ASSOCIATION RULE MINING

Week13

Association Rule

What is Market Basket Analysis?

- Finding some useful information in 'market basket'
- What kinds of information?
  - Who customers are
  - Which products tend to be purchased together
  - Why some products tend to be purchased together
- Association rule: Information like "If item A then item B" $(A \Rightarrow B)$

Customer 1

. . .

Customer n

Collecting transactions & finding useful information

Point of Sale Transactions

- Transaction and item

Datetime	Customer	Items
2015-07-15 14:03	1	orange juice, banana
2015-07-15 16:20	2	orange juice, milk
2015-07-16 10:14	3	detergent, banana, orange juice
2015-07-25 19:34	2	milk, bread, soda
2015-07-29 09:41	4	detergent, window cleaner
2015-08-01 20:55	1	bread, milk

- Find pair of items that is more likely to be purchased together based on transactions
  - Banana and orange juice are more likely to be purchased together
  - Milk and bread are more likely to be purchased together

Association Rules

- Association rules obtained from transactions are like
  “If item A, then item B” 만약 A를 산다면 B도 산다
  - Rules are defined from co-occurrence of items in the same market basket

- Three types of rules
  - Useful: contains high quality, actionable information
    - On Thursday, customer who purchase diapers are likely to purchase beer
  - Trivial: already known by anyone familiar with the business
    - Customers purchasing paint buy pain brushes
  - Inexplicable: new but no explanation about customer behavior
    - When a new hardware store opens, one of the most commonly sold items is toilet rings

General Process for Finding Rules

1. • Gathering transactions for selected items 거래 데이터 수집.
2. • Check co-occurrence of set of items 항목의 동시 등장 빈도 계산
3. • Finding the most frequent combination from the matrix
4. • Making a distinction between "condition' and 'result' from the combination

조건과 결과를 구분하여 규칙 생성

Performance Measure for Rules

Rule: If 'condition', then 'result'

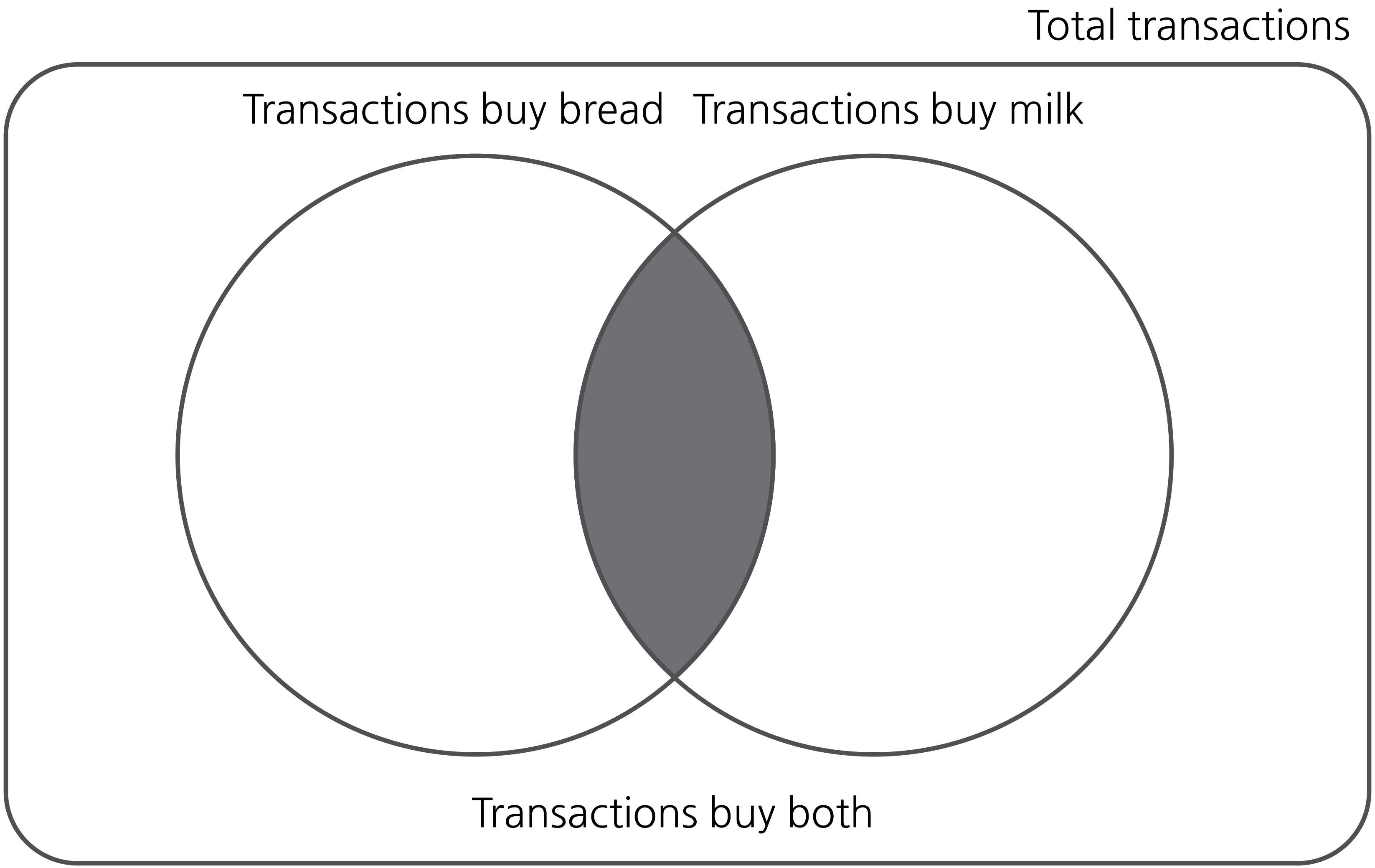
- Support
  - How many transactions that contain 'condition(X)' and 'result(Y)' simultaneously

$$\mathrm{Supp}(X \Longrightarrow Y) = \mathrm{Supp}(X \cup Y) =$$
$$= \frac{\#\ of\ transactions\ that\ include\ both\ condition\ and\ result}{\#\ of\ total\ transactions}$$

- Confidence
  - How many transactions that contain 'condition(X)' and 'result(Y)' among transactions including 'condition'

$$\mathrm{Conf}(X \Longrightarrow Y) = P(Y|X) = \frac{\mathrm{Supp}(X \cup Y)}{\mathrm{Supp}(X)}$$
$$= \frac{\#\ of\ transactions\ that\ include\ both\ condition\ and\ result}{\#\ of\ transactions\ that\ include\ condition}$$

Performance Measure for Rules



Performance Measure for Rules

- Low support
  - 거의 발생X
  - This rule rarely happens → not interesting
- High support, but low confidence
  - Both 'condition' and 'result' are quite often observed, but comparing with the number of transactions that include condition are much more
  - The reason that support of the rule is high may be that the number of transactions that include condition is high
- High support and high confidence
  - This rule is significant rule
  - However, high support and high confidence do not guarantee usefulness of the rule

Example: Association Rule

- Example rules from given transactions

TID	Items
1	bread, milk, butter
2	bread, butter
3	bread, juice, butter
4	bread, beer
5	beer, juice

  - If bread, then butter (bread ⇒ butter)

$$support = \frac{3}{5}, confidence = \frac{3}{4}$$

  - If beer, then bread (beer ⇒ bread)

$$support = \frac{1}{5}, confidence = \frac{1}{2}$$

Performance Measure for Rules

- Lift or improvement
  - How much better a rule is at predicting the result than just guessing the result at random

$$\text{lift}(X \Longrightarrow Y) = \frac{P(Y|X)}{P(Y)} = \frac{\text{Supp}(X \cup Y)}{\text{Supp}(X) \times \text{Supp}(Y)}$$

$$= \frac{(\#\ of\ transactions\ that\ include\ both\ condition\ and\ result) \times (\#\ of\ transactions)}{(\#\ of\ transactions\ that\ include\ condition)(\#\ of\ transactions\ that\ include\ result)}$$

Improvement	Interpretation	Example
1	Two items are independent 독립	pepper and cookies
>1	Complementary 보완	Bread and butter
<1	Substitutional 대체	Butter and margarine

In this case, If A, then NOT B is better than If A, then B

Performance Measure for Rules

- Conviction
  - Conviction measures the implication strength of the rule from statistical independence

$$\text{conv}(X \Longrightarrow Y) = \frac{1 - \text{supp}(Y)}{1 - \text{conf}(X \Longrightarrow Y)} = \frac{P(X) \times P(\sim Y)}{P(X \cup \sim Y)}$$

    - $P(\sim Y)$ is the probability that Y does not appear in a transaction
  - Conviction compares the probability that X appears without Y if they were dependent with the actual frequency of the appearance of X without Y
  - Unlike confidence, conviction factors in both $P(X)$ and $P(Y)$ and always has a value 1 when the relevant items are completely unrelated.
  - In contrast to lift, conviction is directed measure because it also uses the information of the absence of the consequent

Question

- Calculate performance measures of the rule
  - Rule1: $a \Longrightarrow b$
  - Rule2: $e \Longrightarrow f$
  - Rele3: $b \text{ and } c \Longrightarrow g$

TID	Items
1	b, c, g
2	a, b, d, e, f
3	a, b, c, g
4	b, c, e, f
5	b, c, e, f, g

1) Calculate support of above three rules $\frac{2}{5}$

2) Calculate confidence of above three rules $\frac{2}{2} = 1$

3) Calculate lift of above three rules $\frac{\frac{2}{2}}{\frac{2}{5} \times \frac{5}{5}} = 1$

4) Calculate conviction of above three rules

$$\frac{1 - \text{supp}(Y)}{1 - \text{confidence}(X \Rightarrow Y)} =$$

Pros and Cons of Market Basket Analysis

Pros	Cons
• Produces understandable and clear results (association rules)	• Require much more computation resource as the problem size grows
• Handle transactions themselves	• Sometimes require to utilize the taxonomy for mining better rules and reducing complexity
• Computational method is simple to implement and understand	• Discount rare items

효과적으로
association rule을 생성하는 알고리즘 하나

Apriori Algorithm

Practical Issues on Market Basket Analysis

- Exponential growth on distinct combinations as the number of items increases
  - If 100 items are sold in the store, the number of combinations with 3 items

$$C(100,3) = \frac{100!}{3!\,97!} = \frac{100 \times 99 \times 98}{3 \times 2} = 161{,}700$$

- Methods to solve rapid growth on problem size
  - Use the taxonomy: generalize items that can meet criterion
    - Vanilla ice cream ∈ Ice cream ∈ Frozen food ∈ Food
    - When there are too many items to handle, use higher level of category instead to reduce combinations
  - Use pruning: throw out item or combination of items that do not meet criterion
    - Minimum support pruning is the most common method

Apriori Algorithm

- Apriori is the algorithm to mine rules from transactions
  - Key idea is that any subsets of a frequent item set are also frequent item sets

$\{1,2,3\}$ is frequent item set $\Rightarrow \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}$ are frequent item set

Phase 1

- Find all frequent item sets having specified minimum support s_{min}

Phase 2

- Consider a subset A of a frequent item set L
  - For a specified confidence c_{min}
    if $support(L)/support(A) \geq c_{min}$,
    then generate a rule $R: A \Rightarrow (L - A)$

Apriori Algorithm - Phase 1

1

- [initial step] Specify the minimum support s_{min} and set $k = 1$
  - $C_1 = \{\{i_1\}, \{i_2\}, \dots, \{i_n\}\}$ $L_1 = \{c \in C_1 | support(c) \geq s_{min}\}$

2

- Set $k = k$ and Generate new candidate item sets C_k from L_{k-1}
  - Generate item sets C_k by joining like $C = L_{k-1} \times L_{k-1}$
  - Delete all item sets whose any subsets are not in L_{k-1} from C_k

3

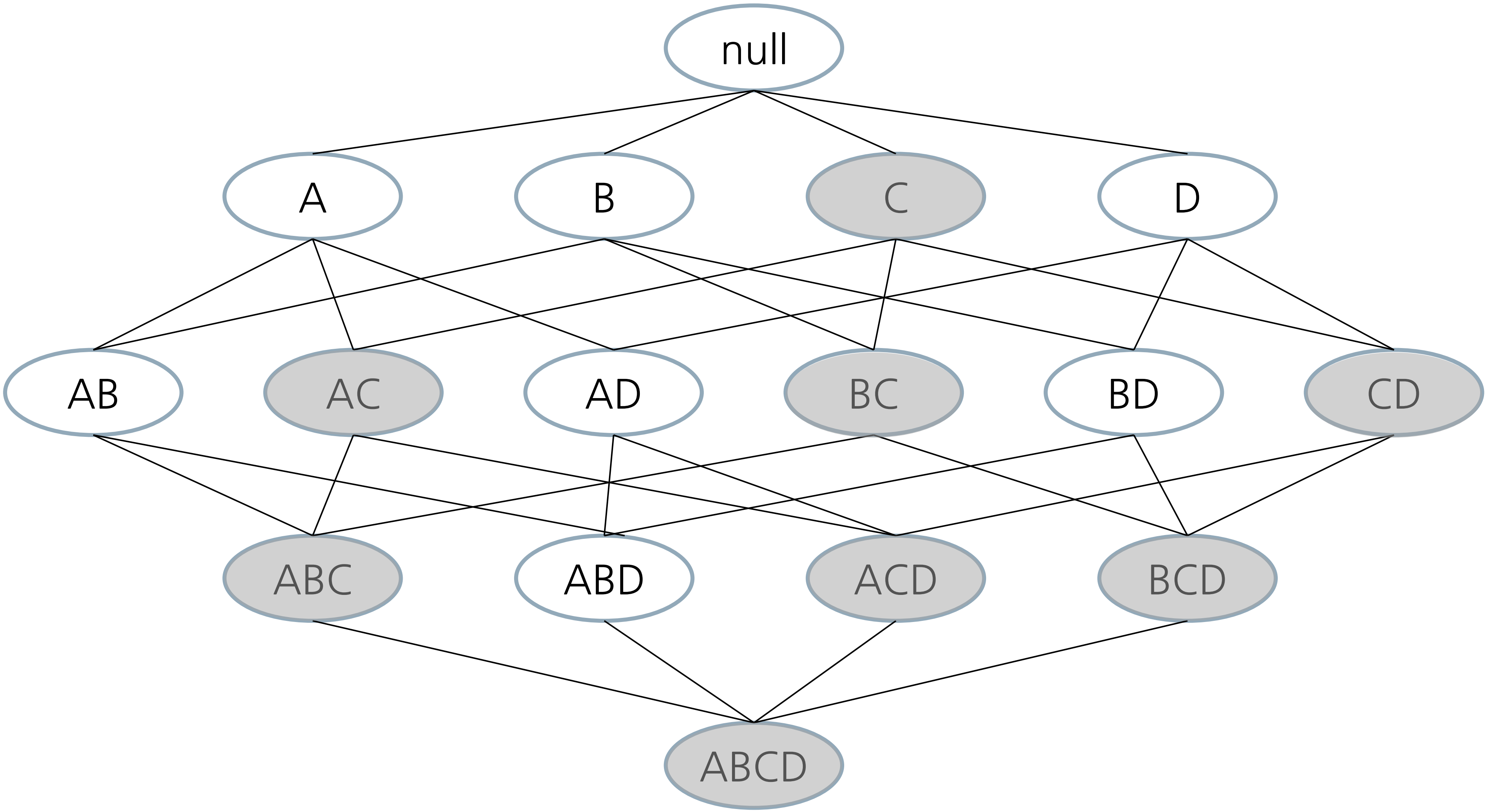
- Generate L_k such that $L_k = \{c \in C_k | support(c) \geq s_{min}\}$

4

- Repeat step 2 and 3 until $C_k = \phi$

Apriori Algorithm - Phase 1

- Key idea of phase 1



Example: Apriori Algorithm - Phase 1

- Generate C_k and L_k
  - Set s_{min}=0.4

TID	Items
1	b, c, g
2	a, b, d, e, f
3	a, b, c, g
4	b, c, e, f
5	b, c, e, f, g

Candidate Item set

$C_1 = \{\{a\},\{b\},\{c\},\{d\},\{e\},\{f\},\{g\}\}$

Remove infrequent item sets

$L_1 = \{\{a\},\{b\},\{c\},\{e\},\{f\},\{g\}\}$

Generate item sets by joining

$C_2 = \{\{a,b\},\{a,c\},\{a,e\},\{a,f\},\{a,g\},\{b,c\},\{b,e\},\{b,f\},\{b,g\},$
$\{c,e\},\{c,f\},\{c,g\},\{e,f\},\{e,g\},\{f,g\}\}$

$L_2 = \{\{a,b\},\{b,c\},\{b,e\},\{b,f\},\{b,g\},\{c,e\},\{c,f\},\{c,g\},\{e,f\}\}$

$C_3 = \{\{b,c,e\},\{b,c,f\},\{b,c,g\},\{b,e,f\},\{c,e,f\}\}$

$L_3 = \{\{b,c,e\},\{b,c,f\},\{b,c,g\},\{b,e,f\},\{c,e,f\}\}$

$\{a,b,c\}$ is removed from C_3 because $\{a,c\}$ does not belong to L_2

$C_4 = \{\{b,c,e,f\}\}$

$L_4 = \{\{b,c,e,f\}\}$

Question

- Generate C_k and L_k
  - Set s_{min}=0.4

TID	Items
1	bread, milk, butter
2	bread, butter
3	bread, juice, butter
4	bread, beer
5	beer, juice

1) Generate C_1 and L_1

C_1 = {{bread} {milk} {butter} {juice} {beer}}

L_1 = {{bread} {butter} {juice} {beer}}

2) Generate C_2 and L_2

C_2 = {{bread, butter} {bread, juice} {bread, beer} {beer, juice} {butter, beer}
{juice, beer}}

Example: Apriori Algorithm - Phase 2

- Rule generation
  - Candidate frequent item set $L = \{b, c, g\}$
  - Rules having 1 item in result

$$R_1: \{b, c\} \Longrightarrow \{g\}$$
$$R_2: \{b, g\} \Longrightarrow \{c\}$$
$$R_3: \{c, g\} \Longrightarrow \{b\}$$

TID	Items
1	b, c, g
2	a, b, d, e, f
3	a, b, c, g
4	b, c, e, f
5	b, c, e, f, g

Rule	Support($\{b, c, g\}$)	Support(condition)	Confidence
$R_1: \{b, c\} \Longrightarrow \{g\}$	0.6	0.8	0.6/0.8=0.75
$R_2: \{b, g\} \Longrightarrow \{c\}$	0.6	0.6	0.6/0.6=1
$R_3: \{c, g\} \Longrightarrow \{b\}$	0.6	0.6	0.6/0.6=1

How to Efficiently Generate Rules

- Confidence is not anti-monotonic
  - $confidence(ABC \Longrightarrow D)$ can be larger or smaller than $confidence(AB \Longrightarrow D)$

- However, confidence of rules generated from the same item set is anti-monotonic with respect to the number of items in result
  - All conditions should be subsets of the largest condition

$$confidence(ABC \Rightarrow D) \geq confidnece(AB \Rightarrow CD) \geq confidence(A \Rightarrow BCD)$$

If the rule $ABC \Rightarrow D$ has lower confidence than certain value

Then, $\boldsymbol{BC \Rightarrow AD}$, $\boldsymbol{AC \Rightarrow BD}$, $\boldsymbol{BD \Rightarrow CD}$, $\boldsymbol{C \Rightarrow ABD}$, $\boldsymbol{B \Rightarrow ACD}$, $\boldsymbol{A \Rightarrow BCD}$ have lower confidence than certain value

Set Up s_{min}

- If s_{min} is too high, we can miss item sets containing interesting rare items (e.g., expensive products)

- If s_{min} is too low, the number of frequent item sets increases and computational cost becomes expensive

Always, it is really hard to set "appropriate" parameter

It is not effective to use a single s_{min}

Frequent Pattern Growth Algorithm

Frequent Pattern Growth Algorithm

- Why Frequent Pattern Growth (FP-Growth) ?
  - The Apriori algorithm is widely used for association rule mining, but it has significant drawbacks
    - It generates a large number of candidate itemsets, leading to high computational costs
    - It requires multiple scans of the database, which can be slow for large datasets

- FP-Growth solves these problems by
  - Using a tree-based data structure (FP-tree) to store frequent items compactly
  - Avoiding candidate generation, reducing computational complexity

Frequent Pattern Growth Algorithm

- The **FP-Growth algorithm** consists of two main steps
  1. Building the FP-Tree
  2. Mining frequent patterns from the FP-Tree

Frequent Pattern Growth Algorithm

- Step 1: Building the FP-Tree
  1. Scan the dataset once to find the frequency of all individual items
  2. Filter out infrequent items based on a minimum support threshold
  3. Sort the frequent items in descending order of support (ties are resolved arbitrarily)
  4. Construct the FP-tree
     - The root is a null node
     - For each transaction
       - Follow the path in the tree corresponding to its frequent items
       - If a node already exists for an item, increment its count
       - Otherwise, create a new node

Frequent Pattern Growth Algorithm

- Step 1-1: Scan the dataset once to find the frequency of all individual items

TID	Items
1	f, a, c, d, g, i, m, p
2	a, b, c, f, l, m, o
3	b, f, h, j, o
4	b, c, k, s, p
5	a, f, c, e, l, p, m, n

Item	Frequency	Item	Frequency
a	3	j	1
b	3	k	1
c	4	l	2
d	1	m	3
e	1	n	1
f	4	o	2
g	1	p	3
h	1	s	1
i	1		

Frequent Pattern Growth Algorithm

- Step 1-2: Filter out infrequent items based on a minimum support threshold
  - Let $s_{min} = 3$

Item	Frequency	Item	Frequency
a	3	j	1
b	3	k	1
c	4	l	2
d	1	m	3
e	1	n	1
f	4	o	2
g	1	p	3
h	1	s	1
i	1		

Item	Frequency
a	3
b	3
c	4
f	4
m	3
p	3

Frequent Pattern Growth Algorithm

- Step 1-3: Sort the frequent items in descending order of support

Item	Frequency
a	3
b	3
c	4
f	4
m	3
p	3

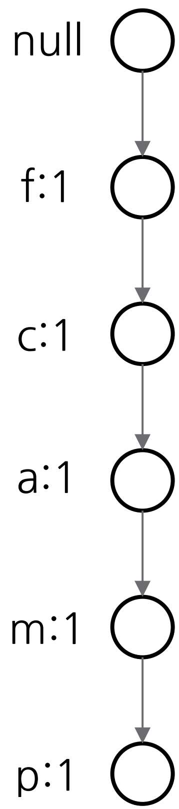
Item	Frequency
f	4
c	4
a	3
b	3
m	3
p	3

TID	Items
1	f, a, c, d, g, i, m, p
2	a, b, c, f, l, m, o
3	b, f, h, j, o
4	b, c, k, s, p
5	a, f, c, e, l, p, m, n

TID	Items
1	f, c, a, m, p
2	f, c, a, b, m
3	f, b
4	c, b, p
5	f, c, a, m, p

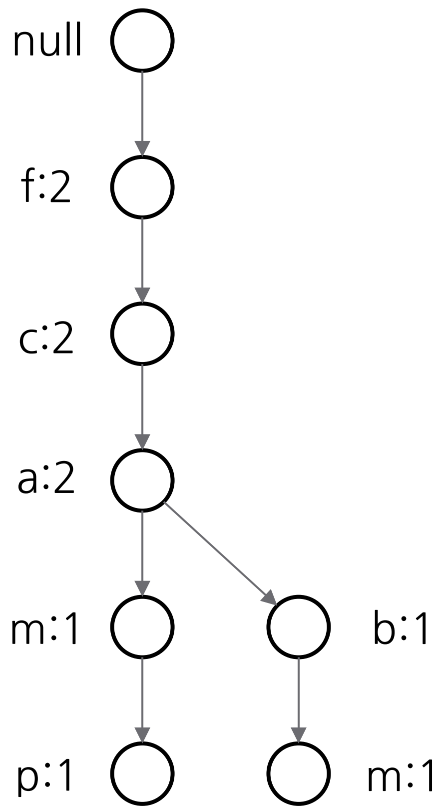
Frequent Pattern Growth Algorithm

- Step 1-4: Construct the FP-tree
  - Reading TID1= {f, c, a, m, p}



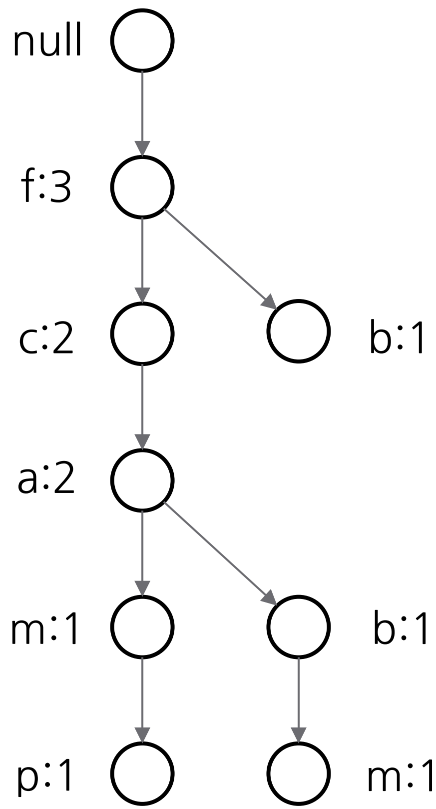
Frequent Pattern Growth Algorithm

- Step 1-4: Construct the FP-tree
  - Reading TID2= {f, c, a, b, m}



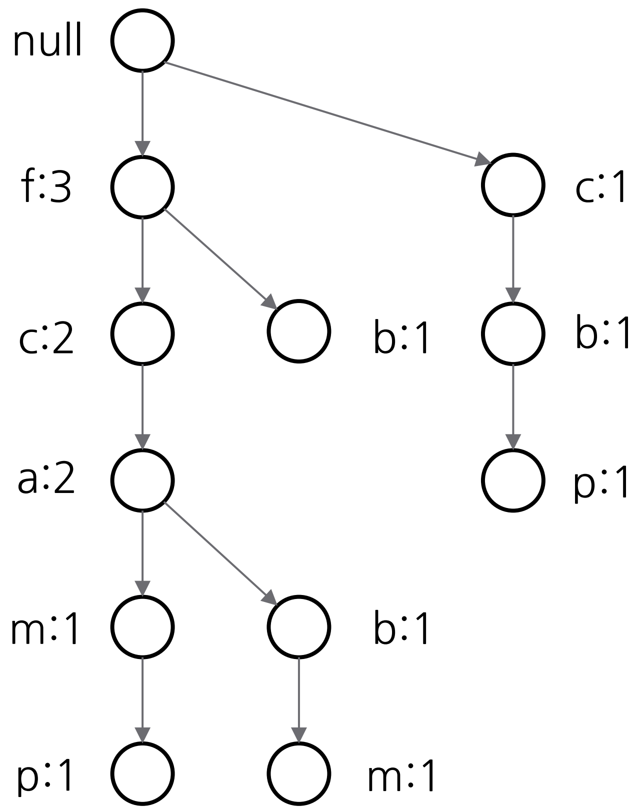
Frequent Pattern Growth Algorithm

- Step 1-4: Construct the FP-tree
  - Reading TID3= {f, b}



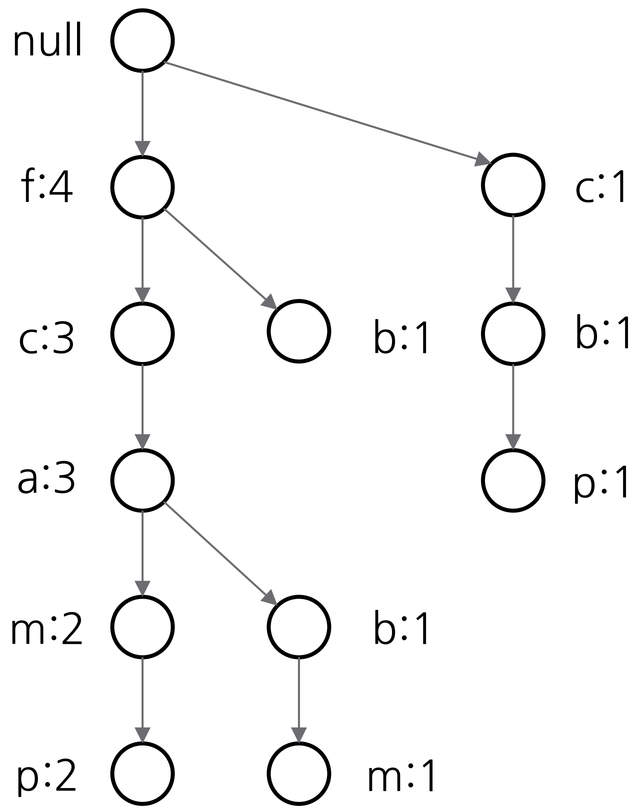
Frequent Pattern Growth Algorithm

- Step 1-4: Construct the FP-tree
  - Reading TID4= {c, b, p}



Frequent Pattern Growth Algorithm

- Step 1-4: Construct the FP-tree
  - Reading TID5= {f, c, a, m, p}

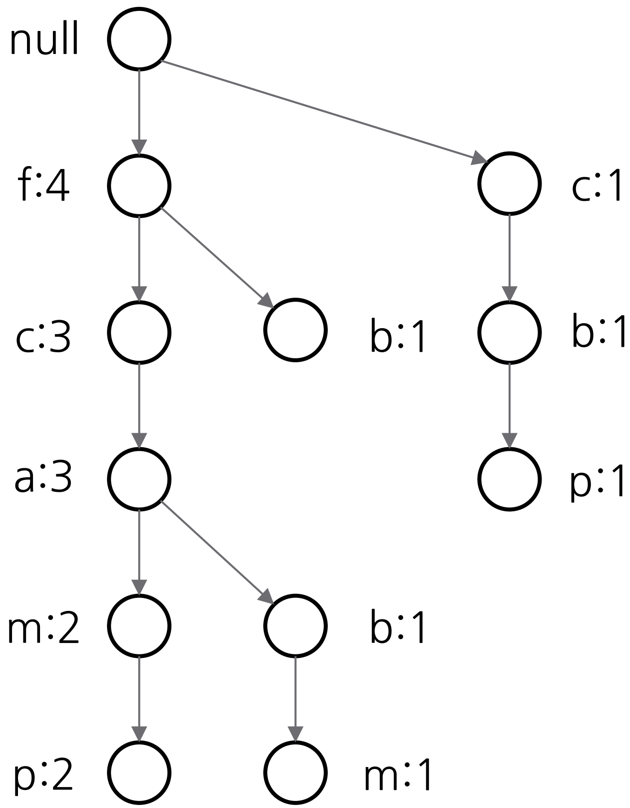


Frequent Pattern Growth Algorithm

- Step 2: Recursive Pattern Growth
  1. For each frequent item, compute conditional pattern bases
  2. Fore each frequent item, construct a conditional FP-tree
  3. Recursively mine the conditional FP-tree to find frequent patterns

Frequent Pattern Growth Algorithm

- Step 2-1: For each frequent item, compute conditional pattern bases



Item	Conditional pattern base
f	{}
c	{{f : 3}}
a	{{f, c : 3}}
b	{{f, c, a : 1}, {f : 1}, {c : 1}}
m	{{f, c, a : 2}, {f, c, a, b : 1}}
p	{{f, c, a, m : 2}, {c, b : 1}}

Frequent Pattern Growth Algorithm

- Step 2-2: For each frequent item, construct a conditional FP-tree

Item	Conditional pattern base	Conditional FP-tree
f	{}	{}
c	{{f : 3}}	{f:3}
a	{{f, c : 3}}	{f, c:3}
b	{{f, c, a : 1}, {f : 1}, {c : 1}}	{}
m	{{f, c, a : 2}, {f, c, a, b : 1}}	{f, c, a:3}
p	{{f, c, a, m : 2}, {c, b : 1}}	{c:3}

Frequent Pattern Growth Algorithm

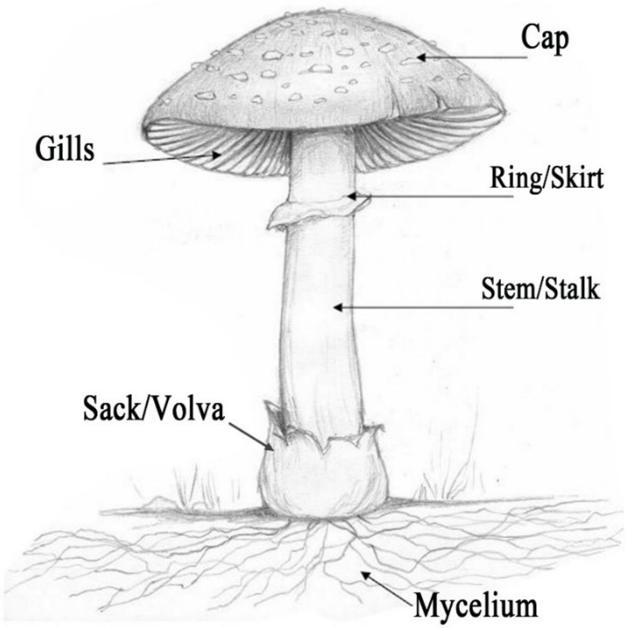
- Step 2-3: Recursively mine the conditional FP-tree to find frequent patterns

Item	Conditional pattern base	Conditional FP-tree	Frequent patterns
f	{}	{}	{}
c	{{f : 3}}	{f:3}	{〈f,c:3〉}
a	{{f, c : 3}}	{f, c:3}	{〈f, a : 3〉, 〈c, a : 3〉, 〈f, c, a:3〉}
b	{{f, c, a : 1}, {f : 1}, {c : 1}}	{}	{}
m	{{f, c, a : 2}, {f, c, a, b : 1}}	{f, c, a:3}	{ 〈f, m : 3〉, 〈c, m : 3〉, 〈a, m : 3〉, 〈f, c, m : 3〉 〈f, a, m : 3〉, 〈c, a, m :3〉, 〈f, c, a, m:3〉}
p	{{f, c, a, m : 2}, {c, b : 1}}	{c:3}	{〈c,p:3〉}

Applications

Applications of Association Rules

- Another example of association rule mining
  - Ex.) Mushroom data
    - Classify edibility based on the descriptions of mushroom
    - https://archive.ics.uci.edu/ml/datasets/mushroom



  - By applying association rule mining, it can be possible to characterize poisoned mushrooms and edible mushrooms

Applications of Association Rules

- Mushroom dataset
  - Attribute Information: (classes: edible=e, poisonous=p)

1) cap-shape: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
2) cap-surface: fibrous=f, grooves=g, scaly=y, smooth=s
3) cap-color: brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
4) bruises?: bruises=t, no=f
5) odor: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
6) gill-attachment: attached=a, descending=d, free=f, notched=n
7) gill-spacing: close=c,crowded=w,distant=d
8) gill-size: broad=b,narrow=n
9) gill-color: black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w,yellow=y
10) stalk-shape: enlarging=e,tapering=t
11) stalk-root: bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?

Applications of Association Rules

- Mushroom dataset
  - Attribute Information: (classes: edible=e, poisonous=p)

12) stalk-surface-above-ring: fibrous=f, scaly=y, silky=k, smooth=s
13) stalk-surface-below-ring: fibrous=f, scaly=y, silky=k, smooth=s
14) stalk-color-above-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
15) stalk-color-below-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
16) veil-type: partial=p, universal=u
17) veil-color: brown=n, orange=o, white=w, yellow=y
18) ring-number: none=n, one=o, two=t
19) ring-type: cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
20) spore-print-color: black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
21) population: abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
22) habitat: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d