$	No
$<\{1,2\}\ \{1\}\ \{3\}\ \{2\}>$	$<\{1\}\ \{2\}>$	No

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.

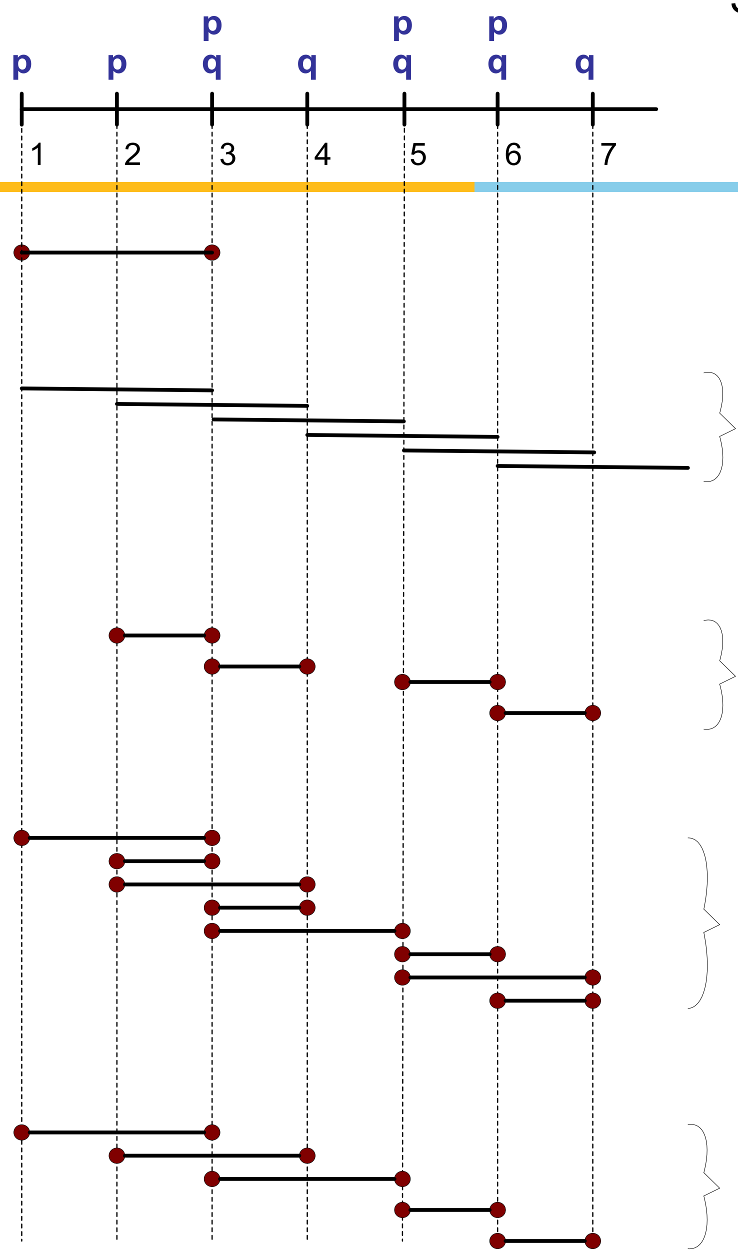
Modified Sequential Apriori for timing constraints

- Modified Apriori principle
  - If a k-sequence is frequent, then all of its contiguous k-1-subsequences are frequent
- Modified algorithm
  - **Candidate generation step remains the same**
    - Merge two frequent k-1 sequences that have the same middle part (excluding first and last event)
  - **During candidate pruning, verify only contiguous k-1-sequences**
    - e.g. Given 5-sequence: 1(23)45 we need to verify 1245,1345, and need not to verify 1(23)5 since maxgap between (23) and 5 is greater than one time unit

Support of a sequential pattern

- Support of a sequential pattern is not as clear cut as itemset support, due to the repetition of the items in the data sequence
- Many support counting methods:
  - **COBJ**: Atleast one occurrence of a given sequence in an object's timeline.
  - **CWIN**: One occurrence per sliding window.
    - A sliding time window of fixed length (maxspan) is moved across an object's timeline, one unit at a time. The support is incremented each time the sequence is encountered in the sliding window.
  - **CMINWIN**: Number of minimal windows of occurrence
    - Minimal window is the time interval such that sequence occurs in that time interval but it does not occur in any proper subinterval of it
  - **CDIST_O**: Distinct occurrences with possibility of event-timestamp overlap.
  - **CDIST**: Distinct occurrences with no event-timestamp overlap allowed.

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 time constraints

- If the minimum time difference (mingap) is zero, then events in one element must occur immediately after the events occurring in the previous element
- Window size threshold (ws) can be defined to specify the maximum allowed time difference between the latest and earliest occurrences of events in any element of a sequential pattern.
    - A window size of 0 means all events in the same element of a pattern must occur simultaneously.

Baseline for support counting

- For frequent itemset mining, the baseline is given by the total number of transactions.
- For sequential pattern mining, the baseline depends on the counting method used.
  - For the COBJ method, the total number of objects in the input data can be used as the baseline.
  - For the CWIN and CMINWIN methods, the baseline is given by the sum of the number of time windows possible in all objects.
  - For methods such as CDIST and CDIST_O, the baseline is given by the sum of the number of distinct timestamps present in the input data of each object.

Thanks!

Next Lecture:

- Subgraph Mining

Readings:

- Chapter 7 - Tan & Kumar