Data Mining II

Sequential Patterns

Rita P. Ribeiro

2016/2017

Computer Science Department

U. PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Summary

Sequential Patterns
Basic Concepts

Motivation

- For Association Rules Mining the order of transactions is not important.
- But, for many applications the order might be significant.
  - Commerce: customer buys a computer, a printer and then photographic paper.
  - Web: sequences of pages visited are useful to find navigational patterns.
  - Text: order of words in a sentence is useful to find linguistic patterns.
  - ...

Sequences

- Given a set of items $I = \{i_1, i_2, \cdots, i_m\}$, a **sequence** is a ordered list of itemsets, i.e. $s =< a_1 a_2 \cdots a_r >$ where a_i is an itemset.
- The items in each element (event) a_i are in lexicographic order.
- A given item can occur only once in a element, but can occur multiple times in a sequence.
- Example:
  - $I = \{a, b, c, d, \cdots\}$
  - $s =< \{a, b\}, \{a, c\}, \{d\}, \{b, c, d\} >$

Sequences (cont.)

- The size of a sequence is the number of elements (itemsets) in the sequence.
- The length of a sequence os the number of items in the sequence.
- A sequence of length k is called a k-sequence
- Example:
  - $s =< \{a, b\}, \{a, c\}, \{d\}, \{b, c, d\} >$
  - $size(s) = 4$
  - $length(s) = 8$

Sequences (cont.)

- A sequence $s_1 = < a_1 a_2 \cdots a_r >$ is a subsequence of $s_2 = < b_1 b_2 \cdots b_v >$ if there exist integers $1 \leq j_1 < j_2 < \cdots < j_{r-1} < j_r \leq v$ such that $a_1 \subseteq b_{j_1}, a_2 \subseteq b_{j_2}, \cdots, a_r \subseteq b_{j_r}$
- In that case s_2 is also said to be a supersequence of s_1.
- Example
    - $s = < \{a\}, \{a, b, c\}, \{a, c\}, \{d\}, \{c, f\} >$

subsequences of s

- $< \{a\}, \{a\}, \{a, c\}, \{d\}, \{c\} >$
- $< \{a, c\}, \{a, c\}, \{d\}, \{c, f\} >$
- $< \{a\}, \{c\} >$

not subsequences of s

- $< \{d\}, \{f\}, \{c, f\} >$
- $< \{c, f\}, \{d\} >$
- $< \{a, b, c\}, \{d\}, \{c\}, \{f\} >$

Sequences (cont.)

- A sequence database consists of ordered elements or events
- Transaction database vs sequence database

Transaction database

TID	Itemsets
1	{a, b, c}
2	{a, c, d}
3	{a, d, e}
4	{b, e, f}

Sequence database

SID	Sequences
1	<{a},{a, b, c},{a,c},{d},{c,f}>
2	<{a,d},{c},{b,c},{a,e}>
3	<{e,f},{a,b},{d,f},{c},{b}>
4	<{e},{g},{a,f},{c},{b},{c}>

Sequences (cont.)

- The support of a sequence is the fraction of total data sequences that contain that sequence.

- Example

Sequence database

SID	Sequences
1	<{a},{a, b, c},{a,c},{d},{c,f}>
2	<{a,d},{c},{b,c},{a,e}>
3	<{e,f},{a,b},{d,f},{c},{b}>
4	<{e},{g},{a,f},{c},{b},{c}>

- sup(<{a}>) = 4
- sup(<{a},{c}>) = 4
- sup(<{a,b},{c}>) = 2

Sequential Pattern Mining Task

- Given
  - a sequence database *S*
  - minimum support *minsup*
- Find
  - **all** frequent sequences, i.e. sequences with support above *minsup*

Sequential Pattern Mining Task (cont.)

- Example

Transactions Database

Customer ID	Transactions
1	30
1	90
2	10,20
2	30
2	10,40,60,70
3	30,50,70,80
4	30
4	30,40,70,80
4	90
5	90

Sequences Database

Customer ID	Data Sequences
1	<{30},{90}>
2	<{10,20},{30},{10,40,60,70}>
3	<{30,50,70,80}>
4	<{30},{30,40,70,80},{90}>
5	<{90}>

	Sequential Patterns with minsup=25% (2)
1-sequences	<{30}>,<{40}>,<{70}>,<{80}>,<{90}>
2-sequences	<{30},{40}>,<{30},{70}>,<{30},{90}>, <{30,70}>,<{30,80}>,<{40,70}>,<{70,80}>
3-sequences	<{30},{40,70}>, <{30,70,80}>

Sequential Pattern Mining Task (cont.)

- Challenges
  - A huge number of possible sequential patterns are hidden in databases
  - A mining algorithm should
    - find the complete set of patterns, when possible, satisfying the minimum support (frequency) threshold
    - be highly efficient, scalable, involving only a small number of database scans
    - be able to incorporate various kinds of user-specific constraints

Mining Sequential Patterns

Methods for Mining Sequential Patterns

- Apriori-based Approaches
  - **GSP** [Srikant and Agrawal, 1996]
  - SPADE [Zaki, 2001]: vertical format-based mining
  - ...
- Pattern-Growth-based Approaches
  - **PrefixSpan** [Pei et al., 2001]
  - CloSpan [Yan et al., 2003]: mining closed sequential patterns
  - ...

GSP Algorithm

GSP: Generalized Sequential Pattern Mining Algorithm

[Srikant and Agrawal, 1996]

- very similar to Apriori
- C_k: set of all candidate k-sequences
- F_k: set of all frequent k-sequences
- uses Apriori Property to prune candidates:
    - if a sequence s is not frequent, then none of the supersequences of s is frequent.
    - Example ($minsup = 25\%(2)$)

Seq. ID	Sequence
1	<(bd)cb(ac)>
2	<(bf)(ce)b(fg)>
3	<(ah)(bf)abf>
4	<(be)(ce)d>
5	<a(bd)bcb(ade)>

<hb> is infrequent, so <hab> and <(ah)b>

(simplified notation: <a(bc)> is <{a},{b,c}>)

- Example:

SID	Data Sequences
1	<3,9>
2	<(12),3,(1467)>
3	<(3578)>
4	<3,(3478),9>
5	<9>

- $minsup = 25\%(2)$
- $C_1 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- $F_1 = \{\langle 3\rangle, \langle 4\rangle, \langle 7\rangle, \langle 8\rangle, \langle 9\rangle\}$

GSP Algorithm: identifying frequent sequences (cont.)

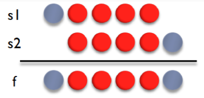
- Example:

SID	Data Sequences
1	<3,9>
2	<(12),3,(1467)>
3	<(3578)>
4	<3,(3478),9>
5	<9>

- $F_1 = \{\langle 3\rangle, \langle 4\rangle, \langle 7\rangle, \langle 8\rangle, \langle 9\rangle\}$
- $C_2 = \{\langle 3, 3\rangle, \langle 34\rangle, \langle 3, 4\rangle, \langle 4, 3\rangle, \langle 37\rangle, \langle 3, 7\rangle, \cdots\}$
- how to generate candidate sequences?

GSP Algorithm: identifying frequent sequences (cont.)

Candidate k-sequence generation



Join step

- $s_1, s_2 \in F_{k-1}$
- drop first item A in s_1 and last item Z in s_2
- if (common = s_1-A) == s_2-Z
  - candidate $c = A + common + Z$
  - two possibilities
    - Z is separate if it was separate in s_2
    - Z is added to the last set if it was part of last set in s_2

Prune step

- Candidate k-sequence is pruned if any of its (k-1)-subsequences is infrequent.

GSP Algorithm: identifying frequent sequences (cont.)

- $F_1 = \{\langle 3\rangle, \langle 4\rangle, \langle 7\rangle, \langle 8\rangle, \langle 9\rangle\}$
- Join step (no prune)
  $C_2 = \{\langle 3,3\rangle, \langle 3,4\rangle, \langle 34\rangle, \langle 3,7\rangle, \langle 37\rangle, \cdots \langle 4,3\rangle, \cancel{\langle 43\rangle}, \cdots, \langle 89\rangle\}$
- Join $\langle a\rangle$ and $\langle b\rangle$
- common part is $\langle\rangle$
- combinations $a + \langle\rangle + b$
  - b was separate in $\langle b\rangle$, so we have $\langle a, b\rangle$
  - b is also part of the last set in $\langle b\rangle$, so we have $\langle ab\rangle$
- if $a = b$ then $\langle ab\rangle$ is excluded because ab is a set.
- if we have $\langle ab\rangle$, $\langle ba\rangle$ is excluded because ab is a set.

GSP Algorithm: identifying frequent sequences (cont.)

- $F_3 = \{\langle 12, 4\rangle, \langle 12, 5\rangle, \langle 1, 45\rangle, \langle 14, 6\rangle, \langle 2, 45\rangle, \langle 2, 4, 6\rangle\}$
- Join step
  $C_4 = \{\langle 12, 4, 6\rangle, \langle 12, 45\rangle\}$
  - Join $\langle 12, 4\rangle$ and $\langle 2, 4, 6\rangle$, common part is $\langle 2, 4\rangle$
  - combination $1 + \langle 2, 4\rangle + 6$
  - 6 is separate, so we have $\langle 12, 4, 6\rangle$
  - Join $\langle 12, 4\rangle$ and $\langle 2, 45\rangle$, common part is $\langle 2, 4\rangle$
  - combination $1 + \langle 2, 4\rangle + 5$
  - 6 part of 45, so we have $\langle 12, 45\rangle$
- Prune step
  - delete each item at a tim and check if subseq is frequent
  - $\xcancel{\langle 12, 4, 6\rangle}$ because $\langle 1, 4, 6\rangle \notin F_3$
- $C_4 = \{\langle 12, 45\rangle\}$

GSP Algorithm: Exercises

1. Given $I = \{1, 2, 3, 4\}$ and $S = \{\langle 1, 2\rangle, \langle 12, 3\rangle, \langle 2, 13, 4\rangle, \langle 2, 12, 3\rangle\}$
   Determine the following sets with $minsup = 50\%$
   - C_1, F_1
   - C_2, F_2
   - C_3, F_3
2. Given the following transactions data base, mine the frequent sequences with $minsup = 70\%$

Customer ID	Itemsets
1	milk,bread
1	coffee, meat
2	water, bread
2	coffee, meat
3	milk, bread
3	coffee, soup

GSP Algorithm Bottlenecks

- GSP benefits from Apriori pruning
  - reduces search space
- GSP Bottlenecks
  - generates a huge set of candidate sequences (especially 2-item candidate sequences)
  - example: for 6 frequent 1-item sequences, it generates $6 * 6 + 6 * 5/2 = 51$ 2-item candidates sequences.

	<a>	<b>	<c>	<d>	<e>	<f>
<a>	<aa>	<ab>	<ac>	<ad>	<ae>	<af>
<b>	<ba>	<bb>	<bc>	<bd>	<be>	<bf>
<c>	<ca>	<cb>	<cc>	<cd>	<ce>	<cf>
<d>	<da>	<db>	<dc>	<dd>	<de>	<df>
<e>	<ea>	<eb>	<ec>	<ed>	<ee>	<ef>
<f>	<fa>	<fb>	<fc>	<fd>	<fe>	<ff>

	<a>	<b>	<c>	<d>	<e>	<f>
<a>		<(ab)>	<(ac)>	<(ad)>	<(ae)>	<(af)>
<b>			<(bc)>	<(bd)>	<(be)>	<(bf)>
<c>				<(cd)>	<(ce)>	<(cf)>
<d>					<(de)>	<(df)>
<e>						<(ef)>
<f>						

GSP Algorithm Bottlenecks (cont.)

- GSP bottlenecks (cont.)
  - scans the database multiple times
  - the length of each candidate grows by one at each database scan
  - a long pattern grows up from short patterns
  - an exponential number of short candidates
  - inefficient for mining long sequential patterns

PrefixSpan: Prefix-Projected Sequential Pattern Growth

[Pei et al., 2001]

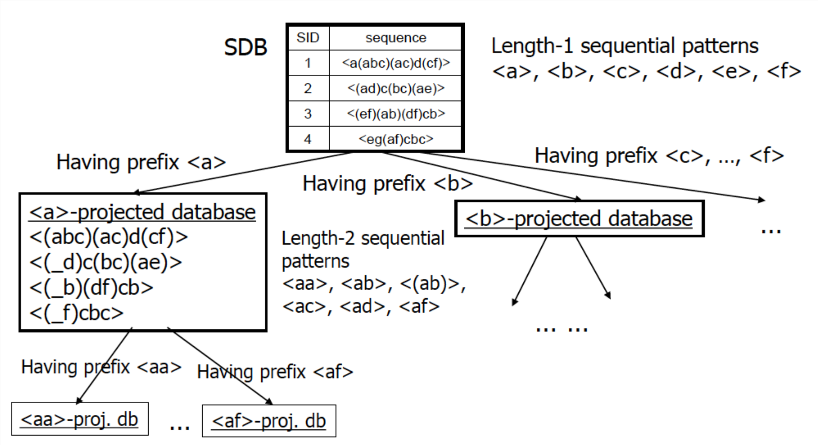
- More efficient and less memory hungry than GSP
- It does not generate candidates
- GSP performs breath-first search
- PrefixSpan performs depth-first search
- It uses prefix-based projection: less projections and quickly shrinking sequences

PrefixSpan Algorithm: Prefix and Suffix (Projection)

- Given the sequence $\langle a(abc)(ac)d(cf)\rangle$
  - $\langle a\rangle$, $\langle aa\rangle$, $\langle a(ab)\rangle$, $\langle a(abc)\rangle$ are **prefixes** of the sequence
  - $\langle ab\rangle$, $\langle a(bc)\rangle$ are **not prefixes** of the sequence
  - For a prefix, we can obtain prefix-projected suffix

Prefix	*Suffix* (Prefix-Based *Projection)*
<a>	<(abc)(ac)d(cf)>
<aa>	<(_bc)(ac)d(cf)>
<ab>	<(_c)(ac)d(cf)>

PrefixSpan Algorithm: Divide-and-Conquer Approach



PrefixSpan Algorithm: Example

- Find length-1 frequent sequential patterns

SID	sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

- $minsup = 50\%$ in support descending order
- $\langle a \rangle$:4, $\langle b \rangle$:4, $\langle c \rangle$:4, $\langle d \rangle$:3, $\langle e \rangle$:3, $\langle f \rangle$:3 $\cancel{\langle g \rangle}$:1

- Divide the search space in 6 (one for each frequent item) to find subsets of sequential patterns with:
  - prefix $\langle a \rangle$ using $\langle a \rangle$-projected database
  - prefix $\langle b \rangle$ using $\langle b \rangle$-projected database
  - ...
  - prefix $\langle f \rangle$ using $\langle f \rangle$-projected database

PrefixSpan Algorithm: Example (cont.)

- Find subsets of sequential patterns with prefix $\langle a \rangle$
  - consider only the subsequences prefixed with the first occurrence of a
  - build $\langle a \rangle$-projected database

    (a: a was a separate element; _: a was a part of the same element)

SID	sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

<a>-projected database
<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>

Support counts on the projected database:

- a: 2, b: 4, c: 4, d: 2, $\cancel{e}$: 1, f: 2
- $(\_b)$: 2, $\cancel{(\_c)}$: 1 $\cancel{(\_d)}$: 1, $\cancel{(\_f)}$: 1

- Find all length-2 frequent sequences with prefix $\langle a \rangle$:
  - $\langle aa \rangle$: 2, $\langle ab \rangle$: 4, $\langle (ab) \rangle$: 2, $\langle ac \rangle$: 4, $\langle ad \rangle$: 2, $\langle af \rangle$: 2
- We already have $\langle a \rangle$ prefixed sequences of length 1 and 2.
- Now we partition again by length-2 sequences

PrefixSpan Algorithm: Example (cont.)

- Find frequent sequences with prefix $\langle aa \rangle$
  - $\langle aa \rangle$-projected database

SID	sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

<a>-projected database

<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>

<aa>-projected database

<(_bc)(ac)d(cf)>
<(_e)>

  - $\cancel{a}$: 1, $\cancel{b}$: 0, $\cancel{c}$: 1, $\cancel{d}$: 1, $\cancel{e}$: 0, $\cancel{f}$: 1
  - $\cancel{(\_b)}$: 1, $\cancel{(\_c)}$: 1, $\cancel{(\_d)}$: 0, $\cancel{(\_e)}$: 1, $\cancel{(\_f)}$: 0
  - no hope to generate frequent $\langle aa \rangle$ prefix sequences

PrefixSpan Algorithm: Example (cont.)

- Find frequent sequences with prefix $\langle ab \rangle$
  - $\langle ab \rangle$-projected database

SID	sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

<a>-projected database

<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>

<ab>-projected database

<(_c)(ac)d(cf)>
<(_c)(ae)>
<(c)>

  - frequent "items": (_c): 2, a:2, c: 2
  - frequent $\langle ab \rangle$ prefix sequences: $\langle a(bc) \rangle$:2, $\langle aba \rangle$:2, $\langle abc \rangle$:2

- Find frequent sequences with prefix $\langle a(bc) \rangle$
  - $\langle a(bc) \rangle$-projected database

SID	sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

<a>-projected database

<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>

<ab>-projected database

<(_c)(ac)d(cf)>
<(_c)(ae)>
<(c)>

<a(bc)>-projected database

<(ac)d(cf)>
<(ae)>

  - frequent “items”: a:2
  - frequent $\langle a(bc) \rangle$ prefix sequences: $\langle a(bc)a \rangle$:2

PrefixSpan Algorithm: Example (cont.)

- Find frequent sequences with prefix $\langle aba \rangle$
  - $\langle aba \rangle$-projected database

SID	sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

<a>-projected database

<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>

<ab>-projected database

<(_c)(ac)d(cf)>
<(_c)(ae)>
<(c)>

<aba>-projected database

<(_c)d(cf)>
<(_e)>

  - no frequent “items”

PrefixSpan Algorithm: Example (cont.)

- Find frequent sequences with prefix $\langle (ab) \rangle$
  - $\langle (ab) \rangle$-projected database

SID	sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

<a>-projected database

<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>

<(ab)>-projected database

<(_c)(ac)d(cf)>
<(df)(cb)>

  - frequent "items": d:2 , c:2 , f:2
  - frequent $\langle (ab) \rangle$ prefix sequences: $\langle (ab)d \rangle$:2, $\langle (ab)c \rangle$:2, $\langle (ab)f \rangle$:2

PrefixSpan Algorithm: Example (cont.)

- Find frequent sequences with prefix $\langle (ab)d \rangle$
  - $\langle (ab)d \rangle$-projected database

SID	sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

<a>-projected database
<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>

<(ab)>-projected database
<(_c)(ac)d(cf)>
<(df)(cb)>

<(ab)d>-projected database
<(cf)>
<(_f)(cb)>

  - frequent “items”: c:2
  - frequent $\langle (ab)d \rangle$ prefix sequences: $\langle (ab)dc \rangle$:2

PrefixSpan Algorithm: Example (cont.)

- At the end, for the sequences database

Sequence_id	Sequence
10	$\langle a(abc)(ac)d(cf)\rangle$
20	$\langle (ad)c(bc)(ae)\rangle$
30	$\langle (ef)(ab)(df)cb\rangle$
40	$\langle eg(af)cbc\rangle$

- with *minsup* $=$ 50%, we get the following frequent sequences:

Prefix	Projected (postfix) database	Sequential patterns
$\langle a\rangle$	$\langle(abc)(ac)d(cf)\rangle$, $\langle(\_d)c(bc)(ae)\rangle$, $\langle(\_b)(df)cb\rangle$, $\langle(\_f)cbc\rangle$	$\langle a\rangle$, $\langle aa\rangle$, $\langle ab\rangle$, $\langle a(bc)\rangle$, $\langle a(bc)a\rangle$, $\langle aba\rangle$, $\langle abc\rangle$, $\langle(ab)\rangle$, $\langle(ab)c\rangle$, $\langle(ab)d\rangle$, $\langle(ab)f\rangle$, $\langle(ab)dc\rangle$, $\langle ac\rangle$, $\langle aca\rangle$, $\langle acb\rangle$, $\langle acc\rangle$, $\langle ad\rangle$, $\langle adc\rangle$, $\langle af\rangle$
$\langle b\rangle$	$\langle(\_c)(ac)d(cf)\rangle$, $\langle(\_c)(ae)\rangle$, $\langle(df)cb\rangle$, $\langle c\rangle$	$\langle b\rangle$, $\langle ba\rangle$, $\langle bc\rangle$, $\langle(bc)\rangle$, $\langle(bc)a\rangle$, $\langle bd\rangle$, $\langle bdc\rangle$, $\langle bf\rangle$
$\langle c\rangle$	$\langle(ac)d(cf)\rangle$, $\langle(bc)(ae)\rangle$, $\langle b\rangle$, $\langle bc\rangle$	$\langle c\rangle$, $\langle ca\rangle$, $\langle cb\rangle$, $\langle cc\rangle$
$\langle d\rangle$	$\langle(cf)\rangle$, $\langle c(bc)(ae)\rangle$, $\langle(\_f)cb\rangle$	$\langle d\rangle$, $\langle db\rangle$, $\langle dc\rangle$, $\langle dcb\rangle$
$\langle e\rangle$	$\langle(\_f)(ab)(df)cb\rangle$, $\langle(af)cbc\rangle$	$\langle e\rangle$, $\langle ea\rangle$, $\langle eab\rangle$, $\langle eac\rangle$, $\langle eacb\rangle$, $\langle eb\rangle$, $\langle ebc\rangle$, $\langle ec\rangle$, $\langle ecb\rangle$, $\langle ef\rangle$, $\langle efb\rangle$, $\langle efc\rangle$, $\langle efcb\rangle$.
$\langle f\rangle$	$\langle(ab)(df)cb\rangle$, $\langle cbc\rangle$	$\langle f\rangle$, $\langle fb\rangle$, $\langle fbc\rangle$, $\langle fc\rangle$, $\langle fcb\rangle$

PrefixSpan Algorithm: Exercise

1. Given the following sequences database

Sequences Database

Customer ID	Data Sequences
1	<{30},{90}>
2	<{10,20},{30},{10,40,60,70}>
3	<{30,50,70,80}>
4	<{30},{30,40,70,80},{90}>
5	<{90}>

find all the frequent $\langle 30 \rangle$ prefix sequences with *minsup* $= 25\%$.

PrefixSpan Efficiency

- No candidate sequence needs to be generated
- Projected databases keep shrinking
- Major cost of PrefixSpan: constructing projected databases with recursively similar suffixes.
  - if it fits on memory, keeping pointers to the suffix offset of the sequence, avoids physical copy.
  - can also be improved by bi-level projections

Summary

- Sequential Pattern Mining is useful in many application, e.g. weblog analysis, financial market prediction, BioInformatics, etc.
- It is similar to the frequent itemsets mining, but with consideration of ordering.
- We have looked at different approaches that are descendants from two popular algorithms in mining frequent itemsets
  - Candidates Generation: AprioriAll and GSP
  - Pattern Growth: FreeSpan and PrefixSpan

Advanced Topics

Advanced Topics: Sequence Mining Constraints

- Item constraint
  - Find web log patterns only about online-bookstores
- Length constraint
  - Find patterns having at least 20 items
- Super pattern constraint
  - Find super patterns of “PC,digital camera”
- Aggregate constraint
  - Find patterns that the average price of items is over $100

Advanced Topics: Sequence Mining Constraints (cont.)

- Regular expression constraint
  - Find patterns “starting from Yahoo homepage, search for hotels in Washington DC area”
  - Yahootravel(WashingtonDC|DC)(hotel|motel|lodging)
- Duration constraint
  - Find patterns about ±24 hours of a shooting
- Gap constraint
  - Find purchasing patterns such that “the gap between each consecutive purchases is less than 1 month”

Advanced Topics: Generating Rules

- In classic sequential pattern mining, no rules are generated.
- It is, however possible to define and generate many type of rules
  - Sequential rules $X \rightarrow Y$, where Y is a sequence and X is a proper subsequence of Y
  - Class sequential rules $X \rightarrow y$, where X is a sequence and y is a class

References

References

Aggarwal, C. C. (2015).
Data Mining, The Texbook.
Springer.

Gama, J. (2016).
Sequence mining.
Slides.

Gama, J., Oliveira, M., Lorena, A. C., Faceli, K., and de Leon Carvalho, A. P. (2015).
Extração de Conhecimento de Dados - Data Mining.
Edições Sílabo, 2nd edition.

Han, J., Kamber, M., and Pei, J. (2011).
Data Mining: Concepts and Techniques.
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition.

Jorge, A. (2016).
Sequential patterns.
Slides.

Liu, B. (2011).
Web Data Mining. Exploring Hyperlinks, Contents, and Usage Data.
Springer, 2nd edition.

References (cont.)

Pei, J., Han, J., Mortazavi-asl, B., Pinto, H., Chen, Q., Dayal, U., and chun Hsu, M. (2001).
Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth.
In *17th International Conference on Data Engineering (ICDE'01)*, pages 215–224.

Srikant, R. and Agrawal, R. (1996).
***Mining sequential patterns: Generalizations and performance improvements*, pages 1–17.**
Springer Berlin Heidelberg.

Tan, P.-N., Steinbach, M., and Kumar, V. (2005).
***Introduction to Data Mining*.**
Addison Wesley.

Torgo, L. (2017).
***Data Mining with R: Learning with Case Studies*.**
Chapman and Hall/CRC, 2nd edition.

Yan, X., Han, J., and Afshar, R. (2003).
Clospan: Mining closed sequential patterns in large datasets.
In *In SDM*, pages 166–177.

Zaki, M. J. (2001).
SPADE: An efficient algorithm for mining frequent sequences.
Machine Learning Journal, 42(1/2):31–60.