NUS | Computing

National University of Singapore

CS5228: Knowledge Discovery and Data Mining

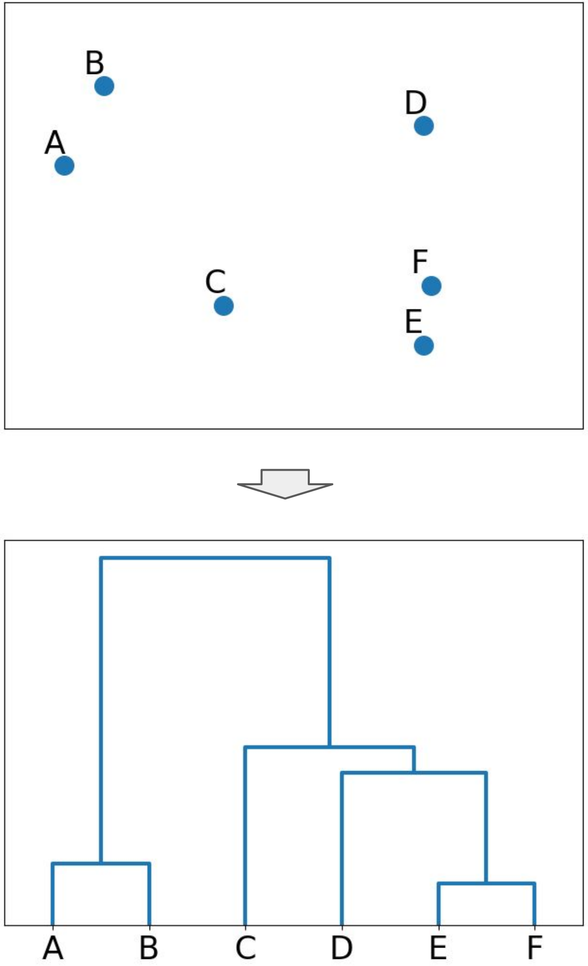
Lecture 4 — Association Rule Mining

Course Logistics — Update

Recap — Hierarchical Clustering

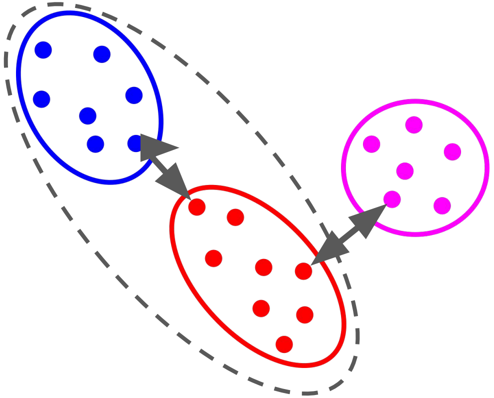
- AGNES (AGglomerative NESting)
  - Start with *N* clusters, one for each data point
  - Iteratively merge nearest clusters into one
  - Stop if all data points are in one cluster

- Core questions: How to calculate distances between clusters?

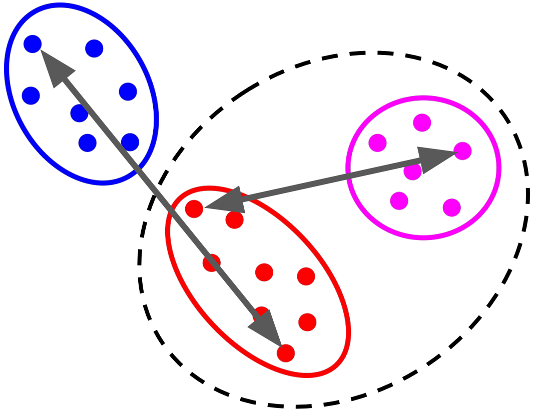


Recap — Linkage Methods

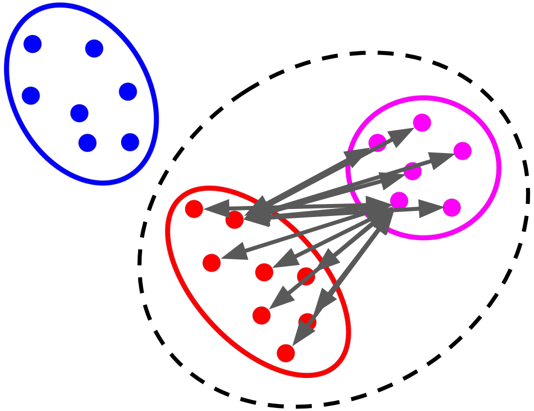
Single Linkage



Complete Linkage



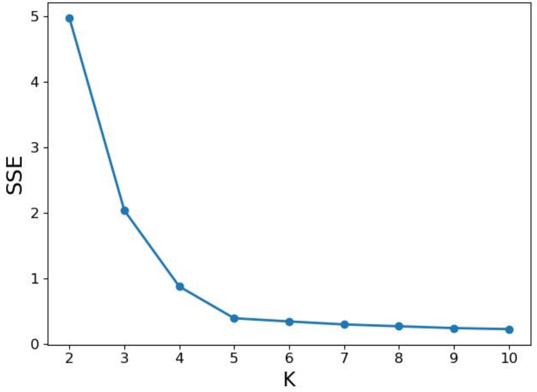
Average Linkage

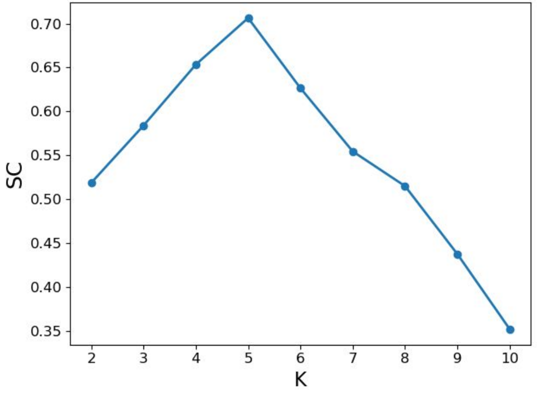


Recap — Cluster Evaluation

- 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 → consequent

$\{item_2, item_3\} \rightarrow \{item_5\}$

$\{item_1\} \rightarrow \{item_3\}$

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 by {a, b} als 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
...	

{anosmia, fatigue}→{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}→{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., {cereal}→{milk} does not mean that customers always by milk when buying cereal
  - each possible combination (e.g., {yogurt, bread}→{milk}) is potential association rule
- Given d unique items ➜ $3^d - 2^{d+1} + 1$ rules
  - $d = 6$ ➜ 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({bread, yogurt, milk}) = 2
- Support S
  - Fraction of transactions containing an itemset
  - e.g., S({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→Y, where X and Y are itemsets
  - e.g., {yogurt, milk}→{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→Y

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

#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
...	

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

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

Definitions — Confidence

- **Confidence** of an association rule X→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 ➜ Y	Low Support	High Support
Low Confidence	• The items in (X ∪ Y) do not frequently appear together • Even if the items in X appear together, they do so often without the items in Y	• The items in (X ∪ Y) frequently appear together • If the items in X appear together, they often do so without the items in Y
High Confidence	• The items in (X ∪ Y) do not frequently appear together • If the items in X appear together, they often do so with the items in Y	• The items in (X ∪ Y) frequently appear together • If the items in X appear together, they do so often with the items in Y

Quick Quiz

Outline

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

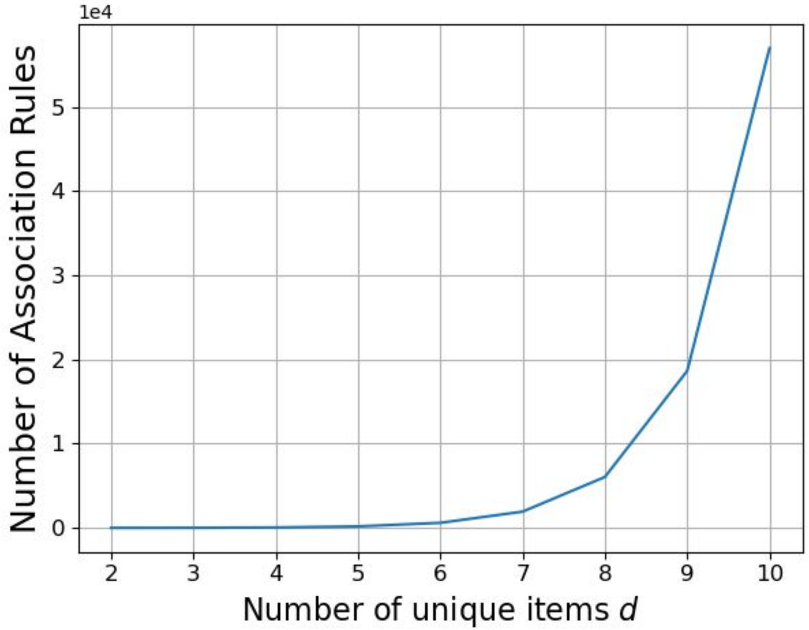
Brute Force Approach — Algorithm

- Given a set of transactions,
  find all association rules X→Y with
  - Support S(X→Y) ≥ *minsup*
  - Confidence C(X→Y) ≥ *minconf*

- Brute force algorithm
  - List all possible association rules X→Y
  - Calculate support S(X→Y) and confidence C(X→Y) for each rule
  - Drop rules with S(X→Y) < *minsup* and C(X→Y) < *minconf*

Brute Force Approach — Computation Complexity

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



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
  - $N = 5, w = 4$ ➜ ≤ 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.

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 (rows), w (items)

True number of different rules: 154

Decoupling Support and Confidence

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

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

- **Observation 1**
  - A rule X→Y has only sufficient support if X∪Y is a frequent itemset
  - No need to calculate confidence of rules where X∪Y is not a frequent item set

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

Two-Part Algorithm for Mining Association Rules

- Part 1 — Frequent Itemset Generation
  - Generate itemsets with support ≥ *minsup*
  - "Only" 2^d-1 possible itemsets to check
- Part 2: — Association Rule Generation
  - Generate rules from frequent itemsets through binary partitioning of itemsets
  - Return rules with confidence ≥ *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*

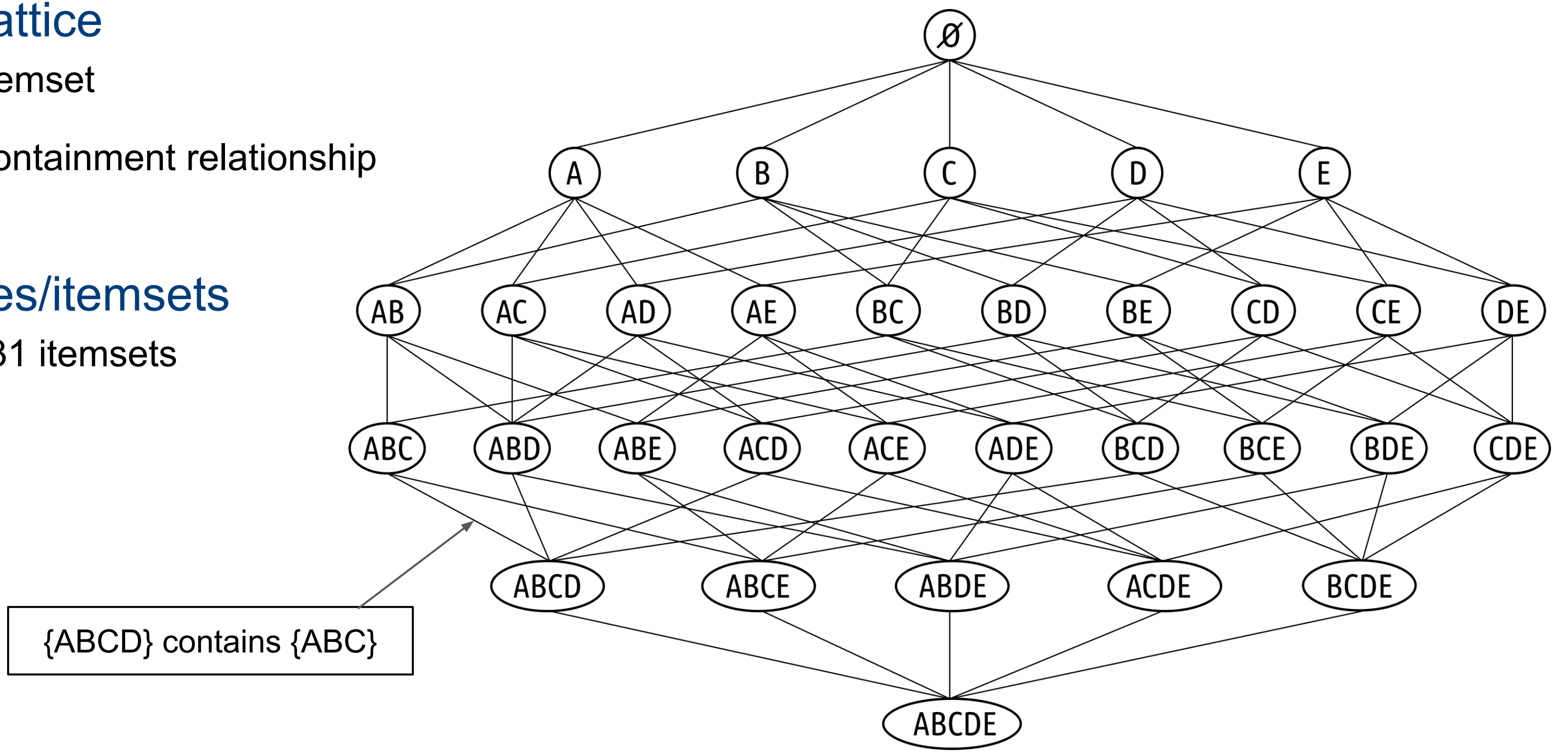
Frequent itemsets:
{milk}, {cereal, milk}, {bread, milk}, ...

⇩ *minconf*

Association rules:
{cereal} → {milk}

Frequent Itemset Generation

- Itemset lattice
  - Node: itemset
  - Edge: containment relationship
- 2^d-1 nodes/itemsets
  - d=5 ➜ 31 itemsets



Frequent Itemset Generation — Brute Force Algorithm

$$support\_counts \leftarrow dict(\{\})$$

$$\textbf{for each } \text{transaction } t \textbf{ in } \text{database:}$$

$$\quad \textbf{for } k \textbf{ in } 1..(t.length):$$

$$\qquad k\_itemsets \leftarrow generate\_itemsets(t, k)$$

$$\qquad \textbf{for each } itemset \textbf{ in } k\_itemsets:$$

$$\qquad\quad support\_counts[itemset] \mathrel{+}= 1$$

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

N

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

w

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

$M = 2^w$

→ $O(N \cdot 2^w)$

reduce using sampling

reduce using Apriori algorithm

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
...	

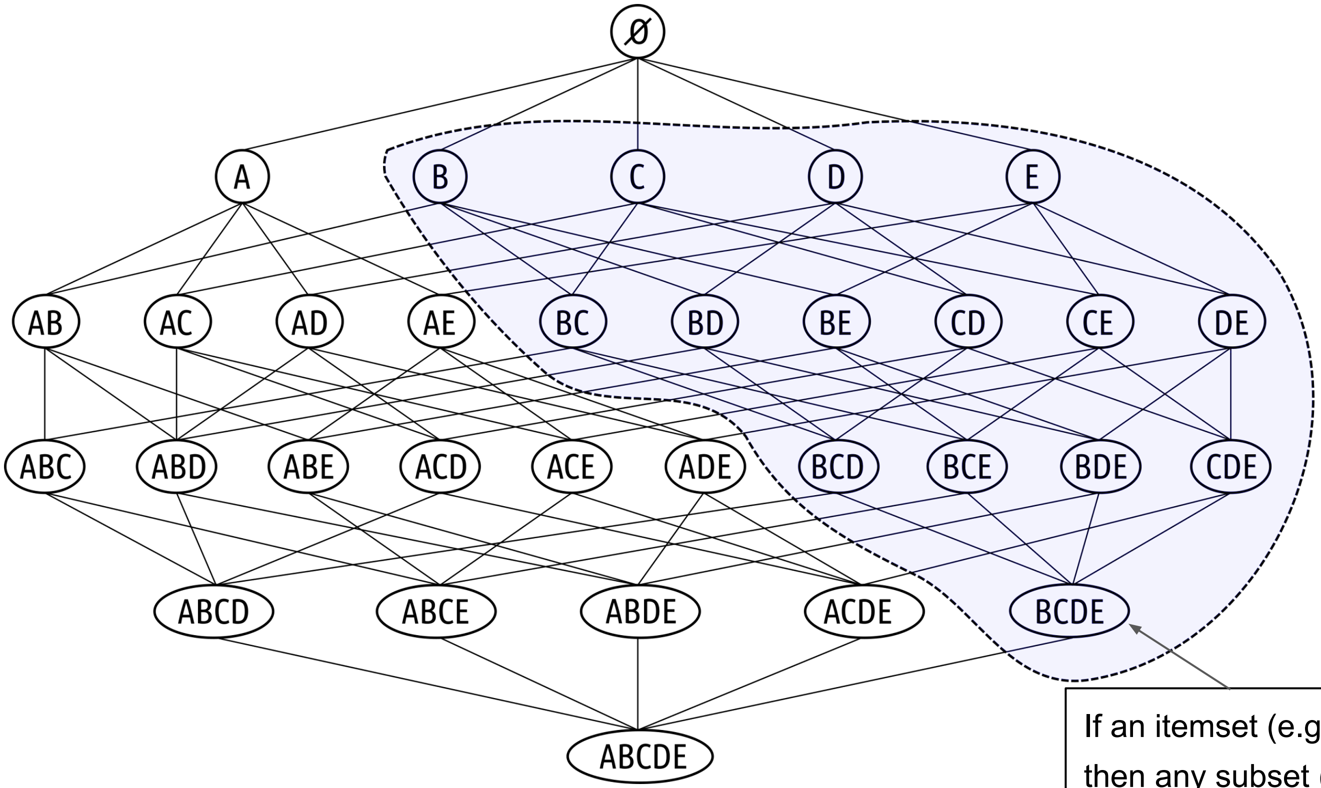
TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese
...	

- **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({bread, yogurt}) = 3/5

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

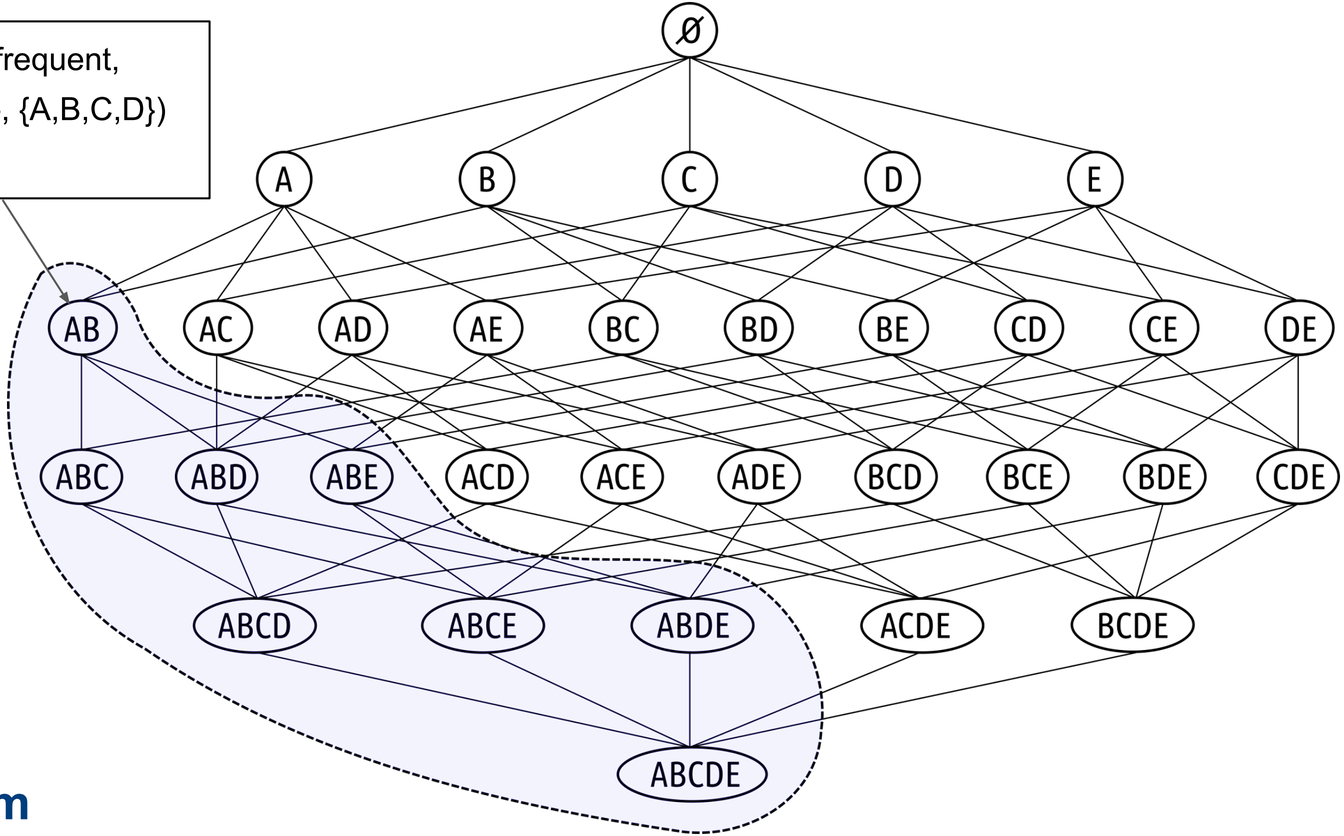
Apriori Principle (Anti-Monotonicity Principle)



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)

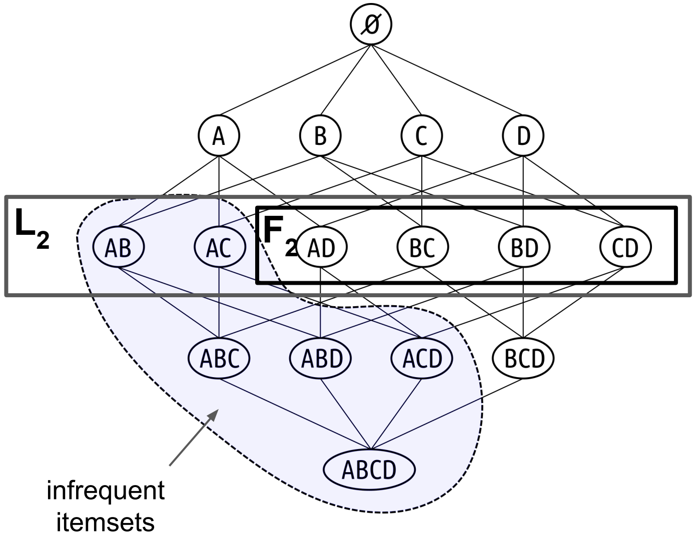
If an itemset (e.g., {A,B}) is not frequent, then any superset (e.g., {A,B,D}, {A,B,C,D}) is also not frequent



➜ 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 ➜ F_k
- If $|F_k| = 0$, stop

Quick Quiz

Apriori Algorithm

minsup = 0.4 ➜ minimum support count: 2

L_1 → F_1 →

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

$L_1 \rightarrow F_1 \rightarrow L_2 \rightarrow F_2 \rightarrow$

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

Generating

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

Calculating

Itemset	SC
{bread, cereal}	2
~~{bread, cheese}~~	1
{bread, milk}	3
{bread, yogurt}	3
~~{cereal, cheese}~~	1
{cereal, milk}	3
{cereal, yogurt}	2
{cheese, milk}	2
{cheese, yogurt}	2
{milk, yogurt}	3

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

Filtering:

Remove all L_2 itemsets with insufficient support count SC

Apriori Algorithm

minsup = 0.4 ➜ minimum support count: 2

$L_1 \to F_1 \to L_2 \to F_2 \to L_3 \to F_3$

k-1 itemsets

{bread, cheese}	✘
{bread, milk}	✔
{cheese, milk}	✔

Generating

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

L_1 ➜ F_1 ➜ L_2 ➜ F_2 ➜ L_3 ➜ F_3 ➜ L_4 ➜ F_4

k-1 itemsets

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

Generating

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 F_3

➜ {bread, cheese, milk, yogurt} $\notin F_4$

➜ 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}

F_1: frequent 1-itemsets

Itemset
{bread}
{cereal}
{cheese}
{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}

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∈T
  - If s∈T, **update** counter of s

➜ This is the step we want to minimize!

L_3: 3-itemsets

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

Calculating ⇒

L_3: 3-itemsets with SC values

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

Two-Part Algorithm for Mining Association Rules

- Part 1 — Frequent Itemset Generation ✔
  - General itemsets with support ≥ *minsup*
  - Apriori algorithm
- Part 2: — Association Rule Generation
  - Generate rules from frequent itemsets through binary partitioning of itemsets
  - Return rules with confidence ≥ *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}

Rule Generation

- For each frequent itemset S, derive candidate rules X→Y
  - A rule is a binary split of s, i.e., Y=S-X

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

- For each rule X→Y
  - Calculate confidence C(X→Y)
  - If confidence ≥ *minconf*, add rule to final result set

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

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 $X_1 \cup Y_1 = X_2 \cup Y_2 = S$

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

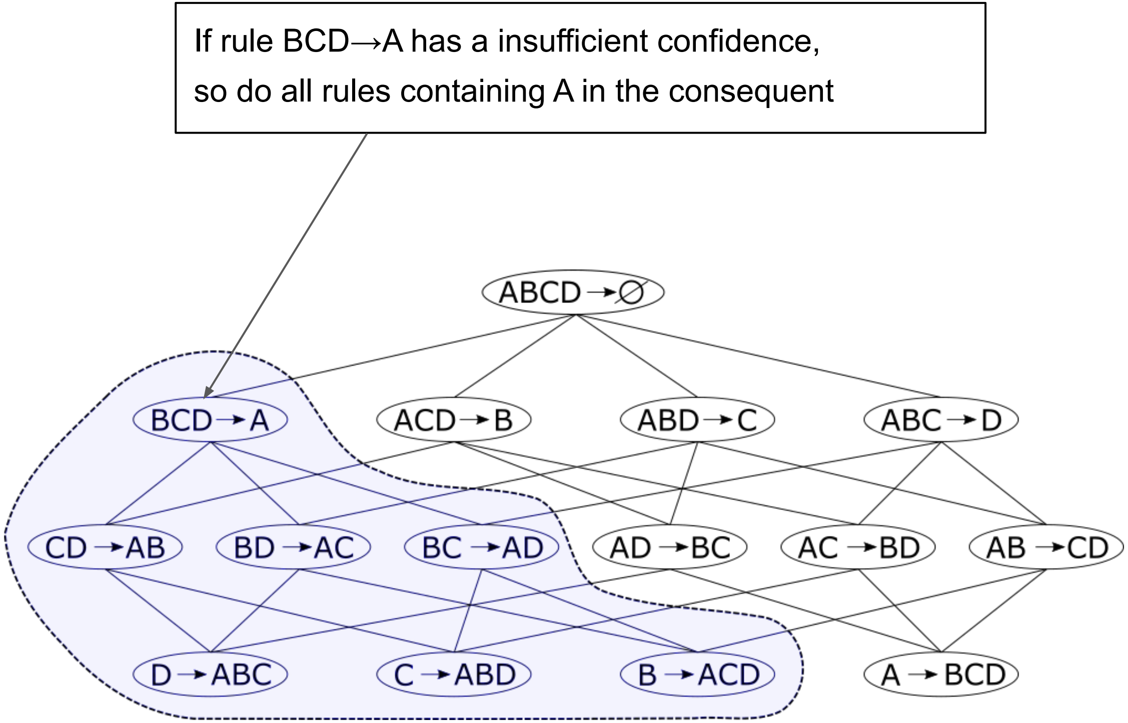
$$X_1 \subseteq X_2 \Rightarrow S(X_1) \geq S(X_2)$$

$$\Rightarrow C(X_1 \rightarrow Y_1) \leq C(X_2 \rightarrow Y_2)$$

- Example: If {A,B,C}→{D} has low confidence, so have:
  - {A}→{B,C,D}, {B}→{A,C,D}, {C}→{A,B,D}, {A,B}→{C,D}, {A,C}→{B,D}, {B,C}→{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 ➜ 14 rules



Rule Generation Algorithm (for a single itemset S)

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

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

repeat :

  YS_valid = **evaluate**(S, YS, $minconf$)

  YS = **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 $YS - YS\_obsolete$

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

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

Rule Generation Algorithm (for a single itemset S)

$YS = \{\ \{s\} \mid s \in S\ \}$
repeat:
  $YS\_valid =$ **evaluate**$(S,\ YS,\ minconf)$
  $YS =$ **generate**$(\ YS\_valid)$
until $|\, YS\, | = 0$

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

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 $YS - YS\_obsolete$

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

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

Rule Generation Algorithm (for a single itemset S)

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

$\textbf{repeat}:$

$\quad YS\_valid = \textbf{evaluate}(S,\ YS,\ minconf)$

$\quad YS = \textbf{generate}(\ YS\_valid)$

$\textbf{until}\ |\ YS\ | = 0$

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

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

$\quad YS\_obsolete \leftarrow \{\}$

$\quad \textbf{for each}\ Y\ \textbf{in}\ YS:$

$\quad\quad X = S - Y$

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

$\quad\quad\quad \text{output}\ (X \rightarrow Y)\ \text{as a valid rule}$

$\quad\quad \textbf{else}:$

$\quad\quad\quad YS\_obsolete \leftarrow YS\_obsolete \cup Y$

$\quad \textbf{return}\ YS - YS\_obsolete$

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

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

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

Rule Generation Algorithm (for a single itemset S)

$YS = \{\ \{s\} \mid s \in S\ \}$
repeat :
 $YS\_valid = \textbf{evaluate}(S,\ YS,\ minconf)$
 $YS = \textbf{generate}(\ YS\_valid)$
until $|\,YS\,| = 0$

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

Done!

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 $YS - YS\_obsolete$

Two-Part Algorithm for Mining Association Rules

- Part 1 — Frequent Itemset Generation
  - General itemsets with support ≥ *minsup*
  - Apriori algorithm

✔

- Part 2: — Association Rule Generation
  - Generate rules from frequent itemsets through binary partitioning of itemsets
  - Return rules with confidence ≥ *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}

Definitions — Lift

- **Lift** of an association rule X→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"

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→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

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∈T
  - If s∈T, **update** counter of s

➜ This can be slow! How to speed it up?

L_3: 3-itemsets

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

Calculating ⇨

L_3: 3-itemsets with SC values

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

Calculating SC — Prerequisite

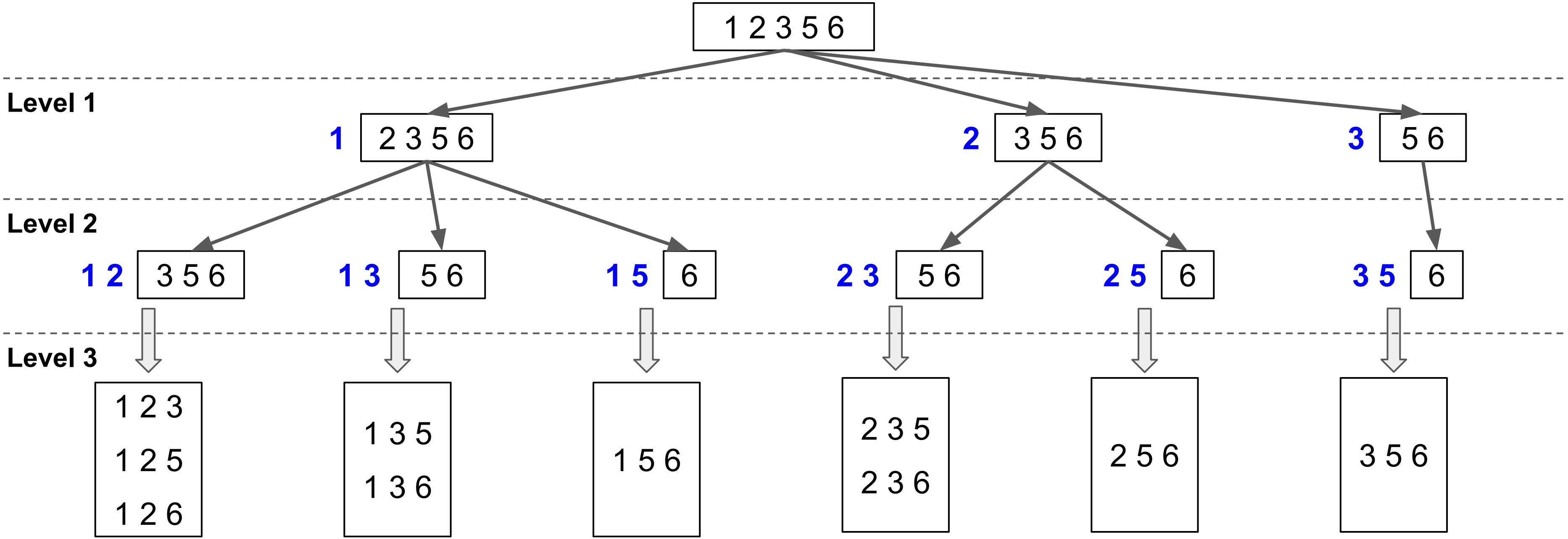
- Enumerate all items
  - bread➜1, cereal➜2, cheese➜3, eggs➜4, milk➜5, yogurt➜6
  - Sort items within each transaction

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	1, 6
2	1, 2, 4, 5
3	2, 3, 5, 6
4	1, 2, 5, 6
5	1, 3, 5, 6

Calculating SC: Generate all K-Itemsets from Transaction

- Example: Generate all 3-itemsets in transaction T={1,2,3,5,6}
  - **Subset enumeration:** Systematic way for enumerating all 3-itemsets in T



Calculating SC — Candidate Itemset Counter

- Assume the following set L_3 of 15 candidate k-itemset

  {1 2 4}, {1 2 5}, {1 3 6}, {1 4 5}, {1 5 9},
  {2 3 4}, {3 4 5}, {3 5 6}, {3 5 7}, {3 6 7},
  {3 6 8}, {4 5 7}, {4 5 8}, {5 6 7}, {6 8 9}

- Implement itemset counters as a lookup table, e.g.,
  - Dictionary (Python)
  - HashMap (Java)

Itemset	Count
{1 2 4}	0
{1 2 5}	5
{1 3 6}	0
{1 4 5}	3
{1 5 9}	10
{2 3 4}	2
{3 4 5}	0
{3 5 6}	14
{3 5 7}	0
{3 6 7}	0
{3 6 8}	0
{4 5 7}	6
{4 5 8}	8
{5 6 7}	0
{6 8 9}	2

Example: Itemset counters after processing several transactions

Calculating SC — Candidate Itemset Counter

- We already know that T={1,2,3,5,6} yields the following ten 3-itemsets

➜ Increase counters for all matching itemsets

{1 2 3}
{1 2 5}
{1 2 6}
{1 3 5}
{1 3 6}
{1 5 6}
{2 3 5}
{2 3 6}
{2 5 6}
{3 5 6}

Itemset	Count
{1 2 4}	0
{1 2 5}	**6**
{1 3 6}	**1**
{1 4 5}	3
{1 5 9}	10
{2 3 4}	2
{3 4 5}	0
{3 5 6}	**15**
{3 5 7}	0
{3 6 7}	0
{3 6 8}	0
{4 5 7}	6
{4 5 8}	8
{5 6 7}	0
{6 8 9}	2