Chapter 6

Association Analysis:
Advanced Concepts

Association Analysis: Advanced Concepts

- The association rule mining formulation in Ch.5 assumes that
  - The input data consists of ***binary attributes*** called items
  - The presence of an item in a transaction is also assumed to be more important than its absence 나타나는게 중요하다고 포커스를 둠
  - As a result, an item is treated as an ***asymmetric binary attribute*** and only frequent patterns are considered interesting ↳ 0,1 인데 1에 집중함!

- This chapter extends the formulation to data sets with
  - Categorical attributes
  - Continuous attributes
  - A concept of hierarchy 계층구조
  - Sequences

Handling Categorical Attributes

Handling Categorical Attributes

- There are many data that contain ***categorical*** attributes
  - (ex) Internet survey data with categorical attributes

Gender	Level of Education	State	Computer at Home	Chat Online	Shop Online	Privacy Concerns
Female	Graduate	Illinois	Yes	Yes	Yes	Yes
Male	College	California	No	No	No	No
Male	Graduate	Michigan	Yes	Yes	Yes	Yes
Female	College	Virginia	No	No	Yes	Yes
Female	Graduate	California	Yes	No	No	Yes
...	...	...	...	...	...	...

- Using association analysis, we may uncover ***interesting rules***
  - (ex) {Shop Online = Yes} → {Privacy Concerns = Yes}

Transformation of Categorical Attributes

- To extract such patterns, we must ***transform*** the categorical attributes into "items" first
  - So that existing association rule mining algorithms can be applied

    개수는 매우 많아지겠지만 classification이 아니므로 차원의 저주 X

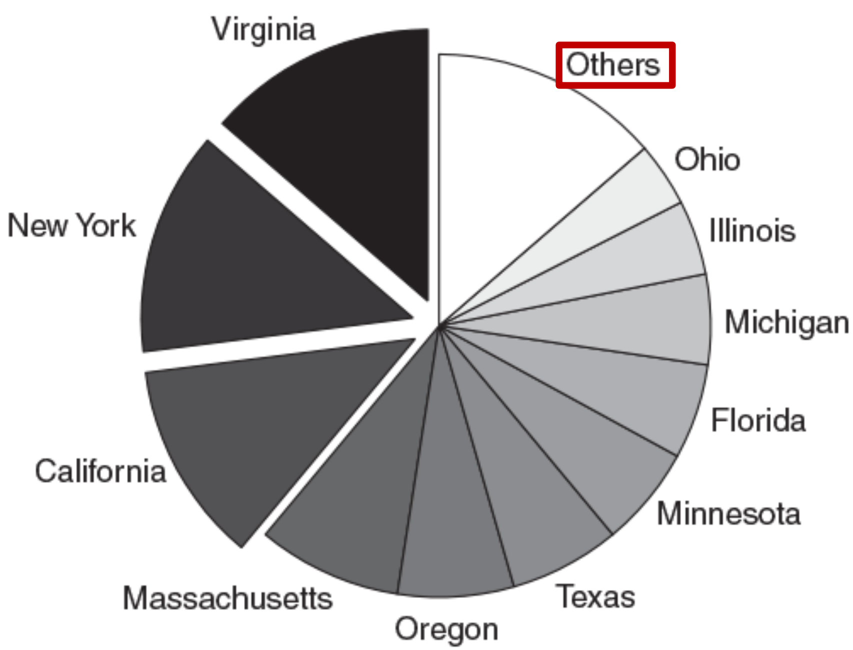
- We create a new ***item*** for each distinct ***attribute-value pair***
  - (ex) Internet survey data after transformation

Male	Female	'Education = Graduate' 이것 자체가 item	Education = College	...	Privacy = Yes	Privacy = No
0	1	1	0	...	1	0
1	0	0	1	...	0	1
1	0	1	0	...	1	0
0	1	0	1	...	1	0
0	1	1	0	...	1	0
...	...	...	...	...	...	...

Several Issues to Consider (1/3)

1. Some attribute values may ***not*** be frequent
   - They are not enough to be part of a frequent pattern
   - **Solution 1**: group related attribute values into a small number of categories

     구역으로 묶기 (minsup을 넘기게 하려고)
     - Many state names → Midwest, Pacific Northwest, Southwest, East Cost
   - **Solution 2**: aggregate the less frequent values into a single category
     - Many small state names → Others



Several Issues to Consider (2/3)

2. Some attribute values may be ***too*** frequent ex) 성심당의 튀김소보로

   – For example, if 85% of the survey participants own a home computer, we may potentially generate many redundant patterns (압도적인것제거)

{**Computer at Home = Yes**, Shop Online = Yes} → {Privacy Concerns = Yes}
{…, **Computer at Home = Yes**, …} → {…} 의미없는 rule만 많이 만듦

   – Because the high-frequency items correspond to the typical values of an attribute, they ***seldom*** carry any new information

   – **Solution**: remove such items before applying association analysis

     • It may be more useful to better understand the pattern

Several Issues to Consider (3/3)

candidate를 만들때

3. There can be many ***meaningless*** candidate itemsets
   - For example, we do not have to generate a candidate itemset such as $\{State = X, State = Y, \ldots\}$ because its support count is 0
   - **Solution**: Avoid generating candidate itemsets that contain more than one item from the ***same*** attribute

$\{\ldots, Education = Graduate, Education = College, \ldots\} \rightarrow \{\ldots\}$ **(X)**

$\{\ldots, Privacy = Yes, Privacy = No, \ldots\} \rightarrow \{\ldots\}$ **(X)**

↳ 동시에 두개 불가능!
버린다

$X_1 \rightarrow V_1$

(a, b, c)
(a, b, d) → (a, b, c, d)

$X_1 \rightarrow V_2$

$X_1 \rightarrow V_1$ 또는 $X_1 \rightarrow V_2$
하나만 선택해야하는 categorical 한 X_1이므로
어차피 support = 0

⇒ 같은 attribute에서 온 것인지 확인하고
의미없는 것은 바로 버린다

Handling Continuous Attributes

Handling Continuous Attributes

- There are many data that contain ***continuous*** attributes
  - (ex) Internet survey data with continuous attributes

Gender	...	Age	Annual Income	No. of Hours Spent Online per Week	No. of Email Accounts	Privacy Concern
Female	...	26	90K	20	4	Yes
Male	...	51	135K	10	2	No
Male	...	29	80K	10	3	Yes
Female	...	45	120K	15	3	Yes
Female	...	31	95K	20	5	Yes
...	...	...	...	...	...	...

value가 너무 많아서
특정값으로 item을 만들기 불가능하거나 어려움

- Mining continuous attributes may reveal ***useful insights***
  - (ex) {Annual Income > 120K)} → {Age ∈ [45, 60)}
  - (ex) {Email Accounts > 3, Hours Spent Online > 15} → {Privacy = Yes}

딱히 방법이 없어서
그냥 적당히 categoric하게
바꿈
(구역을 value로)

Discretization-Based Methods

나머지 방법은 너무 느림

- Group the adjacent values of a continuous attribute into a finite number of intervals → 인접한 것을 group으로 묶음
  - (ex) Age → Age ∈ [12, 16), Age ∈ [16, 20), …, Age ∈ [56, 60)

- We can use any of the discretization techniques described before
  - Equal interval width, equal frequency, or clustering
  - (ex) Internet survey data after transformation

그럼 구역을 어떻게 나누지?!

* 구역을 촘촘하게
  confidence↑
  support↓

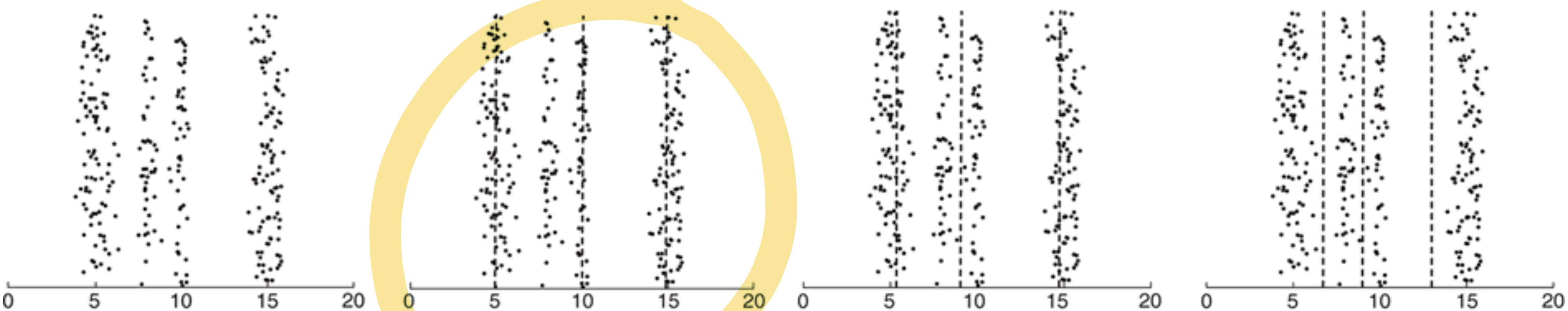
* 구역을 듬성듬성하게
  confidence↓
  support↑

Male	Female	...	'Age < 13' item	'Age ∈ [13, 21)' item	Age ∈ [21, 30)	...	Privacy = Yes	Privacy = No
0	1	...	0	0	1	...	1	0
1	0	...	0	0	0	...	0	1
1	0	...	0	0	1	...	1	0
0	1	...	0	0	0	...	1	0
0	1	...	0	0	0	...	1	0
...	...	...	...	...	...	...	...	...

The Number of Intervals → 구역의 개수는 구역의 폭과 연결됨

- A ***key*** parameter in attribute discretization

- This parameter is typically provided by the users
  - Equal interval width approach → the interval width
  - Equal frequency approach → the number of transactions per interval
  - Clustering-based approach → the number of desired clusters



(a) Original data. (b) Equal width discretization. (c) Equal frequency discretization. (d) K-means discretization.

Difficulty in Determining the Interval Width

<구역을 잘게 자른 예시>

Age Group	Chat Online = Yes	Chat Online = No
[12, 16)	12	13
[16, 20)	11	2
[20, 24)	11	3
[24, 28)	12	13
[28, 32)	14	12
[32, 36)	15	12
[36, 40)	16	14
[40, 44)	16	14
[44, 48)	4	10
[48, 52)	5	11
[52, 56)	5	10
[56, 60)	4	11

계산해보기
support : 전체 中
confidence :

- There are ***two*** strong rules (assume $minsup = 5\%$, $minconf = 65\%$)
  - R_1: Age ∈ [16, 24) → Chat Online = Yes (s = 8.8%, c = 81.5%)
  - R_2: Age ∈ [44, 60) → Chat Online = No (s = 16.8%, c = 70.5%)

강력하게 rule이 보임!

Issues with the Interval Width (1/3)

1. If the interval is too ***wide***
   – We may lose some patterns because of their lack of ***confidence***

     → 아무 rule을 찾을수 없게됨

- Example
  – If the interval width is 24 years, R_1 and R_2 are replaced by follows rules:

    R_1': Age ∈ [12, 36) → Chat Online = Yes (s = 30%, **c = 57.7%**)

    R_2': Age ∈ [36, 60) → Chat Online = No (s = 28%, **c = 58.3%**)

  – Despite their higher supports, their confidences drop below *minconf*
  – As a result, both patterns are lost after discretization

Issues with the Interval Width (2/3)

2. If the interval is too ***narrow***
   - We may lose some patterns because of their lack of ***support***

     → minsup를 채우지 못함

- Example
  - If the interval width is 4 years, R_1 is broken up into the following two rules:

R_{11}: Age $\in$ [16, 20) → Chat Online = Yes (**s = 4.4%**, c = 84.6%)

R_{12}: Age $\in$ [20, 24) → Chat Online = Yes (**s = 4.4%**, c = 78.6%)

  - Since their supports are less than *minsup*, R_1 is lost after discretization
  - Similarly, R_2, which is broken up into four subrules, will also be lost because the support of each subrule is less than *minsup*

Issues with the Interval Width (3/3)

중간!

아직 범위는 최적으로 잡는 방법이 없다!

3. If the interval width is 8 years

– R_2 is broken into the following two subrules:

건졌다!

R_{21}: Age $\in$ [44, 52) → Chat Online = No (s = 8.4%, c = 70%)
R_{22}: Age $\in$ [52, 60) → Chat Online = No (s = 8.4%, c = 70%)

– R_{21} and R_{22} have sufficient support and confidence
  - Thus, we can recover R_2 by aggregating the two subrules

ㅠ.ㅠ

– Meanwhile, R_1 is broken into the following two subrules:

support (o)
confidence (x)
↳ 애매한 애들과 grouping 되면서 강력한 특징이 희미해짐

R_{11}: Age $\in$ [12, 20) → Chat Online = Yes (s = 9.2%, **c = 60.5%**)
R_{12}: Age $\in$ [20, 28) → Chat Online = Yes (s = 9.2%, **c = 60.0%**)

– R_{11} and R_{12} fail the confidence threshold
  - Thus, we cannot recover R_1 by aggregating the two subrules

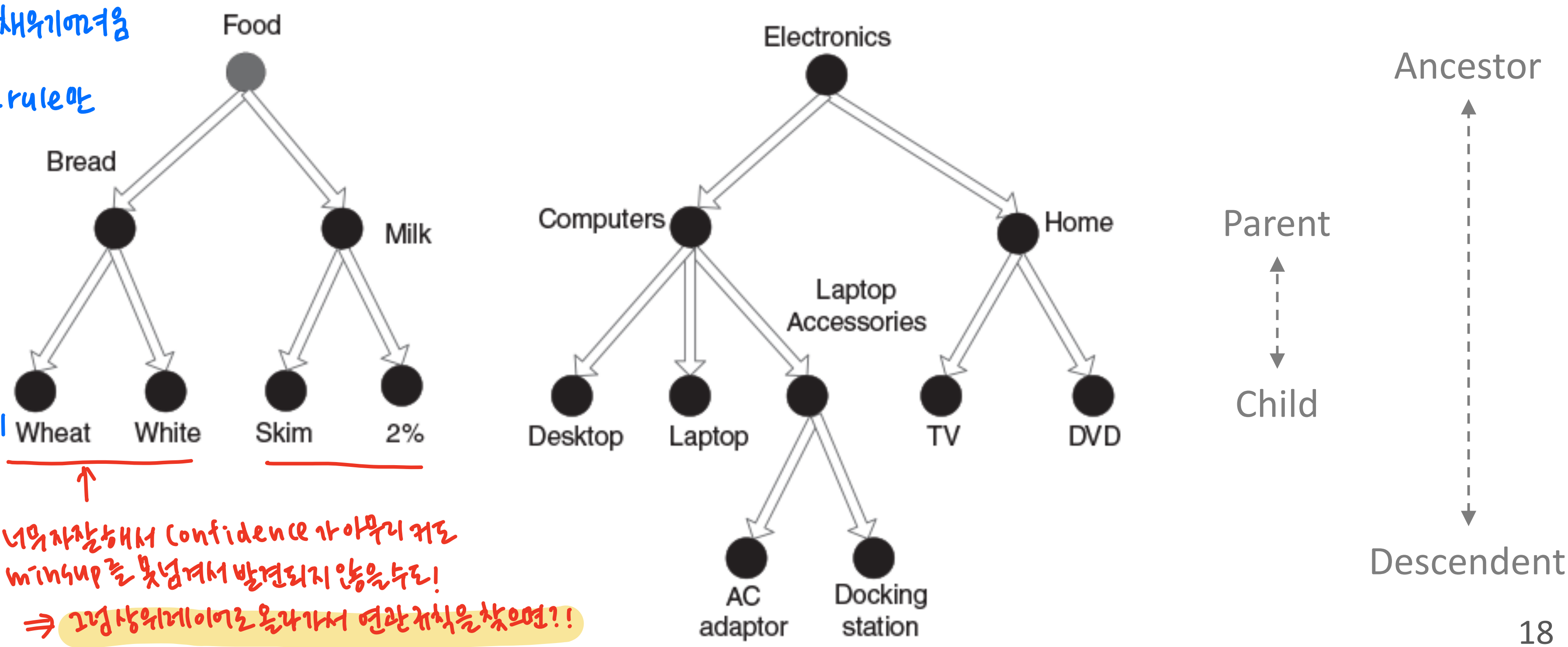
Handling a Concept Hierarchy

Handling a Concept Hierarchy

- A concept hierarchy
  - A multilevel organization of the various entities or concepts defined in a particular domain
  - (ex) an item taxonomy in market basket analysis
    - e.g., milk is a kind of food, DVD is a kind of home electronics

confidence를 채우기 어려움
+
너무 일반적인 rule만 찾게 됨

support 채우기 어려움



너무 자잘해서 confidence가 아무리 커도 minsup을 못넘겨서 발견되지 않을수도!
⇒ 그럼 상위레이어로 올라가서 연관규칙을 찾으면?!

Advantages of Using Concept Hierarchies

1. Items at the lower levels of a hierarchy may ***not*** have enough support to appear in any frequent itemset

→ minsup이 너무 작다!

- Examples
  - Although the sale of {AC adaptors} and {docking stations} may be low, the sale of {laptop accessories} may be high
  - Also, rules involving high-level categories may have lower confidence than ones generated using low-level categories

✓ Unless the concept hierarchy is used, there is a potential to miss interesting patterns at ***different levels*** of categories

Advantages of Using Concept Hierarchies

너무디테일

2. Rules found at the lower levels tend to be ***overly specific*** and may not be as interesting as rules at the higher levels

- Examples
  - Staple items such as milk and bread tend to produce many low-level rules
  - (ex) {skim milk} → {wheat bread}, {2% milk} → {wheat bread}, {skim milk} → {white bread}, {2% milk} → {white bread}

✓ Using a concept hierarchy, they can be summarized into a single rule

- (ex) {milk} → {bread}

Advantages of Using Concept Hierarchies

3. Considering only items at the top level of their hierarchies also may ***not*** be good enough → 너무 일반화되거나, confidence를 채우기 어려움
   - Because such rules may not be of any practical use

- Example
  - The rule {electronics} → {food} may satisfy *minsup* and *minconf*
  - However, it is not informative because it ***overgeneralizes*** the situation

✓ If {milk, DVD} are the only items sold together frequently, then we may find {milk} → {DVD} using a concept hierarchy

Extending Standard Association Analysis

알고리즘확장 x, 데이터확장 o

- We can extend standard association analysis to incorporate concept hierarchies in the following way:

좀무식한방법...

- Basic strategy
  - Replace each transaction t with its ***extended transaction*** t'
    - t' contains all the items in t along with their corresponding ***ancestors*** 아이템인것처럼 넣어줌
    - (ex) t = {DVD, wheat} → t' = {DVD, wheat, home, electronics, bread, food}
  - Then apply existing algorithms to the database of extended transactions
    - Such as Apriori and FP-growth

- Such an approach would find rules that span ***different levels*** of the concept hierarchy

Limitations of This Extension (1/2)

1. T.T Items at the higher levels tend to have higher support counts than those at the lower levels
   - As a result, if *minsup* is set too high, then ***only*** patterns involving the high-level items are extracted

     ↳ 밑의것에서는 rule 발견
   - On the other hand, if *minsup* is set too low, then the algorithm generates far ***too many*** patterns and becomes computationally inefficient

     ↳ 너무 자잘한 rule 까지 발견

느려짐! 데이터가 점점 커짐..

2. T.T Introduction of a concept hierarchy tends to ***increase*** the computation time of association analysis algorithms
   - Because of the larger number of items and wider transactions
   - The number of candidate patterns and frequent patterns may also grow exponentially with wider transactions

Limitations of This Extension (2/2)

3. Using a concept hierarchy may produce ***redundant*** rules

선택!

– A rule $X \rightarrow Y$ is redundant if there exists a more general rule $X' \rightarrow Y'$, where X' and Y' are the ***ancestors*** of X and Y, respectively

더 일반적인게 있으면 그걸 채택

– Example

- Suppose we have the rule $\{\text{bread}\} \rightarrow \{\text{milk}\}$
- Then $\{\text{white bread}\} \rightarrow \{\text{2\% milk}\}$, $\{\text{wheat bread}\} \rightarrow \{\text{2\% milk}\}$, $\{\text{white bread}\} \rightarrow \{\text{skim milk}\}$, and $\{\text{wheat bread}\} \rightarrow \{\text{skim milk}\}$ are all redundant
  - Because they can be summarized by $\{\text{bread}\} \rightarrow \{\text{milk}\}$
- An itemset such as $\{\text{skim milk, milk, food}\}$ is also redundant
  - Because food and milk are ancestors of skim milk

– Fortunately, it is easy to eliminate such redundant rules during frequent itemset generation

잘못된 자료?!

- (ex) eliminate a frequent itemset $\{X, Y\}$ if there is a frequent itemset $\{X', Y'\}$

통과되는게 있을수도, 아닐수도 있으므로 쉽게 날릴 수는 없음

Sequential Patterns

Sequential Patterns (1/2)

- Market basket data often contains ***temporal*** information about when an item was purchased by customers

  로그의 timestamp

  - (ex) the ***sequence*** of transactions made by a customer
    - $<\{\text{Wine, Wallet}\}\{\text{Lego}\}\{\text{Gift card, Flower}\}\ldots>$

- Similarly, event-based data have an inherent ***sequential*** nature
  - (ex) data collected from scientific experiments or the monitoring of physical systems, such as telecommunication networks and computer networks, and wireless sensor networks
    - $<\ldots\{\text{Low Pressure}\}\ldots\{\text{Heavy Cloud}\}\ldots\{\text{Rain}\}\ldots>$

- However, association rules so far emphasize only "***co-occurrence***" relationships and disregard the ***sequential*** information of the data
  - (ex) $\{Diapers, Milk\}$ vs. $<\{Diaper\}\{Milk\}>$

    ↑ sequential

Sequential Patterns (2/2)

- The latter information may be valuable for identifying recurring features of a dynamic system or predicting future occurrences of certain events

SID	Sequence
1	$\langle\{a, b\}, \{c\}, \{f, g\}, \{g\}, \{e\}\rangle$
2	$\langle\{a, d\}, \{c\}, \{b\}, \{a, b, e, f\}\rangle$
3	$\langle\{a\}, \{b\}, \{f, g\}, \{e\}\rangle$
4	$\langle\{b\}, \{f, g\}\rangle$

주어야하는 minsup

Sequential patterns

with support ≥ 60%

$<\{b\}\{f,g\}>$,

$<\{a\}\{e\}>$,

…

minimum confidence는 없음!

- This section presents the basic concept of sequential patterns and an algorithm developed to discover them

Preliminaries

- **Input**: a ***sequence*** data set
  - Each row records the occurrences of ***events*** with an object at a given ***time***
  - The ***timestamp*** information enables a different style of association analysis

현실데이터는 대부분 timestamp로 sort돼있음

Object	Timestamp	Events
A	10	2, 3, 5
A	20	1, 6
A	23	1
B	11	4, 5, 6
B	17	2
B	21	1, 2, 7, 8
C	14	1, 6
C	28	1, 7, 8

↑ 로그데이터

preprocessing

꼭 가공해야함!

여기서는 정확히 언제 말고
순서만을 따짐!

A: $\langle\{2,3,5\}\{1,6\}\{1\}\rangle$
B: $\langle\{4,5,6\}\{2\}\{1,2,7,8\}\rangle$
C: $\langle\{1,6\}\{1,7,8\}\rangle$

sequence data set

- **Output**: association patterns of events that commonly occur in a ***sequential order*** across objects
  - (ex) $\langle\{6\}\{1\}\rangle$ (i.e., event 6 is followed by event 1)
  - Such a pattern cannot be inferred using the traditional association analysis

Sequences (1/2)

- An ordered list of ***elements*** (transactions)

우리의 case에서는
각각의 elements = transactions 라고 봐도됨

- A sequence can be denoted as $s = \langle e_1 e_2 e_3 \ldots e_n \rangle$
  - $e_j = \{i_1, i_2, \ldots, i_k\}$: a collection of one or more ***events*** (items)

- Examples of sequences
  - Sequence of web pages viewed by a web site visitor
    - (ex) $\langle$\{Homepage\} \{Electronics\} \{Cameras\} \{Shopping Cart\} …$\rangle$
  - Sequence of events leading to the nuclear accident
    - (ex) $\langle$\{clogged resin\} \{outlet valve closure\} \{loss of feedwater\}, …$\rangle$
  - Sequence of classes taken by a software major student in each semester
    - (ex) $\langle$\{Algorithm, OS\} \{DB, CA\} \{Network, SE\} \{Graphics, Mining\}, …$\rangle$

Sequences (2/2)

- k-sequence
  - A sequence that contains k events (items)
  - Examples
    - 2-sequences: $<\{a,b\}>$, $<\{a\}\ \{b\}>$
    - 3-sequences: $<\{a,b\}\ \{c\}>$, $<\{a\}\ \{b\}\ \{c\}>$, $<\{a\}\ \{b,c\}>$

- 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	The 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

Subsequences

하나씩 매핑

- A sequence $t = <t_1 t_2 \ldots t_m>$ is a ***subsequence*** of $s = <s_1 s_2 \ldots s_n>$
  - If there exist $1 \le j_1 < j_2 < \ldots < j_m \le n$ such that $\boldsymbol{t_1 \subseteq s_{j_1}, t_2 \subseteq s_{j_2}, \ldots, t_m \subseteq s_{j_m}}$

    순서중요!
  - In other words, t can be derived from s by simply deleting some events from elements in s or even deleting some elements in s completely

- If t is a subsequence of s, then we say that t is ***contained*** in s

- Examples

s안에 t가 sub로 들어가있는지?

Sequence s	Sequence t	Is t a subsequence of s?
$\langle\{2,4\}\{3,5,6\}\{8\}\rangle$	$\langle\{2\}\{3,6\}\{8\}\rangle$ 순서도 일치해야함	Yes
$\langle\{2,4\}\{3,5,6\}\{8\}\rangle$	$\langle\{2\}\{8\}\rangle$	Yes
$\langle\{1,2\}\{3,4\}\rangle$ 여기에 2가 없음!	$\langle\{1\}\{2\}\rangle$	No
$\langle\{2,4\}\{2,4\}\{2,5\}\rangle$	$\langle\{2\}\{4\}\rangle$	Yes

minsup 계산을 위해 count

Sequential Pattern Discovery

- Let D be a data set that contains one or more ***data sequences***
  - Data sequence: an ordered list of elements associated with a ***single*** object
  - (ex) the data set shown below contains five data sequences

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

전처리

A: $<\{1,2,4\}\ \{2,3\}\ \{5\}>$
B: $<\{1,2\}\ \{2,3,4\}>$
C: $<\{1,2\}\ \{2,3,4\}\ \{2,4,5\}>$
D: $<\{2\}\ \{3,4\}\ \{4,5\}>$
E: $<\{1,3\}\ \{2,4,5\}>$

5 data sequences

- The support of a sequence s 지정해주기
  - The fraction of all data sequences that ***contain*** s
  - (ex) the support of $<\{1\}\ \{2\}> = 4/5 = 80\%$

Problem Definition

Given a sequence data set D and a user-specified minimum support threshold $minsup$, find ***all*** sequences with support ≥ $minsup$

- Example ($minsup = 50\%$)

Sequence data set D:

A: <{1,2,4} {2,3} {5}>
B: <{1,2} {2,3,4}>
C: <{1,2} {2,3,4} {2,4,5}>
D: <{2} {3,4} {4,5}>
E: <{1,3} {2,4,5}>

★ 몇번 등장하는지?
시험문제~.~

50%(2.5건)
이상 발생하는
subsequence

Examples of Sequential Patterns:

<{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%

Challenges in Sequential Pattern Discovery

- The set of all possible sequences is ***exponentially*** large and difficult to enumerate
  - (ex) a collection of n events can result in the following examples of 1-sequences, 2-sequences, and 3-sequences:

1-sequences:	$\langle i_1 \rangle, \langle i_2 \rangle, \ldots, \langle i_n \rangle$
2-sequences:	$\langle \{i_1, i_2\} \rangle, \langle \{i_1, i_3\} \rangle, \ldots, \langle \{i_{n-1}, i_n\} \rangle, \ldots$ $\langle \{i_1\}\{i_1\} \rangle, \langle \{i_1\}\{i_2\} \rangle, \ldots, \langle \{i_n\}\{i_n\} \rangle$
3-sequences:	$\langle \{i_1, i_2, i_3\} \rangle, \langle \{i_1, i_2, i_4\} \rangle, \ldots, \langle \{i_{n-2}, i_{n-1}, i_n\} \rangle, \ldots$ $\langle \{\underline{i_1}\}\{\underline{i_1}, i_2\} \rangle, \langle \{i_1\}\{i_1, i_3\} \rangle, \ldots, \langle \{i_{n-1}\}\{i_{n-1}, i_n\} \rangle, \ldots$ $\langle \{i_1, i_2\}\{i_2\} \rangle, \langle \{i_1, i_2\}\{i_3\} \rangle, \ldots, \langle \{i_{n-1}, i_n\}\{i_n\} \rangle, \ldots$ $\langle \{i_1\}\{i_1\}\{i_1\} \rangle, \langle \{i_1\}\{i_1\}\{i_2\} \rangle, \ldots, \langle \{i_n\}\{i_n\}\{i_n\} \rangle$

① 똑같은것이 여러번 등장가능 →
② 순서가 다르면 요소가 같아도 다른 sequence
↳ 매우 많아짐!

- The number of candidate sequences is even ***substantially larger*** than the number of candidate itemsets
  - Which makes their enumeration difficult

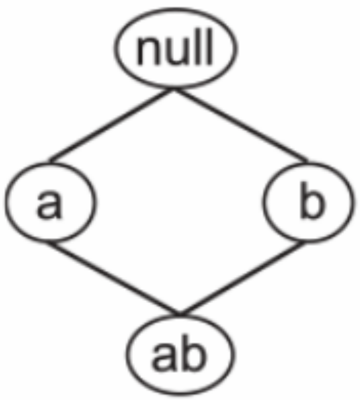
Reasons for Additional Candidates (1/2)

1. An item can appear at most once in an itemset, but an event can appear ***more than once*** in a sequence
   - (ex) Given two items i_1 and i_2
     - Candidate 2-itemsets: $\{i_1, i_2\}$ (only one)
     - Candidate 2-sequences: $<\{i_1\}\{i_1\}>$, $<\{i_1\}\{i_2\}>$, $<\{i_2\}\{i_1\}>$, $<\{i_2\}\{i_2\}>$, $<\{i_1, i_2\}>$

2. Order ***matters*** in sequence, but not for itemsets
   - (ex) Given two items i_1 and i_2
     - $\{i_1, i_2\}$ and $\{i_2, i_1\}$ are the same itemset
     - $<\{i_1\}\{i_2\}>$, $<\{i_2\}\{i_1\}>$, and $<\{i_1, i_2\}>$ are different sequences, and thus must be generated separately

Reasons for Additional Candidates (2/2)

3. For n items, the number of possible itemsets is $(2^n - 1)$, whereas the number of possible sequences are ***infinite***
   - (ex) comparing the number of itemsets with the number of sequences generated using two events (items)



Itemset Lattice

1-subsequences:

<{a}>, <{b}>

2-subsequences:

<{a} {a}>, <{a} {b}>, <{b} {b}>, <{a,b}>

3-subsequences:

<{a} {a} {a}>, <{a} {a} {b}>, <{a} {b} {a}>, <{a} {b} {b}>
<{b} {a} {a}>, <{b} {a} {b}>, <{b} {b} {a}>, <{b} {b} {b}>
<{a} {a,b}>, <{b} {a,b}>, <{a,b} {a}>, <{a,b} {b}>

⋮

Apriori Principle for Sequential Data

알고리즘은 그대로쓴다!

- Despite these challenges, the ***Apriori principle*** still holds
  - If a sequence is frequent, all of its subsequences ***must*** also be frequent
  - (ex) if $<\{a\}\{b\}>$ is frequent, then $<\{a\}>$ and $<\{b\}>$ must be frequent

- 차이점! Thus, it is possible to generate candidate k-sequences from the frequent $(k-1)$-sequences using the Apriori principle
  - ② (ex) $<\{a\}\{b\}\{c\}> + <\{b\}\{c\}\{d\}> = <\{a\}\{b\}\{c\}\{d\}>$

    1개 1개

    ↳ 앞뒤로하나뺀 부분이 일치해야함

- This allows us to extract sequential patterns from a sequence data set using an ***Apriori-like*** algorithm

①기존방식

{a, b, c}
{a, b, d} → {a, b, c, d}

(알파벳순 sort)

Apriori-Like Algorithm for Sequential Data

- Initially, find all frequent 1-subsequences, F_1 원소가 1개, minsup을 넘김
  - (ex) $F_1 = \{<\{a\}>, <\{b\}>, <\{c\}>, <\{d\}>, \dots\}$

- Next, iteratively perform the following for $k = 2, 3, \dots$ 자기랑 자기를 결합해서 Ck를 만듦
  1. **Generate** candidate k-sequences C_k from frequent $(k-1)$-sequences $F_{k\text{-}1}$
  2. **Prune** candidates in C_k whose $(k-1)$-subsequences are infrequent
  3. **Determine** F_k by making an additional pass over the data set and counting the supports of the remaining candidates in C_k
  4. Terminate if $F_k = \varnothing$

$<\{a\},\{b\},\{c\}>$
$<\{b\},\{c\},\{d\}>$ $\rightarrow <\{a\},\{b\},\{c\},\{d\}>$

단점 T.T 수가 너무 많다

✓ Notice that the structure of the algorithm is almost ***identical*** to Apriori algorithm for frequent itemset discovery

Candidate Generation

- We generate candidate k-sequences by ***merging*** a pair of frequent $(k-1)$-sequences
  - (ex) $<\{1\}\{2\}\{3\}> + <\{2\}\{3\}\{4\}> = <\{1\}\{2\}\{3\}\{4\}>$

- Although this approach is similar to the $F_{k-1} \times F_{k-1}$ strategy for generating candidate itemsets, there are certain ***differences***:

  ① We can merge a $(k-1)$-sequence with ***itself*** to produce a k-sequence
  - (ex) $<\{a\}> + <\{a\}> = <\{a\}\{a\}>$

  ② Although we still use the lexicographical order for arranging ***events within an element***, the arrangement of ***elements in a sequence*** may ***not*** follow the lexicographical order
  - (ex) $<\{b,c\}\{a\}\{d\}>$ (O), $<\{c,b\}\{a\}\{d\}>$ (X) 순서를 바꾸면 안됨!
  - (ex) $<\{a\}\{b\}\{c\}>$ (O), $<\{c\}\{b\}\{a\}>$ (O) transition 내의 alphabet sorting (o)

Sequence Merging Procedure (1/2)

k-1

- Let s_1 and s_2 be two sequences
  - We arrange the events within every elements ***lexicographically***
  - (ex) $<\{a, b\}, \{b, c, d\}>$ (O), $<\{b, a\}, \{c, d, b\}>$ (X) → 순서바꾸면안됨!
- We merge s_1 and s_2 ***only if***
  - Let s'_1 be the subsequence obtained by dropping the ***first event*** in s_1
  - Let s'_2 be the subsequence obtained by dropping the ***last event*** in s_2
  - We merge s_1 with s_2 only if $s'_1 = s'_2$
- Examples coding할수있어야함..!
  - $<\{1\}\{2\}\{3\}> + <\{2\}\{3\}\{4\}> = <\{1\}\{2\}\{3\}\{4\}>$
  - $<\{1\}\underbrace{\{5\}\{3\}}_{s'_1}> + <\underbrace{\{5\}\{3,}_{s'_2}4\}> = <\{1\}\{5\}\{3,4\}>$

Sequence Merging Procedure (2/2)

- How can we merge s_1 with s_2 to obtain the merged sequence?

- **Case 1**: If the last element of s_2 has only one event
  - Append the last element of s_2 to the end of s_1
  - (ex) $<\{1\}\,\{2\}\,\{3\}> + <\{2\}\,\{3\}\,\{4\}> = <\{1\}\,\{2\}\,\{3\}\,\{4\}>$

두번째 sequence의
마지막 transition이 1개의 event를 가지고있을때로일반화

- **Case 2**: If the last element of s_2 has more than one event
  - Append the last event from the last element of s_2 to the last element of s_1
  - (ex) $<\{1\}\,\{5\}\,\{3\}> + <\{5\}\,\{3,4\}> = <\{1\}\,\{5\}\,\{3,4\}>$

두개이상의 event

Analysis of the Merging Procedure (1/2)

어떤 frequent한 것을
→ 빼놓지 않고 생성돼야함

- The sequence merging procedure is ***complete***
  - i.e., it generate ***every*** frequent k-sequences
  - This is because every frequent k-sequences s includes
    - A frequent $(k - 1)$-sequence s_1 that does not contain the last event of s
    - A frequent $(k - 1)$-sequence s_2 that does not contain the first event of s
  - Since s_1 and s_2 are frequent and follow the criteria for merging sequences, they will be merged to produce every frequent k-sequences s

s_1, s_2 모두 frequent,
그들을 결합

$$\underbrace{<\{1\}\{2\}\{3\}>}_{s_1} + \underbrace{<\{2\}\{3\}\{4\}>}_{s_2} = \underbrace{<\overbrace{\{1\}\{2\}\underbrace{\{3\}}\{4\}}^{s_1}>}_{s}$$

$$s = <\overbrace{\{1\}\{2\}\{3\}}^{s_1}\{4\}> = <\{1\}\underbrace{\{2\}\{3\}\{4\}}_{s_2}>$$

Analysis of the Merging Procedure (2/2)

- The sequence merging procedure is ***non-redundant***
  - i.e., it does not generate duplicate candidate sequences
  - Because the sequence merging procedure ensures that there is a ***unique*** way of generating s only by merging s_1 and s_2
  - Examples 시험!
    - $<\{1\}\{2,5\}> + <\{2,5\}\{3\}> = <\{1\}\{2,5\}\{3\}>$ (allowed) ✓ 이방법만 가능!
    - $<\{1\}\{2\}\{3\}> + <\{1\}\{2,5\}> = <\{1\}\{2,5\}\{3\}>$ (***not*** allowed)
    - In other words, $<\{1\}\{2,5\}\{3\}>$ is generated ***only*** by merging $<\{1\}\{2,5\}>$ and $<\{2,5\}\{3\}>$

candidate를 만든뒤 pruning을 할때

① $<\{2,5\},\{3\}>$

② $<\{1\},\{5\},\{3\}>$

③ $<\{1\},\{2\},\{3\}>$

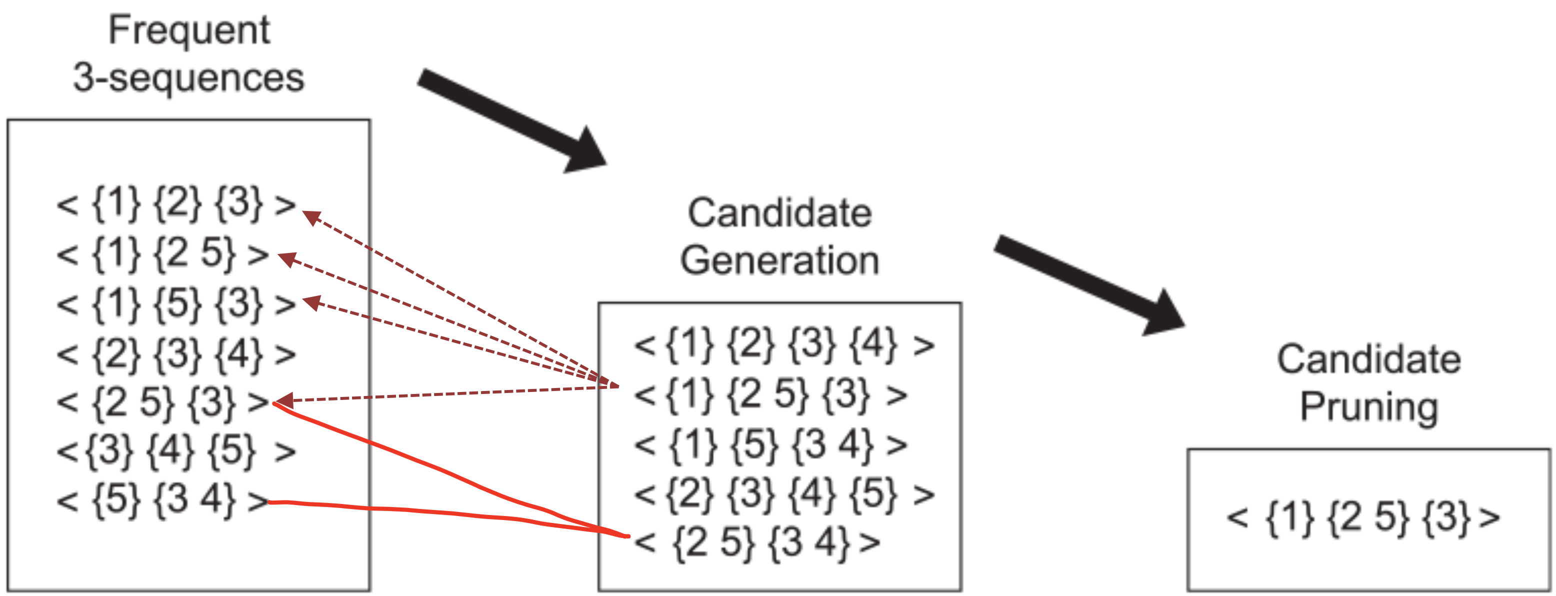
④ $<\{1\},\{2,5\}>$

→ 모두 frequent한지 확인

Candidate Pruning

- We prune a candidate k-sequence if at least one of its $(k-1)$-sequence is ***infrequent***

- Example
  - $<\{1\}\{2\}\{3\}\{4\}>$ can be eliminated because $<\{1\}\{2\}\{4\}>$ is infrequent
  - $<\{1\}\{2,5\}\{3\}>$ survives because all of its 3-sequences are frequent



Support Counting

- During support counting
  - We identify all candidate *k*-sequences belonging to a particular data sequence and increment their support counts
- After performing this step for each data sequence
  - We identify the frequent *k*-sequences and discard all candidate sequences whose support < *minsup*

얘를보고
얘를확인

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

Sequence data set

<{1} {2} {3}> → *support* = 1/5 = 20%
<{1} {2,5}> → *support* = 2/5 = 40%
<{2,5} {3}> → *support* = 1/5 = 20%

Candidate 3-sequences