Chapter 7

Association Analysis (Part 2)

Xike Xie

The slides are based on Prof. Ben Kao's work

Apriori & FP-growth

A database has 5 transactions. Let $min\_sup = 60\%$ and $min\_conf = 80\%$.

TID	*items_bought*
T100	{M, O, N, K, E, Y}
T200	{D, O, N, K, E, Y }
T300	{M, A, K, E}
T400	{M, U, C, K, Y}
T500	{C, O, O, K, I ,E}

(a) Find all frequent itemsets using Apriori and FP-growth, respectively. Compare the efficiency of the two mining processes.

(b) List all of the *strong* association rules (with support s and confidence c) matching the following metarule, where X is a variable representing customers, and $item_i$ denotes variables representing items (e.g., "*A*", "*B*", etc.):

$$\forall x \in transaction,\ buys(X, item_1) \wedge buys(X, item_2) \Rightarrow buys(X, item_3) \quad [s, c]$$

Solution

(a) Find all frequent itemsets using Apriori and FP-growth, respectively. Compare the efficiency of the two mining processes.

i. For Apriori, one finds the following frequent itemsets, and candidate itemsets (after deletion as a result of `has_infrequent_subset`):

$L_1 = \{E, K, M, O, Y\}$

$C_2 = \{EK, EM, EO, EY, KM, KO, KY, MO, MY, OY\}$
$L_2 = \{EK, EO, KM, KO, KY\}$

$C_3 = \{EKO\}$
$L_3 = \{EKO\}$

$C_4 = \emptyset$
$L_4 = \emptyset$

Finally resulting in the complete set of frequent itemsets:
$\{E, K, M, O, Y, EK, EO, KM, KO, KY, EKO\}$

Overview

- Mining quantitative association rules
- Mining sequential patterns

Quantitative Association Rules (QAR)

- In binary association mining, attributes are assumed to contain binary values (0/1).
- In practice, many datasets contain *quantitative* attributes.
- Example: a dataset of customer records may record information like “age”, “income”.
- These attributes are quantitative (i.e., they take on numerical values).
- Quantitative association rules deal with, not only binary attributes, but also quantitative attributes.

Examples

age:40..60
married:Y
income:500K-1M → properties:1..1
cars:1..2

age:25..30
married:Y
cars:1..2 → children:1..2
insurance:Y

example dataset D

Record ID	Age	Income (K)	married	children
1	20	250	N	0
2	22	300	N	0
3	24	300	Y	0
4	26	290	Y	0
5	28	400	N	0
6	30	350	Y	1
7	32	700	N	0
8	34	750	Y	1
9	36	1,000	Y	2
10	38	1,000	N	0
11	20	300	N	0
12	22	280	Y	1
13	24	290	N	0
14	26	340	N	0
15	28	800	Y	2
16	30	350	Y	1
17	32	370	Y	1
18	34	350	Y	0
19	36	450	Y	1
20	38	750	Y	2
21	39	900	Y	2
22	36	600	N	0
23	35	900	N	0
24	37	900	Y	2
25	34	650	Y	2

Problem Definition

- based on Srikant and Agrawal
- Given a dataset D, define the set I_R which contains 2 kinds of elements:
  - <x: $l..u$>, where x is a quantitative attribute; l and u define a range of x.
  - <x: v>, where x is a binary attribute; v is either 0 or 1 (or "yes/no")
- E.g., <age: 20..23>, <income: 500..620>, <married: 1>, <children: 1..1> are example elements of I_R.

Problem Definition

- A QAR is an implication rule of the form:
  $X \Rightarrow Y$,
  where X and Y are subsets of I_R and that no attribute appears in both X and Y.
- Example: which one of the following is a QAR?

<age:20..23>
<income:500..620>
<married:1>
$\Rightarrow$
<children:1..1>

<age:20..23>
<income:500..620>
$\Rightarrow$
<income:500..650>

Support and Confidence

- Given a set $X \subseteq I_R$, a record r in the dataset supports X if r contains values that fall into the corresponding ranges of the attributes mentioned in X.
- E.g., if X = {<age: 30..35>, <income: 500..800>, <married: 1>},
  then only records 8 and 25 in D support X.
- E.g., Record 18 does not support X because the value of "income" is 350, which is outside the range of 500..800.

Support and Confidence

- Similar to binary association rules, our goal is to find all rules
  $X \Rightarrow Y$
  such that the support and confidence conditions are satisfied.
- That is:
  - $\sup(X \cup Y) \,/\, N \geq \rho_s$
  - $\sup(X \cup Y) \,/\, \sup(X) \geq \rho_c$

QAR

- Mining QAR is a lot more expensive than mining binary association rules because the *number of items* considered under QAR is much larger.
- Each (discrete) numerical attribute with a domain of t values derives $O(t^2)$ items.

Mapping QAR to the binary model

- Our approach of finding QARs is to
  - map the dataset with quantitative attributes to one with binary attributes, then
  - apply Apriori on the binary dataset.
  - We discretize each quantitative attribute into intervals, each then derives a binary attribute.

Transforming Attributes

- Each quantitative attribute is transformed into *n* binary attributes, where *n* is the number of intervals created.
- Example:

…	age	…
…	28	…

↓

…	<age:20..24>	<age:25..29>	<age:30..34>	<age:35..39>	…
…	0	1	0	0	…

record id	Age:20..24	Age:25..29	Age:30..34	Age:35..39	Inc:250..499	Inc:500-749	Inc:750..1000	married	children:0..0	children:1..1	children:2..2
1	1	0	0	0	1	0	0	0	1	0	0
2	1	0	0	0	1	0	0	0	1	0	0
3	1	0	0	0	1	0	0	1	1	0	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0	0	1	0	0
6	0	0	1	0	1	0	0	1	0	1	0
7	0	0	1	0	0	1	0	0	1	0	0
8	0	0	1	0	0	0	1	1	0	1	0
9	0	0	0	1	0	0	1	1	0	0	1
10	0	0	0	1	0	0	1	0	1	0	0
11	1	0	0	0	1	0	0	0	1	0	0
12	1	0	0	0	1	0	0	1	0	1	0
13	1	0	0	0	1	0	0	0	1	0	0
14	0	1	0	0	1	0	0	0	1	0	0
15	0	1	0	0	0	0	1	1	0	0	1
16	0	0	1	0	1	0	0	1	0	1	0
17	0	0	1	0	1	0	0	1	0	1	0
18	0	0	1	0	1	0	0	1	1	0	0
19	0	0	0	1	1	0	0	1	0	1	0
20	0	0	0	1	0	0	1	1	0	0	1
21	0	0	0	1	0	0	1	1	0	0	1
22	0	0	0	1	0	1	0	0	1	0	0
23	0	0	0	1	0	0	1	0	1	0	0
24	0	0	0	1	0	0	1	1	0	0	1
25	0	0	1	0	0	1	0	1	0	0	1

Finding frequent itemsets

- After the mapping, each attribute-interval is considered a binary item. Apriori is then applied on the binary dataset to find all the frequent itemsets.

Frequent itemsets

- Example: if we set ρ_s = 20%, then the frequent itemsets found in *D* are:

Itemset	Support
<age:20..24>	6
<age:30..34>	7
<age:35..40>	8
<income:250..499>	14
<income:750..1000>	8
<married:1>	15
<children:0..0>	13
<children:1..1>	6
<children:2..2>	6

Frequent itemsets

Itemset	Support
<age:20..24>, <income:250..499>	6
<age:20..24>, <children:0..0>	5
<age:30..34>, <married:1>	6
<age:35..39>, <income:750..1000>	6
<age:35..39>, <married:1>	5
<income:250..499>, <married:1>	8
<income:250..499>, <children:0..0>	9
<income:250..499>, <children:1..1>	5
<income:750..1000>, <children:2..2>	5
<married:1>, <children:1..1>	6
<married:1>, <children:2..2>	6
<age:20..24>, <income:250..499>, <children:0..0>	5
<income:250..499>, <married:1>, <children:1..1>	5

Rules

- Example: if we set $\rho_c = 70\%$, then 15 rules are found in D.

<age:20..24> → <income:250..499>

<age:20..24> → <children:0..0>

<age:30..34> → <married:1>

<age35..39> → <income:750..1000>

<income:750..1000> → <age35..39>

<children:1..1> → <income:250..499>

<children:2..2> → <income:750..1000>

<children:1..1> → <married:1>

<children:2..2> → <married:1>

Rules

<age:20..24>	→	<income:250..499> <children:0..0>
<age:20..24> <children:0..0>	→	<income:250..499>
<age:20..24> <income:250..499>	→	<children:0..0>
<married:1> <children:1..1>	→	<income:250..499>

<children:1..1>	→	<married:1> <income:250..499>
<income:250..499> <children:1..1>	→	<married:1>

Problems with the mapping approach

- *the partitioning problem*
  - the rules generated depend heavily on how the quantitative attributes are partitioned.
- *the fragmented rules problem*
  - some of the rules generated can be combined to form more concise rules.

The partitioning problem

- The partitioning of a quantitative attribute into intervals cannot be too fine or too coarse.
- Example: if "age" is partitioned into 20 intervals, each containing a single value (i.e., [20..20], [21..21], …, etc.) then no itemset (in our toy example *D*) concerning "age" is frequent. Hence, no rules concerning "age" can be derived.

The partitioning problem

- Example: if "income" is partitioned into only 1 interval (i.e., <income:250..1000>), then many "obvious" rules will be generated.
- For example:

<age:20..24> → <income:250..1000>

The partitioning problem

- To avoid partitioning the intervals too fine, one method is to specify a maximum support (maxsup) parameter. Intervals are allowed to merge with neighboring intervals as long as the resulting support does not exceed maxsup.
- Example: suppose "age" is partitioned into intervals of size 2, the support counts of the intervals are:

item	{<age:20..21>}	{<age:22..23>}	{<age:24..25>}	{<age:26..27>}	{<age:28..29>}
Support count	2	2	2	2	2
item	{<age:30..31>}	{<age:32..33>}	{<age:34..35>}	{<age:36..37>}	{<age:38..39>}
Support count	2	2	4	4	3

suppose maxsup is set to 10

item	{<age:20..23>}	{<age:24..27>}	{<age:28..31>}	{<age:32..35>}	{<age:36..39>}
Support count	4	4	4	6	7

item	{<age:20..27>}	{<age:28..35>}	{<age:36..39>}
Support count	8	10	7

Fragmented rules

- Consider the following rules that were generated in our example:

<children:1..1> → <married:1>

<children:2..2> → <married:1>

- These rules could be integrated into:

<children:1..2> → <married:1>

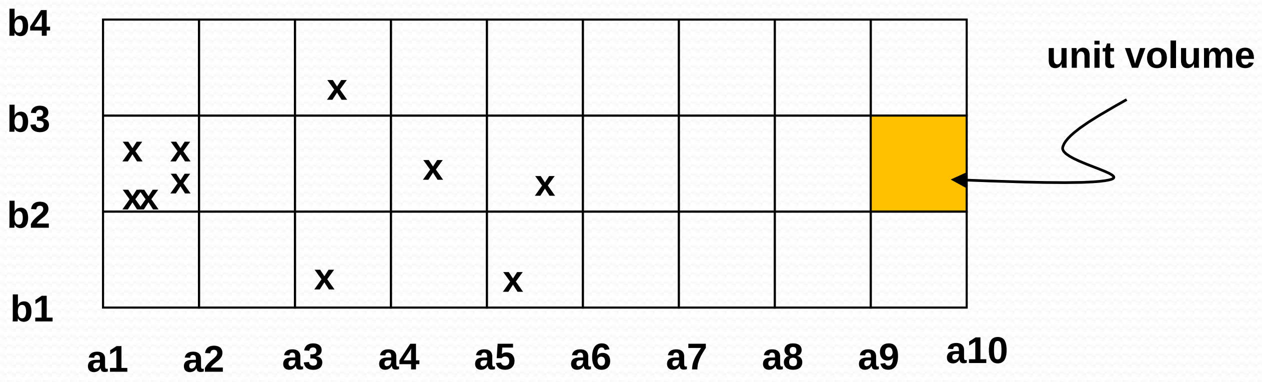
Note that the integrated rule would never be generated in our example because the item <children:1..2> is not generated during the partition process.

Fragmented Rules

- Rule merging needs to be done to handle the fragmented rule problem.
- Rules that share the same right hand side and having the same set of attributes on the left hand side, should be consider for possible merging.

Dense-Region-Based Approach

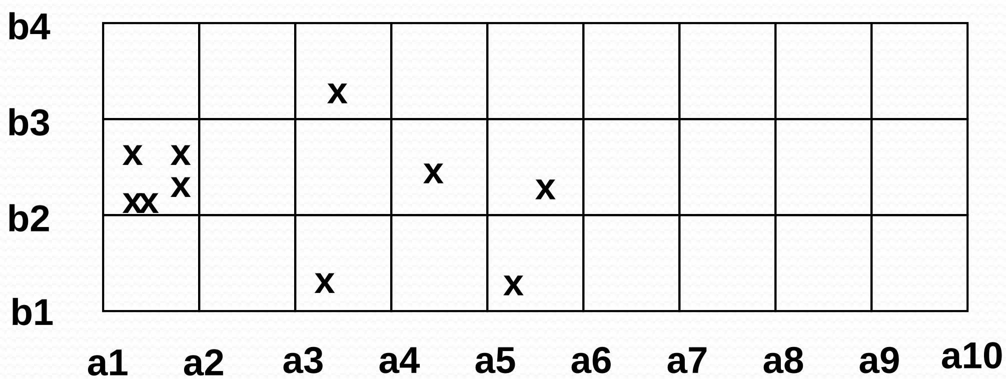
- Consider the following dataset (with only 2 quantitative attributes A and B).



Dense-Region-Based Approach

- Consider the association rules:
  - <A:a1..a2> ⇒ <B:b2..b3>
  - <A:a3..a6> ⇒ <B:b1..b4>

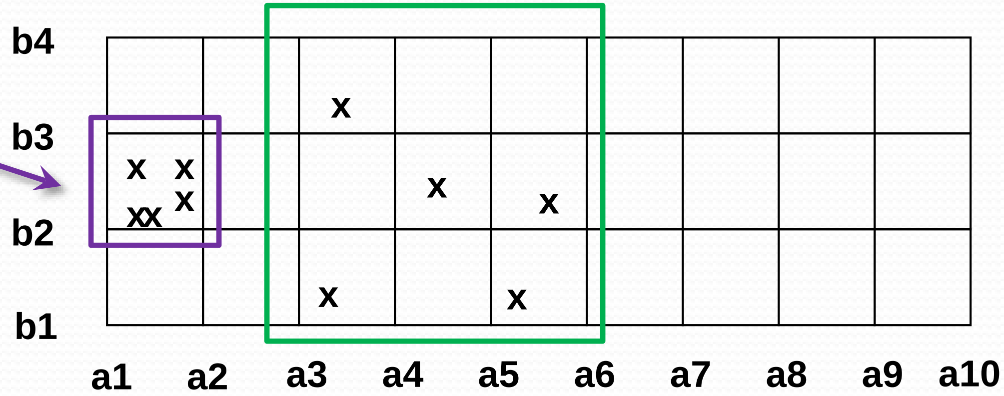
If $\rho_s = \rho_c$ = 50%, both rules are valid rules.



Dense-Region-Based Approach

- Consider the association rules:
  - <A:a1..a2> ⇒ <B:b2..b3>
  - <A:a3..a6> ⇒ <B:b1..b4>

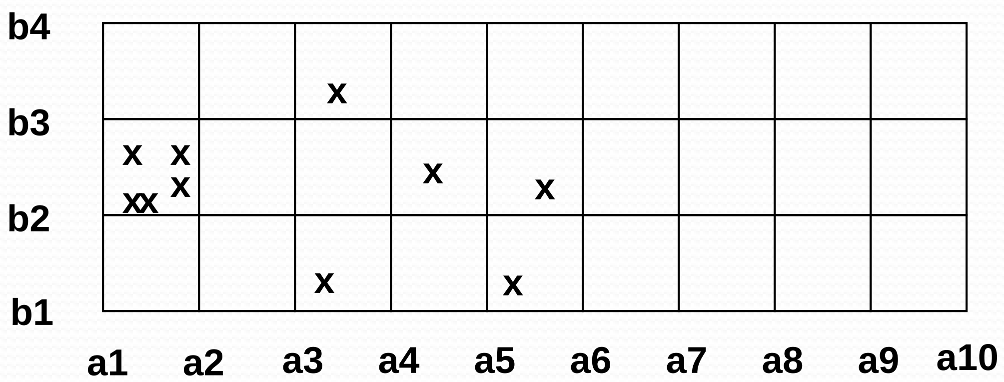
Which rule is more interesting?



Dense-Region-Based Approach

- Consider the association rules:
  - <A:a1..a2> ⇒ <B:b2..b3>
  - <A:a3..a6> ⇒ <B:b1..b4>

Which rule is more interesting?



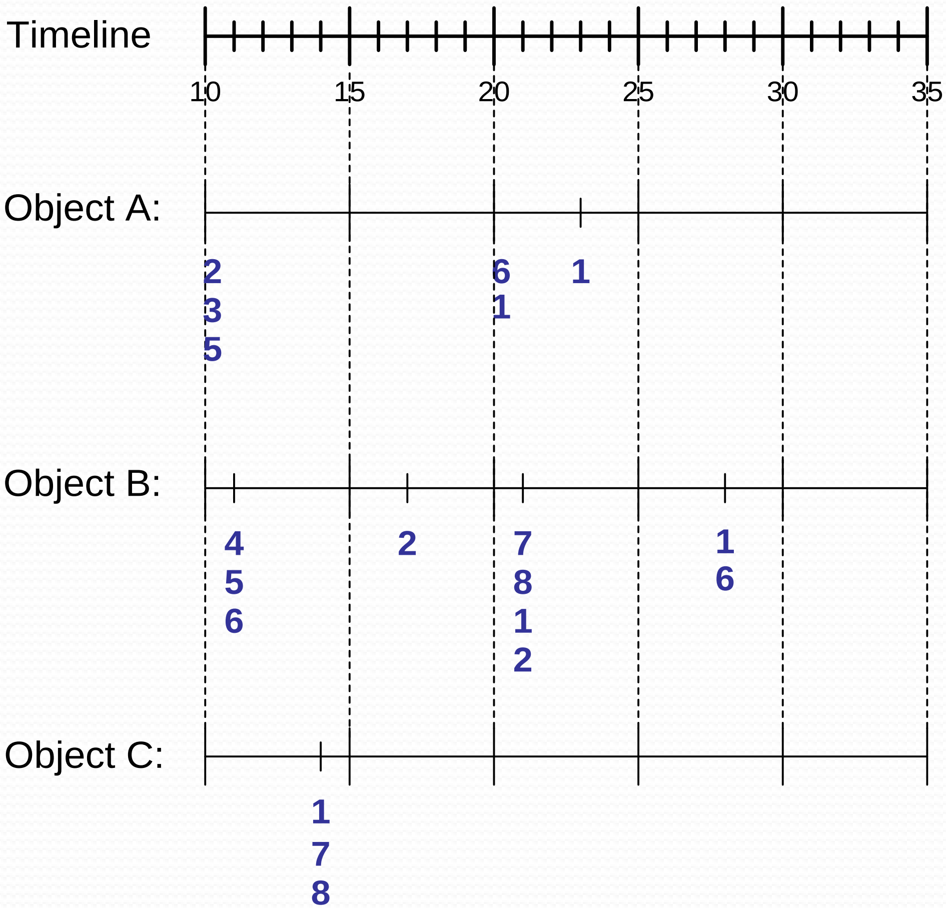
Dense-Region-Based Approach

- Each rectangular region defines a rule
- The denser the region, the more specific and useful is the corresponding rule
- A region is *dense* if its density satisfies a density threshold ρ_d.
- The dense-region-based approach to QAR mining is to discover dense regions and derive rules from those regions.
- Can you think of a scenario in which the dense-region-based approach is ineffective?

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

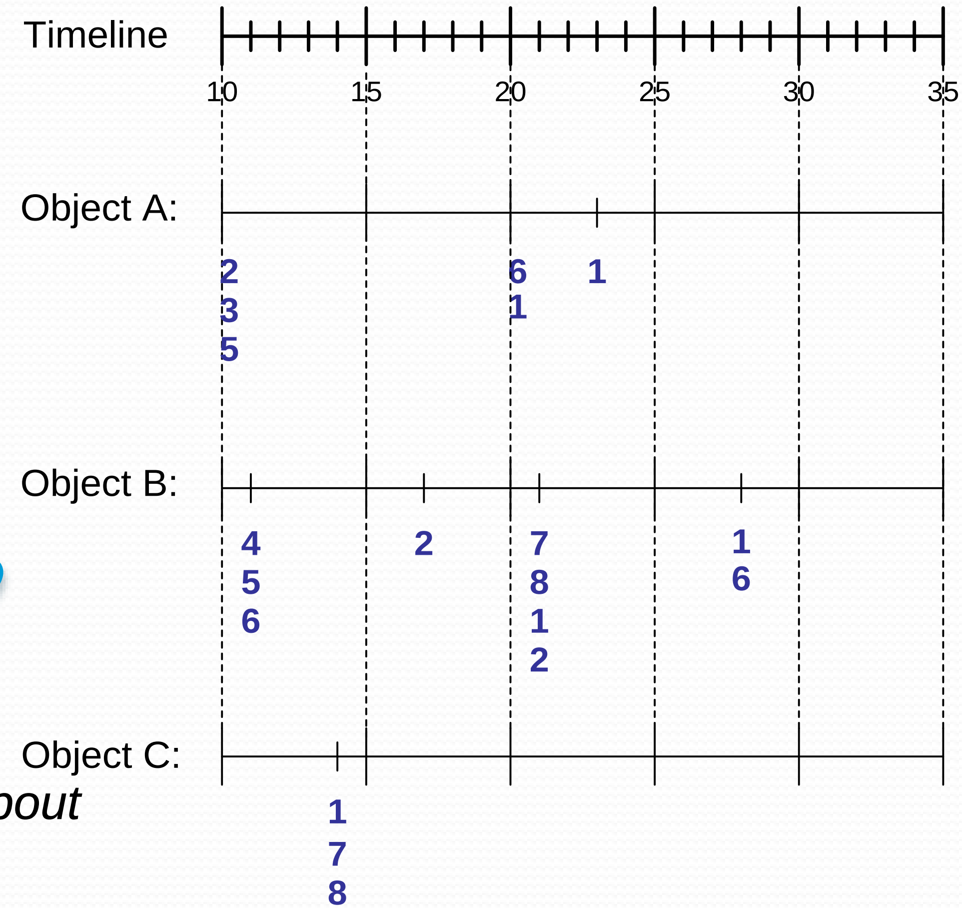


Ex: Objects are customers who buy sets of products (events) at different times

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

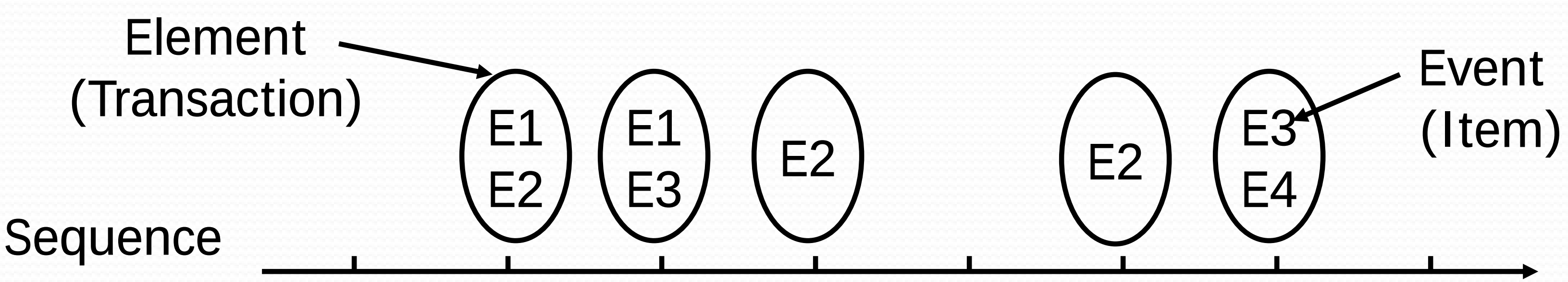


What binary association rule mining cares about

Ex: Objects are customers who buy sets of products (events) at different times

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 web page viewed	Items or contents displayed in the page
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 associated with a specific time

  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)
- Ex. of 3-sequences: <{a,b}, {a}>, <{a,b,c}>, <{a},{b},{b}>

Examples of Sequence

- Web sequence:

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

- Sequence of books checked out at a library:

  <{Fellowship of the Ring} {The Two Towers} {Return of the King}>

Definition

- 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 $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, …, $a_n \subseteq b_{in}$
- 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 $\geq \rho_s$)

Definition

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

Definition

- Given:
  - a database of sequences
  - a user-specified minimum support threshold, ρ_s
- Task:
  - Find all subsequences with support $\geq \rho_s$

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.

Challenge

- 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

ρ_s = 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%

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

ρ_s = 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%

Supported by A, B, C, but not D, E

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:

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

  $<\{i_1, i_2, i_3\}>, <\{i_1, i_2, i_4\}>, \ldots, <\{i_1, i_2\}\ \{i_1\}>, <\{i_1, i_2\}\ \{i_2\}>, \ldots,$
  $<\{i_1\}\ \{i_1, i_2\}>, <\{i_1\}\ \{i_1, i_3\}>, \ldots, <\{i_1\}\ \{i_1\}\ \{i_1\}>, <\{i_1\}\ \{i_1\}\ \{i_2\}>, \ldots$

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)st 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 ρ_s

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}>

Candidate Generation

- General case (k>2):
  - Merge two frequent (k-1)-sequence w_1 and 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
  < {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} >

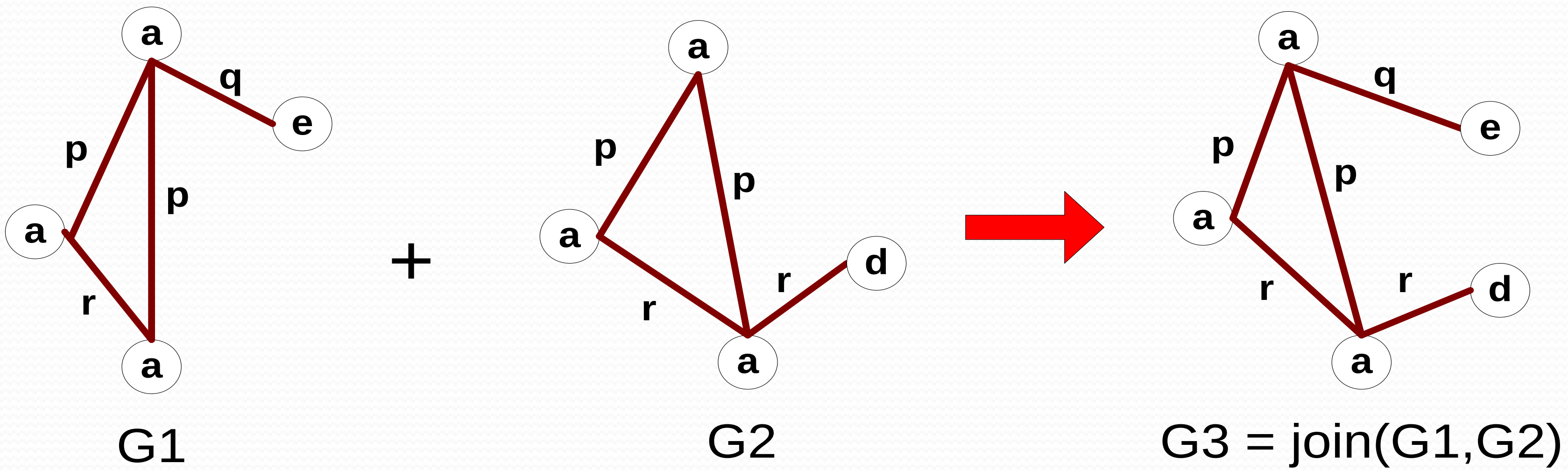
Candidate
Pruning

< {1} {2 5} {3} >

Frequent Subgraph Mining

- Extend association rule mining to finding frequent subgraphs
- Frequent subgraphs can be considered as common features of a graph's structure.
- Useful for graph indexing

Vertex Growing



Summary

- Association rule mining is an important data mining task
- Frequent itemset mining is the main component of association rules mining
  - association rules can be generated from frequent patterns
- The basic Apriori algorithm and some of its optimized versions
- Lift as an interestingness measure of rules
- Quantitative association rules mining, frequent sequences mining, frequent subgraph mining can all be done with the Apriori property.