

CS5228: Knowledge Discovery and Data Mining

Lecture 4 — Association Rule Mining

Course Logistics — Update

- Assignment 1

- Submission deadline: Thu, Sep 12 (11.59 pm)
- Honor code: don't cheat, don't copy, don't steal, don't plagiarize, etc.
- Don't forget to check Discussion and Errata page Canvas

- Project

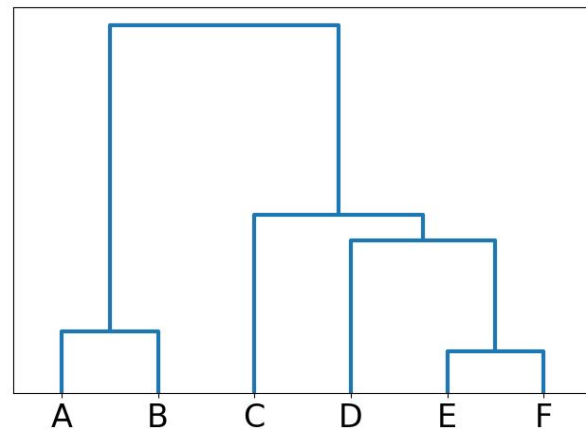
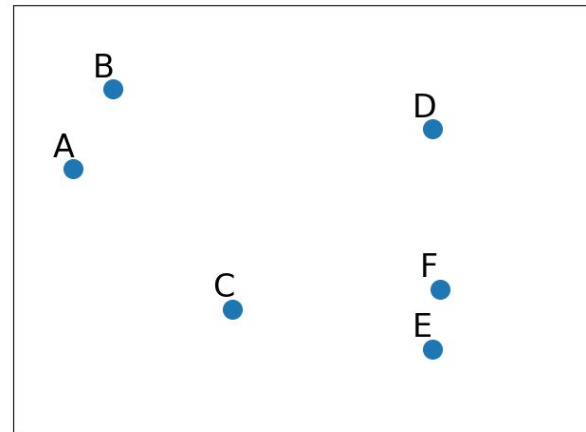
- Teams finalized (please get all in contact)
- Kaggle competition launched

Recap — Hierarchical Clustering

- AGNES (AGglomerative NESting)

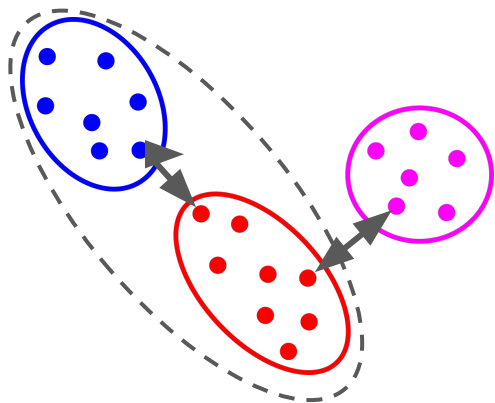
- Start with N clusters, one for each data point
- Iteratively merge nearest clusters into one
- Stop if all data points are in one cluster

- Core questions: How to calculate distances between clusters?

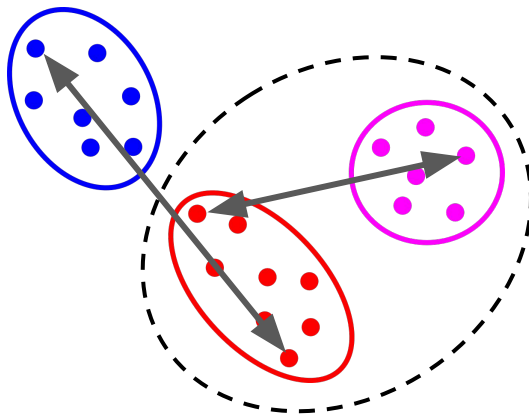


Recap — Linkage Methods

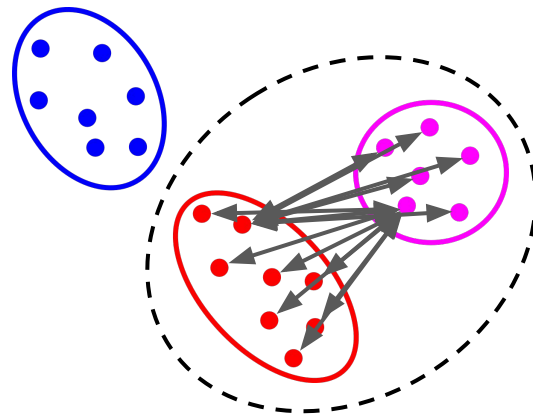
Single Linkage



Complete Linkage



Average Linkage



Recap — Cluster Evaluation

- If ground truth available: external quality measures

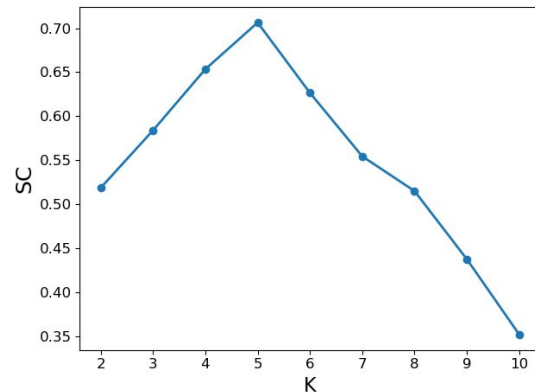
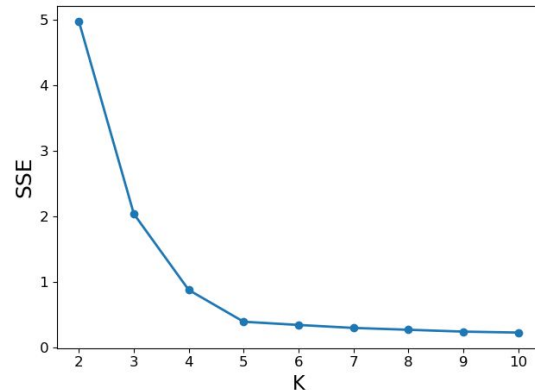
- Cluster purity
- TP/TN/FP/FN-based metrics (e.g., Rand index)

- Unlabelled data: internal quality measures

- Elbow method using SSE
 - Silhouette Coefficient (SC)
- } favor blob-like clusters

- Cluster evaluation in practice (unlabeled data)

- No fool-proof method to find "best" clustering
- Decision on clustering often rather pragmatic



Recap — Clustering as a Means to an End

- Clustering as part of EDA

- SSE plot, SC plot, dendrogram, etc. can provide useful insights into the data
- Little requirements — "only" similarity/distance between data points needed
- In the gray area between (simple) EDA and proper data analysis

- Clustering for data preprocessing — example:

- Cluster persons according to their height into $K=10$ groups
 - Assign each person new height = centroid of cluster
- } form of aggregation or binning & smoothing

Outline

- **Association Rule Mining**
 - Overview
 - Applications
- Definitions
- Algorithms
 - Brute-Force
 - Apriori
- Discussion

Association Rules — Basic Setup

- Input database:
 - Set of **transactions**
 - Transaction = set of **items**
- Output: **Association Rules**
 - Rules predicting the occurrence of some items based on occurrence of other items

antecedent \rightarrow **consequent**

$\{\text{item}_2, \text{item}_3\} \rightarrow \{\text{item}_5\}$

$\{\text{item}_1\} \rightarrow \{\text{item}_3\}$

TID	Items
1	item ₁ , item ₂ , item ₃ , item ₄ , item ₅
2	item ₂ , item ₃ , item ₅
3	item ₁ , item ₄ , item ₅
4	item ₂ , item ₃ , item ₅ , item ₆ , item ₇
5	item ₁ , item ₃ , item ₅ , item ₇
...	

Applications — Market Basket Analysis

- Understanding customers shopping behavior

- **Items:** products in supermarket/store
- **Transaction:** baskets at check-out

- Interesting rules:

- Customers who buy {a, b} also tend to buy {x, y}
- Example: {cereal}→{milk}

- Purpose

- Shelf management / item placement
- Promotions (product bundles)
- Recommendations
- Pricing strategies

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese
...	

Applications — Medical Data Analysis

- **Diagnosis Support Systems**

- Items: symptoms, diseases
- Transaction: patient's medical history

ID	Items
1	covid-19, anosmia, cough, fatigue
2	flu, anosmia, headache
3	covid-19, anosmia, headache, fatigue, fever
4	covid-19, flu, anosmia, fatigue
5	flu, depression, fatigue, fever, headache
...	



$\{\text{anosmia, fatigue}\} \rightarrow \{\text{covid-19}\}$

- **ADR discovery** (adverse drug reaction)

- Items: drugs, reactions/symptoms
- Transaction: patient's medical history

ID	Items
1	d_1, d_2, d_3 , rash, vomit
2	d_1, d_3 , headache, nausea, rash,
3	d_2, d_3 , nausea, vomit
4	d_1 , nausea, rash, vomit
5	d_3, d_4 , headache, depression
...	



$\{d_1\} \rightarrow \{\text{rash}\}$

Applications — Census Data Analysis

- Getting insights into a population
 - Items: demographic data
 - Transaction: census record
- Interesting rules:
 - Correlations among groups of people based on shared demographics
 - Example: {uni-grad, ≥ 30 } \rightarrow {high-income}
- Purpose
 - Policy & decision making
 - Resource allocation
 - Urban planning

TID	Items
1	female, ≥ 25 , uni-grad, hdb, single, high-income
2	male, ≥ 25 , uni-grad, hdb, single, mid-income
3	male, ≥ 25 , uni-grad, hdb, condo, high-income
4	male, ≥ 30 , uni-grad, condo, married, high-income
5	female, ≥ 30 , uni-grad, condo, married, high-income
...	

Applications — Behavior Data Analysis

- User preferences & linkings
 - Items: movies, songs, books, etc.
 - Transaction: viewing/listening/reading history
- Interesting rules (movies):
 - Viewer who watched movies {a, b} also watched movies {x, y}
 - Example: {Jaws}→{It}
- Purpose
 - Recommendation systems

TID	Items
1	Jaws, Halloween, Scream, It
2	Alien, Jaws, Scream, It
3	Tenet, Inception, Interstellar
4	Jaws, Halloween, It
5	Alien, Tenet Jaws, It
...	

Association Rules — Problem Statement

- Association rules are not "hard" rules
 - e.g., $\{\text{cereal}\} \rightarrow \{\text{milk}\}$ does not mean that customers always buy milk when buying cereal
 - each possible combination (e.g., $\{\text{yogurt, bread}\} \rightarrow \{\text{milk}\}$) is potential association rule
- Given d unique items $\rightarrow 3^d - 2^{d+1} + 1$ rules
 - $d = 6 \rightarrow 602$ possible rules!
- Association Rule Mining
 - Finding **interesting/significant** association rules
 - Finding such rules **efficiently**

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese
...	

Outline

- Association Rule Mining
 - Overview
 - Applications
- **Definitions**
- Algorithms
 - Brute-Force
 - A-Priori
- Discussion & Summary

Definitions — Itemset, K-itemset

- **Itemset**

- A subset of items

{bread}, {yogurt}, {bread, yogurt}, {milk}, {cereal},
{eggs}, {bread, milk}, {bread, milk, cereal}, ...

- **K-itemset**

- An itemset containing k items, e.g., k=3:

{bread, milk, cereal}, {bread, yogurt, cheese},
{yogurt, milk, cereal}, {yogurt, cereal, cheese},
{milk, cereal, cheese}, {bread, milk, eggs}, ...

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese

Definitions — Support Count, Support (for itemsets)

- Support count SC

- Number of transactions containing an itemset
- e.g., $SC(\{\text{bread, yogurt, milk}\}) = 2$

- Support S

- Fraction of transactions containing an itemset
- e.g., $S(\{\text{bread, yogurt, milk}\}) = 2/5$

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese

Definitions — Frequent Itemset

- **Frequent itemset**

- Itemset with a support greater or equal than a minimum threshold $minsup$
- e.g., all frequent itemsets if

$$minsup = 2/5$$

```
{yogurt}  
{milk}  
{cheese}  
{cereal}  
{bread}  
{bread, milk}  
{yogurt, milk}  
{bread, cereal}  
{cereal, milk}  
{bread, yogurt}  
{cereal, yogurt}  
{cereal, yogurt, milk}  
{bread, cereal, milk}  
{bread, yogurt, milk}
```

$$minsup = 3/5$$

```
{yogurt}  
{milk}  
{cereal}  
{bread}  
{bread, milk}  
{yogurt, milk}  
{cereal, milk}  
{bread, yogurt}
```

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese

Definitions — Association Rule

- **Association Rule**

- Implication expression $X \rightarrow Y$,
where X and Y are itemsets
- e.g., $\{\text{yogurt, milk}\} \rightarrow \{\text{bread}\}$

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese

Definitions — Support (for association rules)

- **Support** of an association rule

- Fraction of transactions containing all items of an association rule $X \rightarrow Y$

$$S(X \rightarrow Y) = \frac{SC(X \cup Y)}{N} = S(X \cup Y)$$

↙
#transactions

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese
...	

$$S(\{yogurt, milk\} \rightarrow \{bread\}) = \frac{SC(\{yogurt, milk, bread\})}{N} = 2/5$$

$$S(\{yogurt, bread\} \rightarrow \{milk\}) = \frac{SC(\{yogurt, milk, bread\})}{N} = 2/5$$

Definitions — Confidence

- **Confidence** of an association rule $X \rightarrow Y$

- Probability of Y given X

$$C(X \rightarrow Y) = \frac{S(X \rightarrow Y)}{S(X)} = \frac{S(X \cup Y)}{S(X)}$$

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese

$$C(\{yogurt, milk\} \rightarrow \{bread\}) = \frac{S(\{yogurt, milk, bread\})}{S(\{yogurt, milk\})} = 2/3$$

High Support, High Confidence → Interesting Rules

$X \rightarrow Y$	Low Support	High Support
Low Confidence	<ul style="list-style-type: none">• The items in $(X \cup Y)$ do not frequently appear together• Even if the items in X appear together, they do so often without the items in Y	<ul style="list-style-type: none">• The items in $(X \cup Y)$ frequently appear together• If the items in X appear together, they often do so without the items in Y
High Confidence	<ul style="list-style-type: none">• The items in $(X \cup Y)$ do not frequently appear together• If the items in X appear together, they often do so with the items in Y	<ul style="list-style-type: none">• The items in $(X \cup Y)$ frequently appear together• If the items in X appear together, they do so often with the items in Y

Quick Quiz

Given an association rule R, **which inequality** regarding the support $S(R)$ and confidence $C(R)$ **holds**?

A

$$S(R) > C(R)$$

B

$$S(R) \geq C(R)$$

C

$$S(R) \leq C(R)$$

D

$$S(R) < C(R)$$

Outline

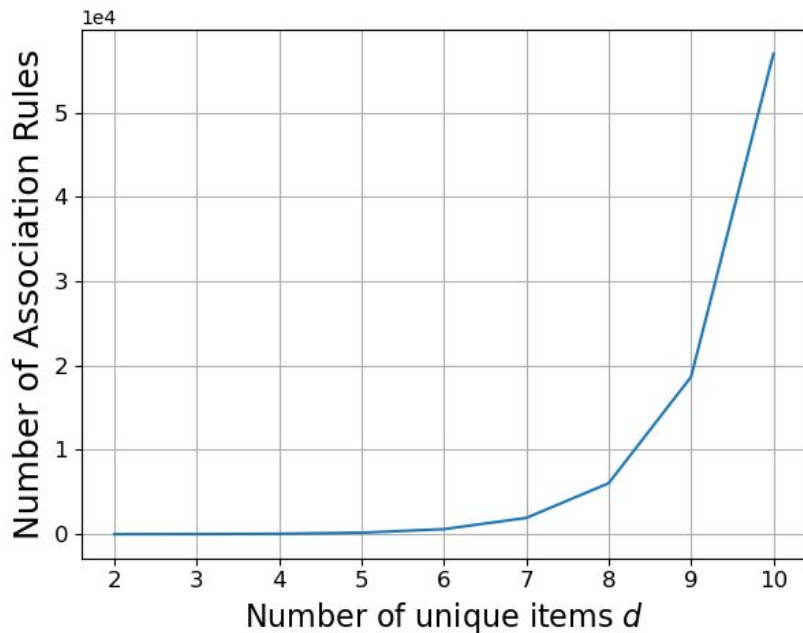
- Association Rule Mining
 - Overview
 - Applications
- Definitions
- **Algorithms**
 - **Brute-Force**
 - A-Priori
- Discussion & Summary

Brute Force Approach — Algorithm

- Given a set of transactions,
find all association rules $X \rightarrow Y$ with
 - Support $S(X \rightarrow Y) \geq \textit{minsup}$
 - Confidence $C(X \rightarrow Y) \geq \textit{minconf}$
- Brute force algorithm
 - List all possible association rules $X \rightarrow Y$
 - Calculate support $S(X \rightarrow Y)$ and confidence $C(X \rightarrow Y)$ for each rule
 - Drop rules with $S(X \rightarrow Y) < \textit{minsup}$ and $C(X \rightarrow Y) < \textit{minconf}$

Brute Force Approach — Computation Complexity

- Given d unique items $\rightarrow 3^d - 2^{d+1} + 1 \in O(3^d)$ rules
 - $d = 6 \rightarrow 602$ (theoretically) possible rules!



Average number items carried in a
supermarket in 2019

Source: FMI

28,112

<https://www.fmi.org/our-research/supermarket-facts>

Brute Force Approach — Computation Complexity

- Let w be the maximum number of items in a transaction within the database → $O(N \cdot (3^w - 2^{w+1} + 1))$ rules
■ $N = 5, w = 4 \rightarrow \leq 250$ "available" rules!

(typically $w \ll d$)

The difference between 250 and 602 seems negligible, but this is only because in this toy example, $d = 6$ and $w = 4$ are of the same magnitude.

The number 250 also ignores duplicate rules.

True number of different rules: 154

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese

N {

}

w

Decoupling Support and Confidence

- **Recall** $S(X \rightarrow Y) = \frac{SC(X \cup Y)}{N}$

$$\left. \begin{array}{l} S(\{yogurt, milk\} \rightarrow \{bread\}) \\ S(\{yogurt, bread\} \rightarrow \{milk\}) \\ S(\{milk, bread\} \rightarrow \{yogurt\}) \end{array} \right\} = \frac{SC(\{yogurt, milk, bread\})}{N} = S(\{yogurt, milk, bread\})$$

- **Observation 1**

- A rule $X \rightarrow Y$ has only sufficient support if $X \cup Y$ is a frequent itemset
- No need to calculate confidence of rules where $X \cup Y$ is not a frequent item set

$$S(X \rightarrow Y) \geq \text{minsup} \iff S(X \cup Y) \geq \text{minsup}$$

Two-Part Algorithm for Mining Association Rules

- Part 1 — Frequent Itemset Generation

- Generate itemsets with support $\geq \text{minsup}$
- "Only" $2^d - 1$ possible itemsets to check

- Part 2: — Association Rule Generation

- Generate rules from frequent itemsets
- Return rules with confidence $\geq \text{minconf}$

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese

↓ *minsup*

Frequent itemsets:

{milk}, {cereal, milk}, {bread, milk}, ...

↓ *minconf*

Association rules:

{cereal} → {milk}

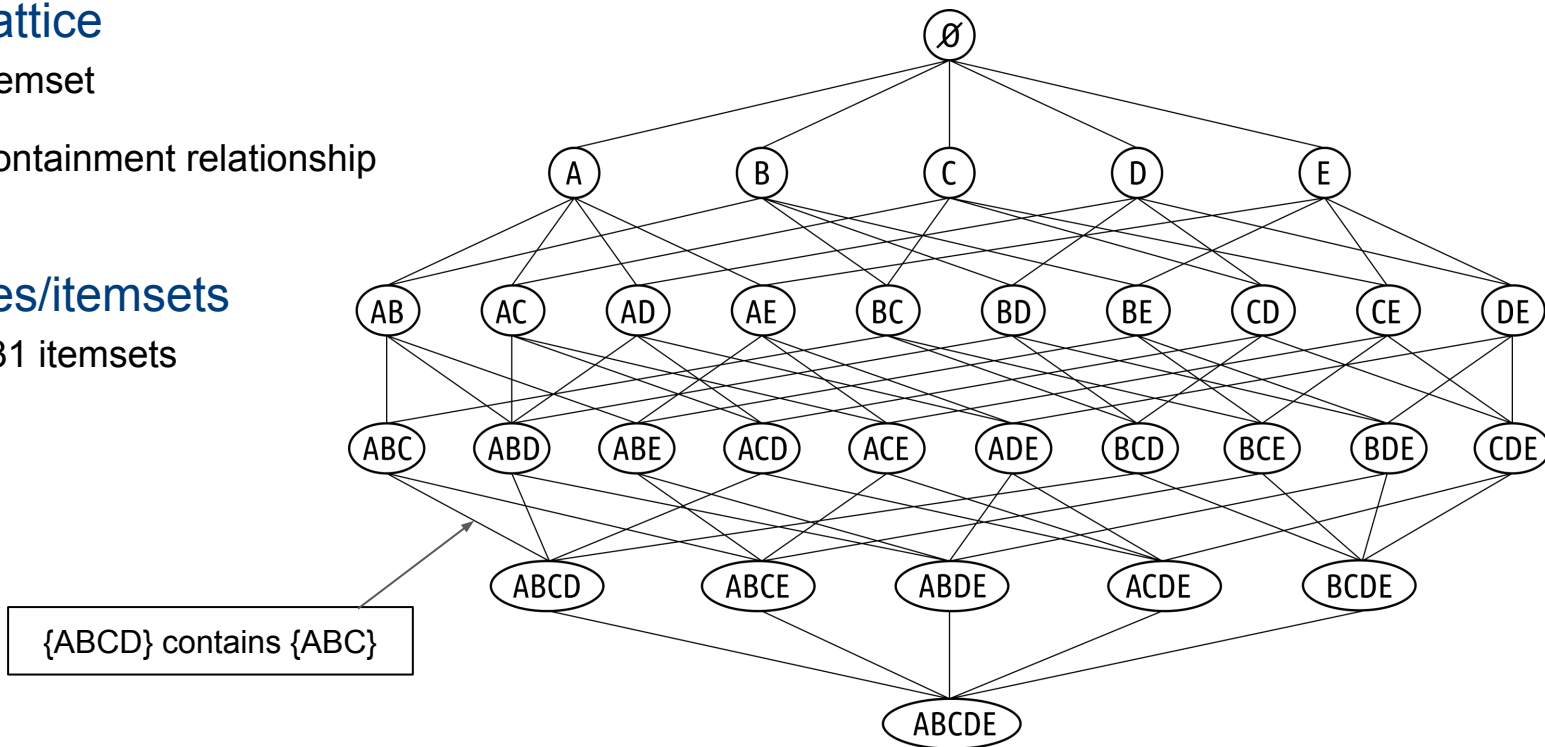
Frequent Itemset Generation

- Itemset lattice

- Node: itemset
- Edge: containment relationship

- $2^d - 1$ nodes/itemsets

- $d=5 \rightarrow 31$ itemsets



Frequent Itemset Generation — Brute Force Algorithm

```
support_counts ← dict({})  
for each transaction t in database:  
    for k in 1..t.length:  
        k_itemsets ← generate_itemsets(t, k)  
        for each itemset in k_itemsets:  
            support_counts[itemset] += 1
```

Global counter for all found itemsets

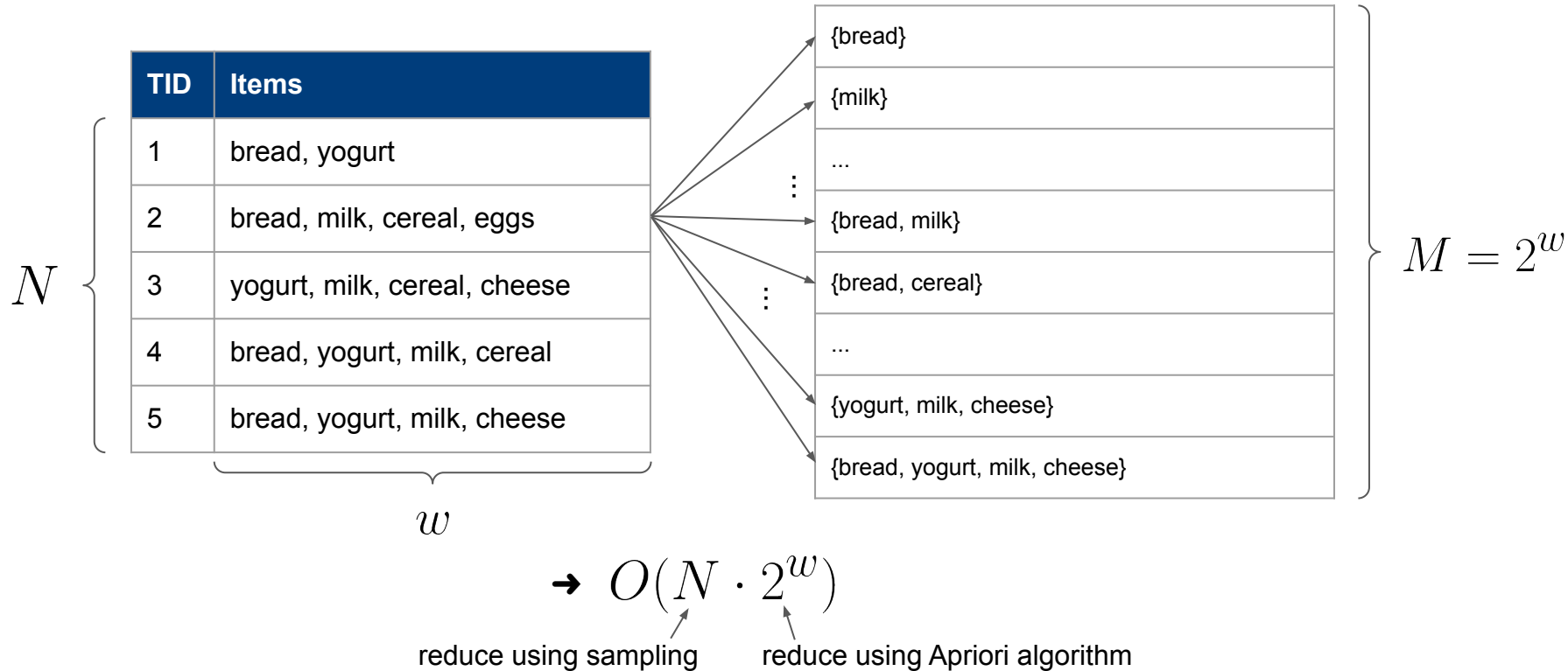
For each transaction, generate k-itemsets, with
k = 1, 2, 3, ... (up to #items in transaction)

For k-itemset, increase its global counter by 1

Question: Why do we need to count 1-itemsets
if an association rule requires at least 2 items?

Frequent Itemset Generation — Brute Force Algorithm

- Complexity Analysis



Outline

- Association Rule Mining
 - Overview
 - Applications
- Definitions
- **Algorithms**
 - Brute-Force
 - **A-Priori**
- Discussion & Summary

Apriori Principle (Anti-Monotonicity Principle)

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese
...	

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese
...	

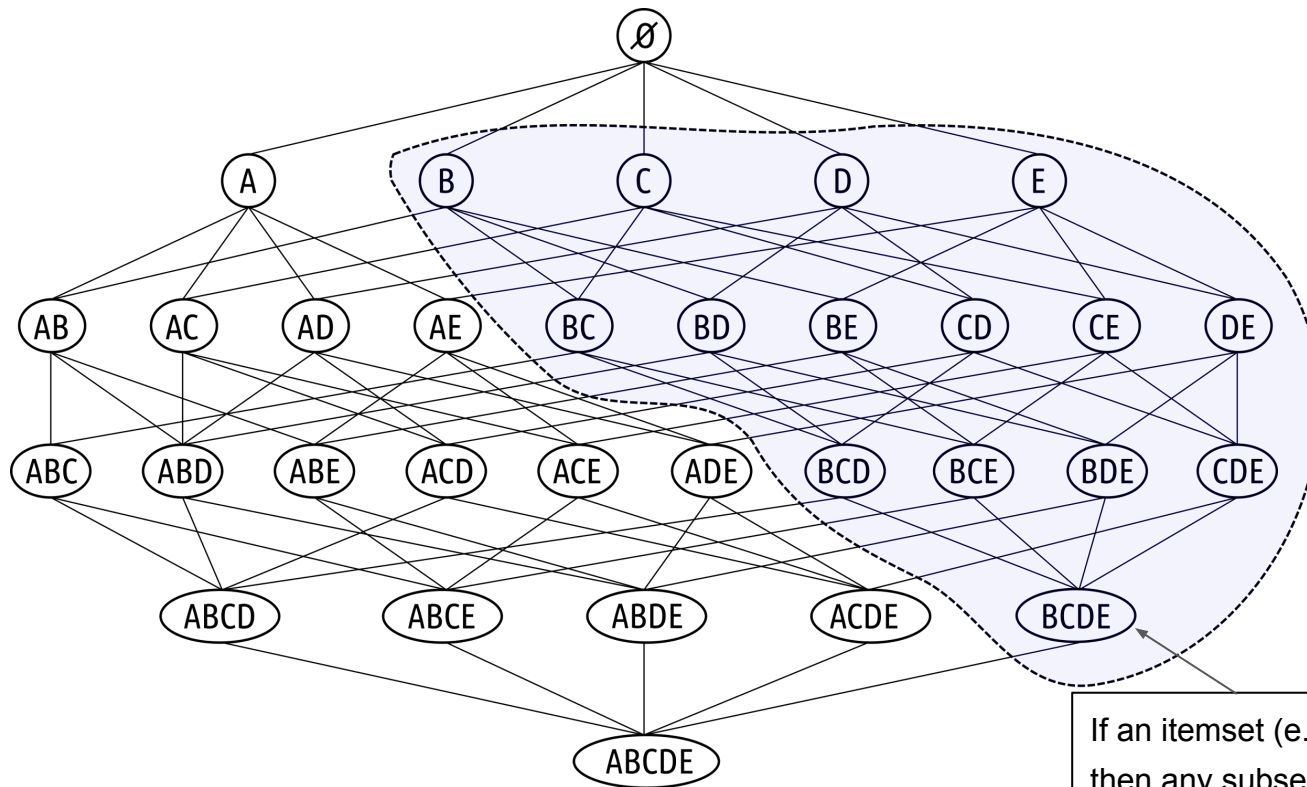
- **Observation 2:** If X and Y are itemsets and $X \subseteq Y$, then

- $S(X) \geq S(Y)$
- If Y is frequent, then X is frequent
- If X is not frequent, then Y is not frequent

$$S(\{\text{bread, yogurt}\}) = 3/5$$

$$S(\{\text{bread, yogurt, milk}\}) = 2/5$$

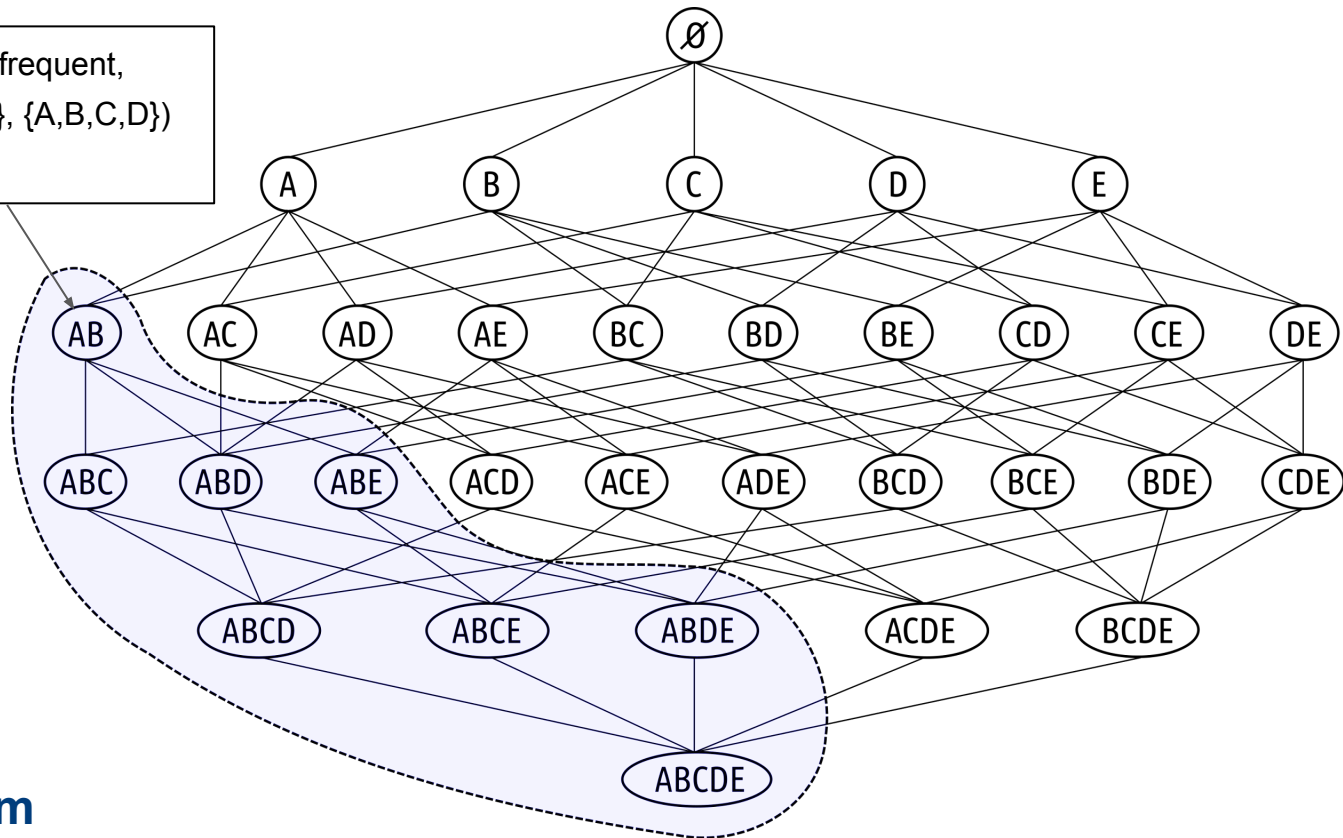
Apriori Principle (Anti-Monotonicity Principle)



If an itemset (e.g., $\{B, C, D, E\}$) is frequent, then any subset (e.g., $\{B, D, E\}$, $\{CE\}$) is also frequent

Apriori Principle (Anti-Monotonicity Principle)

If an itemset (e.g., {A,B}) is not frequent,
then any superset (e.g., {A,B,D}, {A,B,C,D})
is also not frequent



→ Apriori Algorithm

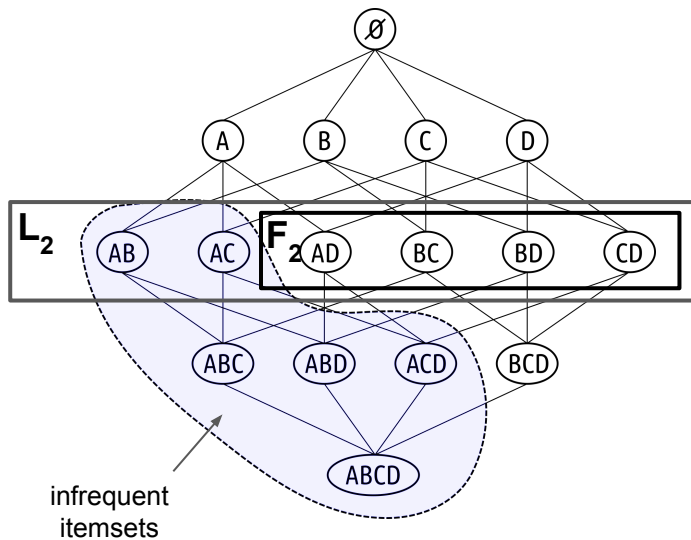
Apriori Algorithm

- Notations

- L_k — candidate k-itemsets
- F_k — frequent k-itemsets ($F_k \subseteq L_k$)

For k in $1..w$:

- **Generate** L_k from F_{k-1}
- **Prune** k-itemsets from L_k using F_{k-1}
- **Calculate** SC for remaining L_k itemsets
- **Filter** L_k itemsets with insufficient SC $\rightarrow F_k$
- If $|F_k| = 0$, stop



Quick Quiz

Which of the four steps is generally the **most expensive** one?

For k in $1..w$:

- **Generate** L_k from F_{k-1}
- **Prune** k -itemsets from L_k using F_{k-1}
- **Calculate** SC for remaining L_k itemsets
- **Filter** L_k itemsets with insufficient SC $\rightarrow F_k$
- If $|F_k| = 0$, stop

A

Generate

B

Prune

C

Calculate

D

Filter

Apriori Algorithm

minsup = 0.4 → minimum support count: 2



Generating

Itemset
{bread}
{cereal}
{cheese}
{eggs}
{milk}
{yogurt}

Calculating

Itemset	SC
{bread}	4
{cereal}	3
{cheese}	2
{eggs}	1
{milk}	4
{yogurt}	4

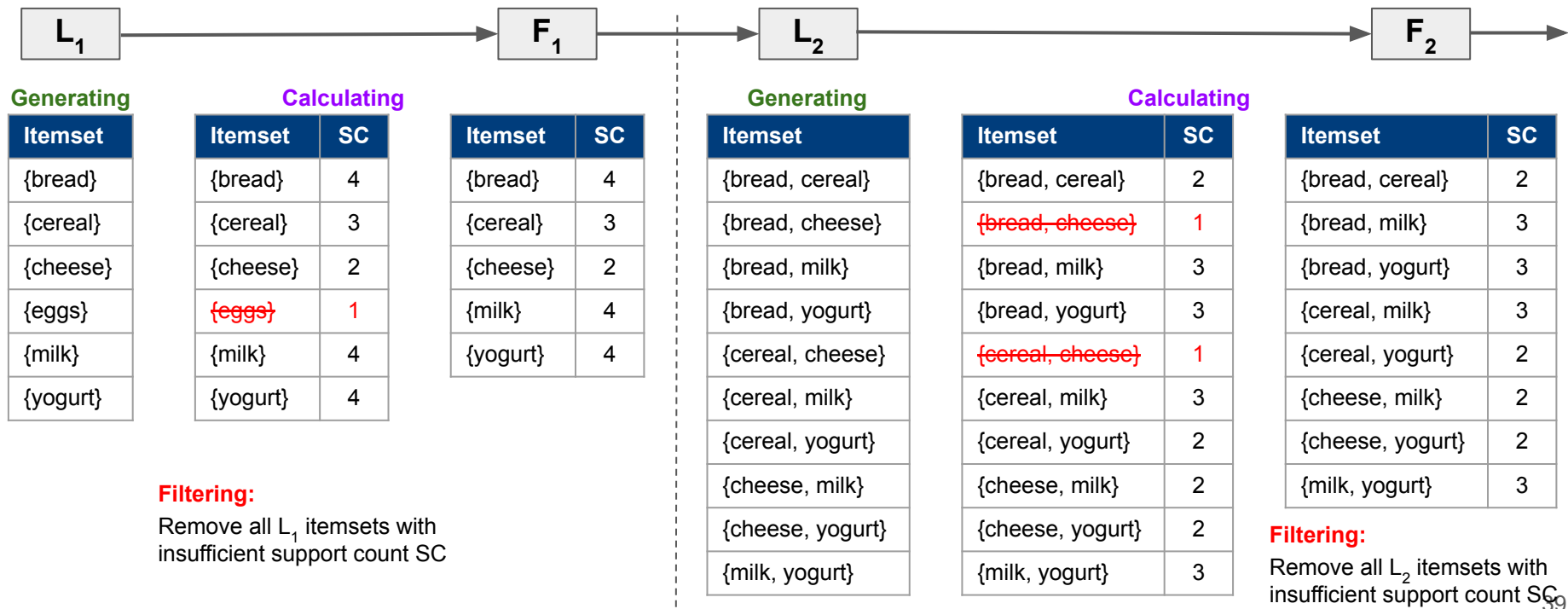
Itemset	SC
{bread}	4
{cereal}	3
{cheese}	2
{milk}	4
{yogurt}	4

Filtering:

Remove all L_1 itemsets with insufficient support count SC

Apriori Algorithm

minsup = 0.4 → minimum support count: 2



Apriori Algorithm

minsup = 0.4 → minimum support count: 2



Generating

k-1 itemsets

{bread, cheese}	✗
{bread, milk}	✓
{cheese, milk}	✓

Itemset
{bread, cereal, cheese}
{bread, cereal, milk}
{bread, cereal, yogurt}
{bread, cheese, milk}
{bread, cheese, yogurt}
{bread, milk, yogurt}
{cereal, cheese, milk}
{cereal, cheese, yogurt}
{cereal, milk, yogurt}
{cheese, milk yogurt}

Calculating

Itemset	SC
{bread, cereal, milk}	2
{bread, cereal, yogurt}	1
{bread, milk, yogurt}	2
{cereal, milk, yogurt}	2
{cheese, milk yogurt}	2

Itemset	SC
{bread, cereal, milk}	2
{bread, milk, yogurt}	2
{cereal, milk, yogurt}	2
{cheese, milk yogurt}	2

Pruning:

{bread, cheese} $\notin F_2$,

→ {bread, cheese, milk} $\notin F_3$

→ SC({bread, cheese, milk}) not needed!

Filtering:

Remove all L_3 itemsets with insufficient support count SC

Apriori Algorithm

minsup = 0.4 → minimum support count: 2



Generating

k-1 itemsets

{bread, cheese, milk}	✗
{bread, cheese, yogurt}	✗
{bread, milk, yogurt}	✓
{cheese, milk, yogurt}	✗

Itemset
{bread, cereal, cheese, milk}
{bread, cereal, milk, yogurt}
{bread, cheese, milk, yogurt}
{cereal, cheese, milk, yogurt}

Itemset	SC
---------	----

F4 is empty → done!

Pruning:

Only {bread, milk, yogurt} is in F₃

→ {bread, cheese, milk, yogurt} ∉ F₄

→ SC({bread, cheese, milk, yogurt}) not needed!

Apriori Algorithm

- Output: All frequent itemsets F_i with
 - $i \geq 2$ — cannot create rules from a single item
 - $|F_i| > 0$ — set of itemsets is not empty
- Implementation details
 - **Generating/Pruning** — How to get from F_{k-1} to L_k ?
 - **Calculating** — How to calculate SC for L_k itemsets efficiently?
(not covered here as this is done on the implementation level)

Itemset	SC	
{bread, cereal}	2	} F_2
{bread, milk}	3	
{bread, yogurt}	3	
{cereal, milk}	3	
{cereal, yogurt}	2	
{cheese, milk}	2	
{cheese, yogurt}	2	
{milk, yogurt}	3	
{bread, cereal, milk}	2	} F_3
{bread, milk, yogurt}	2	
{cereal, milk, yogurt}	2	
{cheese, milk yogurt}	2	

Generating/Pruning: $F_{k-1} \times F_1$ Method

F_2 : frequent 2-itemsets

Itemset
{bread, cereal}
{bread, milk}
{bread, yogurt}
{cereal, milk}
{cereal, yogurt}
{cheese, milk}
{cheese, yogurt}
{milk, yogurt}

Generating:

Merge frequent (k-1)-itemsets and frequent 1-itemsets to get all possible k-itemsets

L_3 : 3-itemsets

Itemset
{bread, cereal, cheese}
{bread, cereal, milk}
{bread, cereal, yogurt}
{bread, cheese, milk}
{bread, cheese, yogurt}
{bread, milk, yogurt}
{cereal, cheese, milk}
{cereal, cheese, yogurt}
{cereal, milk, yogurt}
{cheese, yogurt, milk}

Pruning:

Delete all k-itemsets with at least one containing (k-1)-itemset not in F_{k-1}

L_3 : 3-itemsets (pruned)

Itemset
{bread, cereal, milk}
{bread, cereal, yogurt}
{bread, milk, yogurt}
{cereal, milk, yogurt}
{cheese, milk, yogurt}

F_1 : frequent 1-itemsets

Itemset
{bread}
{cereal}
{cheese}
{milk}
{yogurt}

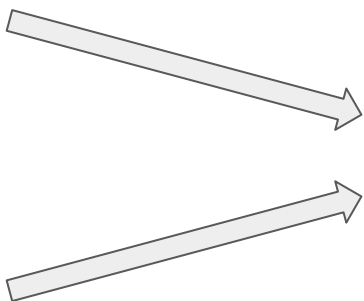
Generating/Pruning: $F_{k-1} \times F_{k-1}$ Method

F_2 : frequent 2-itemsets

Itemset
{bread, cereal}
{bread, milk}
{bread, yogurt}
{cereal, milk}
{cereal, yogurt}
{cheese, milk}
{cheese, yogurt}
{milk, yogurt}

Generating:

Merge frequent (k-1)-itemsets that overlap in (k-2) items to get all possible k itemsets

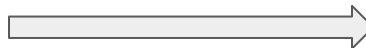


L_3 : 3-itemsets

Itemset
{bread, cereal, milk}
{bread, cheese, milk}
{bread, cereal, yogurt}
{bread, cheese, yogurt}
{bread, milk, yogurt}
{cereal, cheese, milk}
{cereal, cheese, yogurt}
{cereal, milk, yogurt}
{cheese, milk, yogurt}

Pruning:

Delete all k-itemsets with at least one containing (k-1)-itemset not in F_{k-1}



L_3 : 3-itemsets (pruned)

Itemset
{bread, cereal, milk}
{bread, cereal, yogurt}
{bread, milk, yogurt}
{cereal, milk, yogurt}
{cheese, milk, yogurt}

Calculating Support Counts

- Calculating SC for each candidate itemset in L_k
 - Requires **full scan** of database
 - For **each** transactions T, check for **each** itemset s if $s \in T$
 - If $s \in T$, **update** counter of s

→ This is the step we want to minimize!

L ₃ : 3-itemsets		L ₃ : 3-itemsets with SC values	
Itemset		Itemset	SC
{bread, cereal, milk}	Calculating →	{bread, cereal, milk}	2
{bread, cereal, yogurt}		{bread, cereal, yogurt}	1
{bread, milk, yogurt}		{bread, milk, yogurt}	2
{cereal, milk, yogurt}		{cereal, milk, yogurt}	2
{cheese, milk, yogurt}		{cheese, milk, yogurt}	2

Two-Part Algorithm for Mining Association Rules

- Part 1 — Frequent Itemset Generation

- General itemsets with support $\geq \text{minsup}$
- Apriori algorithm



- Part 2: — Association Rule Generation

- Generate rules from frequent itemsets through binary partitioning of itemsets
- Return rules with confidence $\geq \text{minconf}$

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese



minsup

Frequent itemsets:

{milk}, {cereal, milk}, {bread, milk}, ...



minconf

Association rules:

{cereal} \rightarrow {milk}

Rule Generation

- For each frequent itemset S ,
derive candidate rules $X \rightarrow Y$
 - A rule is a binary split of s , i.e., $Y=S-X$
- For each rule $X \rightarrow Y$
 - Calculate confidence $C(X \rightarrow Y)$
 - If confidence $\geq \text{minconf}$,
add rule to final result set

$2^{|S|}-2$ possible rules for each frequent itemset

$$C(X \rightarrow Y) = \frac{SC(X \cup Y)}{SC(X)}$$

Both values have been calculated
during Frequent Itemset Generation!

- No need to access database
- Fast

Apriori Principle (Anti-Monotonicity Principle)

- Given itemset S and two derived rules $X_1 \rightarrow Y_1$, $X_2 \rightarrow Y_2$ with $X_1 \cup Y_1 = X_2 \cup Y_2 = S$

$$C(X_1 \rightarrow Y_1) = \frac{S(X_1 \cup Y_1)}{S(X_1)} \quad C(X_2 \rightarrow Y_2) = \frac{S(X_2 \cup Y_2)}{S(X_2)}$$

$$\begin{aligned} X_1 \subseteq X_2 &\Rightarrow S(X_1) \geq S(X_2) \\ &\Rightarrow C(X_1 \rightarrow Y_1) \leq C(X_2 \rightarrow Y_2) \end{aligned}$$

- Example: If $\{A,B,C\} \rightarrow \{D\}$ has low confidence, so have:
 - $\{A\} \rightarrow \{B,C,D\}$, $\{B\} \rightarrow \{A,C,D\}$, $\{C\} \rightarrow \{A,B,D\}$, $\{A,B\} \rightarrow \{C,D\}$, $\{A,C\} \rightarrow \{B,D\}$, $\{B,C\} \rightarrow \{A,D\}$

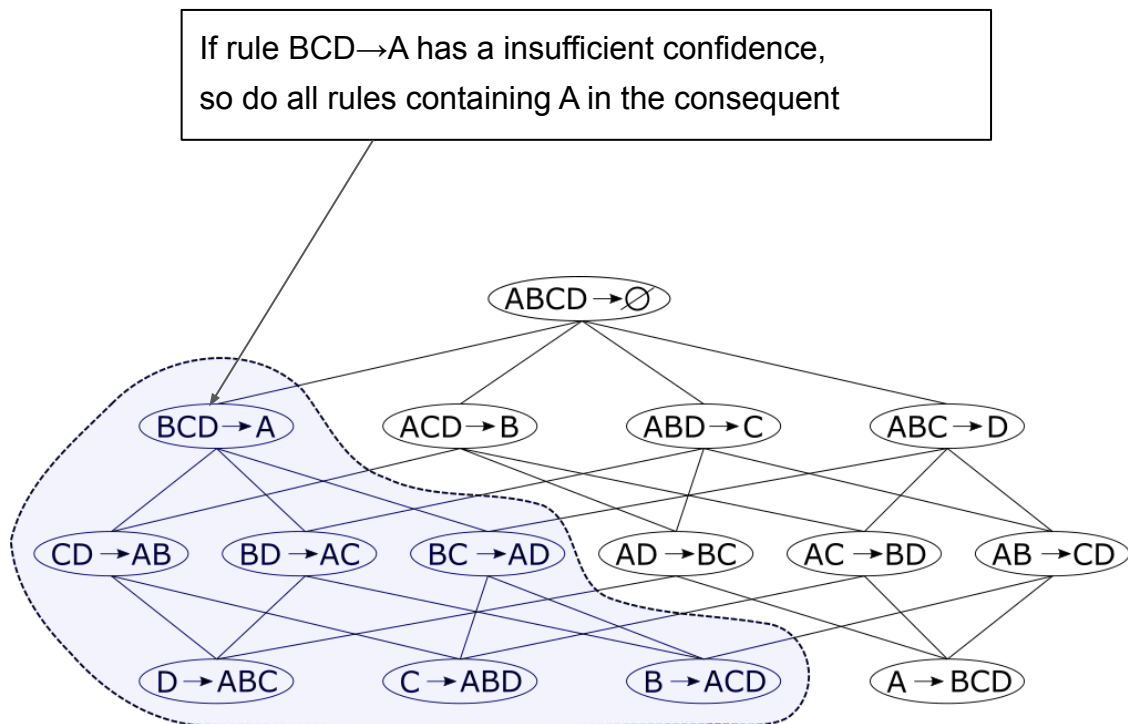
Apriori Principle (Anti-Monotonicity Principle)

- Rule lattice

- Node: association rule
- Edge: containment relationship w.r.t. antecedent/consequent

- $2^{|S|}-2$ rules

- $|S|=4 \rightarrow 14$ rules



Rule Generation Algorithm (for a single itemset S)

$YS = \{ \{s\} \mid s \in S \}$

$YS = \{ \{A\}, \{B\}, \{C\}, \{D\} \}$

repeat:

$YS_valid = \text{evaluate}(S, YS, minconf)$

$YS = \text{generate}(YS_valid)$

until $|YS| = 0$

evaluate $(S, YS, minconf)$:

$YS_obsolete \leftarrow \{\}$

for each Y **in** YS :

$X = S - Y$

if $C(X \rightarrow Y) \geq minconf$:

output $(X \rightarrow Y)$ as a valid rule

else:

$YS_obsolete \leftarrow YS_obsolete \cup Y$

return $YS - YS_obsolete$

$Y = \{A\}$	$Y = \{B\}$	$Y = \{C\}$	$Y = \{D\}$
$\{BCD\} \rightarrow \{A\}$	$\{ACD\} \rightarrow \{B\}$	$\{ABD\} \rightarrow \{C\}$	$\{ABC\} \rightarrow \{D\}$

$\{ \{B\}, \{C\}, \{D\} \}$

Rule Generation Algorithm (for a single itemset S)

$YS = \{ \{s\} \mid s \in S \}$

repeat:

$YS_valid = \text{evaluate}(S, YS, minconf)$

$YS = \text{generate}(YS_valid)$

until $|YS| = 0$

evaluate($S, YS, minconf$):

$YS_obsolete \leftarrow \{\}$

for each Y **in** YS :

$X = S - Y$

if $C(X \rightarrow Y) \geq minconf$:

output $(X \rightarrow Y)$ as a valid rule

else:

$YS_obsolete \leftarrow YS_obsolete \cup Y$

return $YS - YS_obsolete$

$YS_valid = \{ \{B\}, \{C\}, \{D\} \}$
$YS = \{ \{B, C\}, \{B, D\}, \{C, D\} \}$

$Y = \{B, C\}$ $\{AD\} \rightarrow \{BC\}$	$Y = \{B, D\}$ $\{AC\} \rightarrow \{BD\}$	$Y = \{C, D\}$ $\{AB\} \rightarrow \{CD\}$
---	---	---

$\{ \{B, C\}, \{B, D\}, \{C, D\} \}$

Rule Generation Algorithm (for a single itemset S)

$YS = \{ \{s\} \mid s \in S \}$

repeat:

$YS_valid = \text{evaluate}(S, YS, \text{minconf})$

$YS = \text{generate}(YS_valid)$

until $|YS| = 0$

evaluate($S, YS, \text{minconf}$) :

$YS_obsolete \leftarrow \{\}$

for each Y **in** YS :

$X = S - Y$

if $C(X \rightarrow Y) \geq \text{minconf}$:

output $(X \rightarrow Y)$ as a valid rule

else:

$YS_obsolete \leftarrow YS_obsolete \cup Y$

return $YS - YS_obsolete$

$YS_valid = \{ \{B, C\}, \{B, D\}, \{C, D\} \}$
$YS = \{ \{B, C, D\} \}$

$Y = \{B, C, D\}$
$\{A\} \rightarrow \{B, C, D\}$

$\{ \{B, C, D\} \}$

Rule Generation Algorithm (for a single itemset S)

$YS = \{ \{s\} \mid s \in S \}$

repeat:

$YS_valid = \text{evaluate}(S, YS, minconf)$

$YS = \text{generate}(YS_valid)$

until $|YS| = 0$

evaluate($S, YS, minconf$) :

$YS_obsolete \leftarrow \{\}$

for each Y **in** YS :

$X = S - Y$

if $C(X \rightarrow Y) \geq minconf$:

output $(X \rightarrow Y)$ as a valid rule

else:

$YS_obsolete \leftarrow YS_obsolete \cup Y$

return $YS - YS_obsolete$

$YS_valid = \{ \{B, C, D\} \}$
$YS = \{\}$



Done!

Two-Part Algorithm for Mining Association Rules

- Part 1 — Frequent Itemset Generation

- General itemsets with support $\geq \text{minsup}$
- Apriori algorithm



- Part 2: — Association Rule Generation

- Generate rules from frequent itemsets through binary partitioning of itemsets
- Return rules with confidence $\geq \text{minconf}$



TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese



minsup

Frequent itemsets:

{milk}, {cereal, milk}, {bread, milk}, ...



minconf

Association rules:

{cereal} \rightarrow {milk}

Definitions — Lift

- **Lift** of an association rule $X \rightarrow Y$
 - Probability of Y given X while controlling for support of Y (i.e., popularity of Y)

$$L(X \rightarrow Y) = \frac{S(X \rightarrow Y)}{S(X)S(Y)} = \frac{S(X \cup Y)}{S(X)S(Y)}$$

TID	Items
1	bread, yogurt
2	bread, cereal, milk, eggs
3	yogurt, milk, cereal, cheese
4	bread, cereal, yogurt, milk
5	bread, yogurt, milk, cheese

$$L(\{cereal\} \rightarrow \{bread\}) = \frac{S(\{cereal, bread\})}{S(\{cereal\})S(\{bread\})} = \frac{0.4}{0.6 \cdot 0.8} = 0.833$$

Lift — Interpretation

$$L(\{cereal\} \rightarrow \{bread\}) = \frac{S(\{cereal, bread\})}{S(\{cereal\})S(\{bread\})} = \frac{0.4}{0.6 \cdot 0.8} = 0.833$$

- Probability of {bread}

$$S(\{bread\}) = 0.8$$

- Probability of {bread} given {cereal}

$$C(\{cereal\} \rightarrow \{bread\}) = 0.66$$

Presence of cereal **reduces** probability of bread!

$$\Rightarrow L(\{cereal\} \rightarrow \{bread\}) \leq 1.0$$

- Usage of lift (and other metrics for association rules)

- Further filtering and ranking of association rules
- Finding "substitution" items

Note: Lift is not part of Apriori algorithm since anti-monotonicity principle does not hold here

Quick "Quiz"

Which association rule would you expect to have the **smallest lift**?

A

$\{cereal\} \rightarrow \{milk\}$

B

$\{coke\} \rightarrow \{pepsi\}$

C

$\{milo\} \rightarrow \{nutella\}$

D

$\{banana\} \rightarrow \{grapes\}$

Outline

- Association Rule Mining
 - Overview
 - Applications
- Definitions
- Algorithms
 - Brute-Force
 - A-Priori
- Discussion & Summary

Discussion

- **Alternative metric to decide whether a rule is interesting** (beyond confidence and lift)
 - Conviction, all-confidence, collective strength, leverage
- **Additional useful information to consider, for example:**
 - Attributes of items (e.g., quantity and price of products)
 - Sequence of items (e.g., order when products have be added to the cart)
 - Categories of items (e.g., "milk" and "yogurt" are both "dairy" products)
 - User information (e.g., associating multiple transactions to the same user)
- **Reminder: Rules indicate correlations / co-occurrences, NOT causality!**

Summary

- Pattern of interest: Association Rule $X \rightarrow Y$
 - Predicting the occurrence of some items Y based on occurrence of other items X
 - Applicable to a wider range of task for transactional data
 - Various metrics that define whether a rule is useful (e.g., support, confidence, lift)
- Practical algorithm to handle complexity
 - Decoupling calculations of support and confidence
 - Apriori algorithm for Frequent Itemset Generation and Association Rule Generation

Solutions to Quick Quizzes

- Slide 22: C
- Slide 37: C
- Slide 57: B