POLITECNICO DI MILANO

Association Rule Basics

Data Mining and Text Mining (UIC 583 @ Politecnico di Milano)

Prof. Pier Luca Lanzi (Spring 2009)

Lecture outline

- What is association rule mining?
- Frequent itemsets, support, and confidence
- Mining association rules
- The “Apriori” algorithm
- Rule generation

Example of market-basket transactions

- Bread
- Peanuts
- Milk
- Fruit
- Jam

- Bread
- Jam
- Soda
- Chips
- Milk
- Fruit

- Steak
- Jam
- Soda
- Chips
- Bread

- Jam
- Soda
- Peanuts
- Milk
- Fruit

- Jam
- Soda
- Chips
- Milk
- Bread

- Fruit
- Soda
- Chips
- Milk

- Fruit
- Soda
- Peanuts
- Milk

- Fruit
- Peanuts
- Cheese
- Yogurt

What is association mining?

- Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories
- Applications
  - Basket data analysis
  - Cross-marketing
  - Catalog design
  - …

What is association rule mining?

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

Examples

$\{bread\} \Rightarrow \{milk\}$

$\{soda\} \Rightarrow \{chips\}$

$\{bread\} \Rightarrow \{jam\}$

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items, e.g., {milk, bread, jam}
  - k-itemset, an itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - σ({Milk, Bread}) = 3
    σ({Soda, Chips}) = 4
- **Support**
  - Fraction of transactions that contain an itemset
  - s({Milk, Bread}) = 3/8
    s({Soda, Chips}) = 4/8
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a minsup threshold

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

What is an association rule?

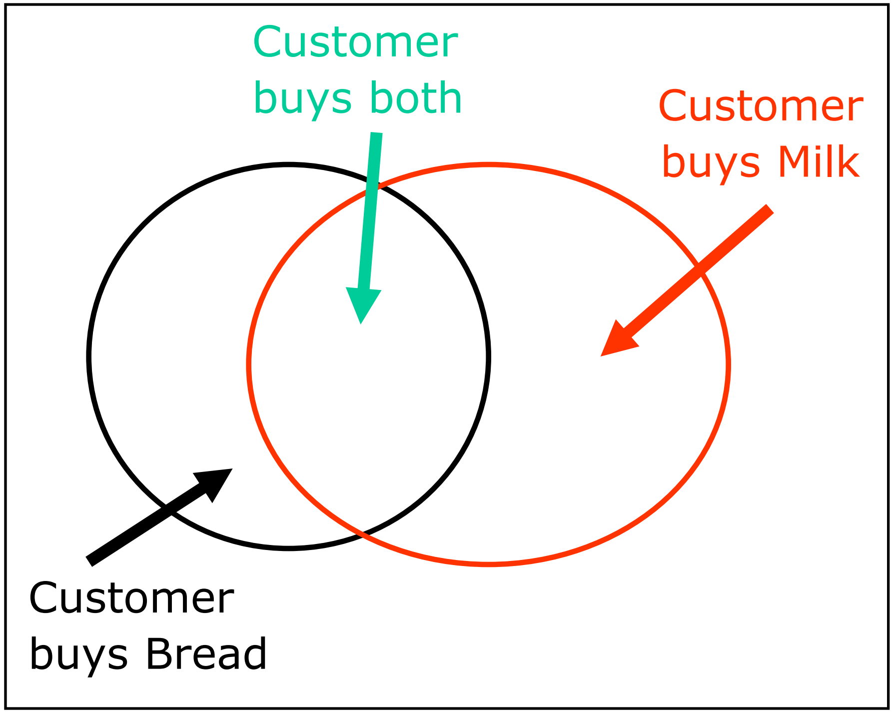
- Implication of the form X $\Rightarrow$ Y, where X and Y are itemsets
- Example, {bread} $\Rightarrow$ {milk}
- Rule Evaluation Metrics, Suppor & Confidence
- Support (s)
  - Fraction of transactions that contain both X and Y

$$s = \frac{\sigma(\{\text{Bread}, \text{Milk}\})}{\#\text{ of transactions}} = 0.38$$

- Confidence (c)
  - Measures how often items in Y appear in transactions that contain X

$$c = \frac{\sigma(\{\text{Bread}, \text{Milk}\})}{\sigma(\{\text{Bread}\})} = 0.75$$

Support and Confidence



What is the goal?

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ minsup threshold
  - confidence ≥ minconf threshold

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the minsup and minconf thresholds

- Brute-force approach is computationally prohibitive!

{Bread, Jam} $\Rightarrow$ {Milk}	s=0.4 c=0.75
{Milk, Jam} $\Rightarrow$ {Bread}	s=0.4 c=0.75
{Bread} $\Rightarrow$ {Milk, Jam}	s=0.4 c=0.75
{Jam} $\Rightarrow$ {Bread, Milk}	s=0.4 c=0.6
{Milk} $\Rightarrow$ {Bread, Jam}	s=0.4 c=0.5

- All the above rules are binary partitions of the same itemset:

  {Milk, Bread, Jam}

- Rules originating from the same itemset have identical support but can have different confidence

- We can decouple the support and confidence requirements!

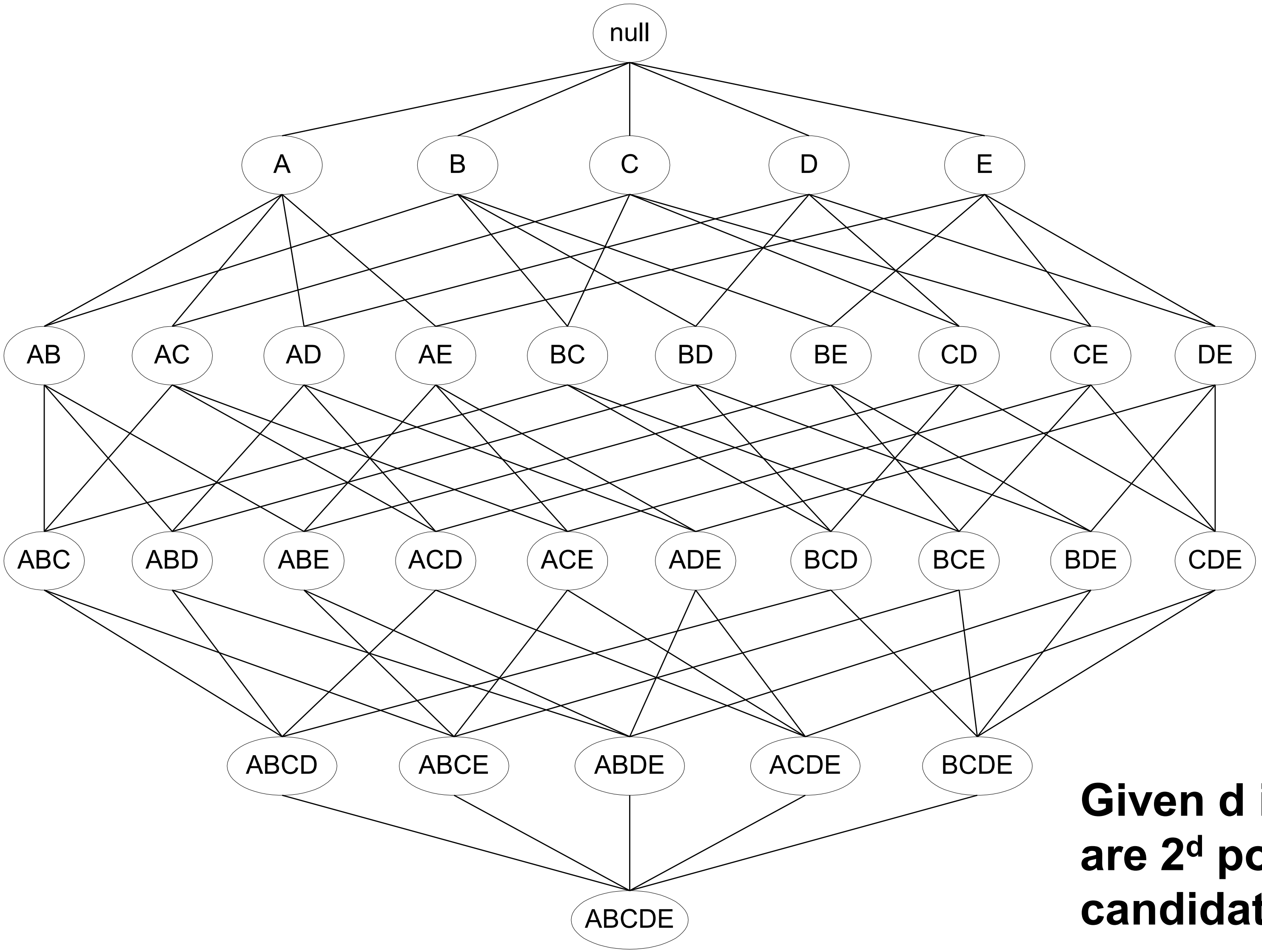
Mining Association Rules: Two Step Approach

- Frequent Itemset Generation
  - Generate all itemsets whose support $\geq$ minsup

- Rule Generation
  - Generate high confidence rules from frequent itemset
  - Each rule is a binary partitioning of a frequent itemset

- Frequent itemset generation is computationally expensive

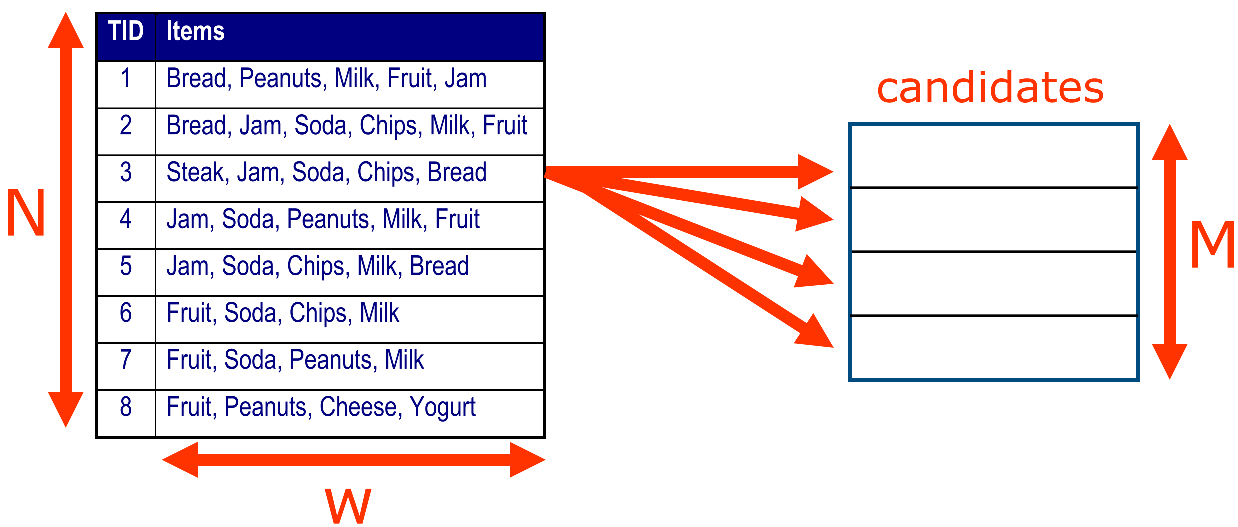
Frequent Itemset Generation



Given d items, there are 2^d possible candidate itemsets

Frequent Itemset Generation

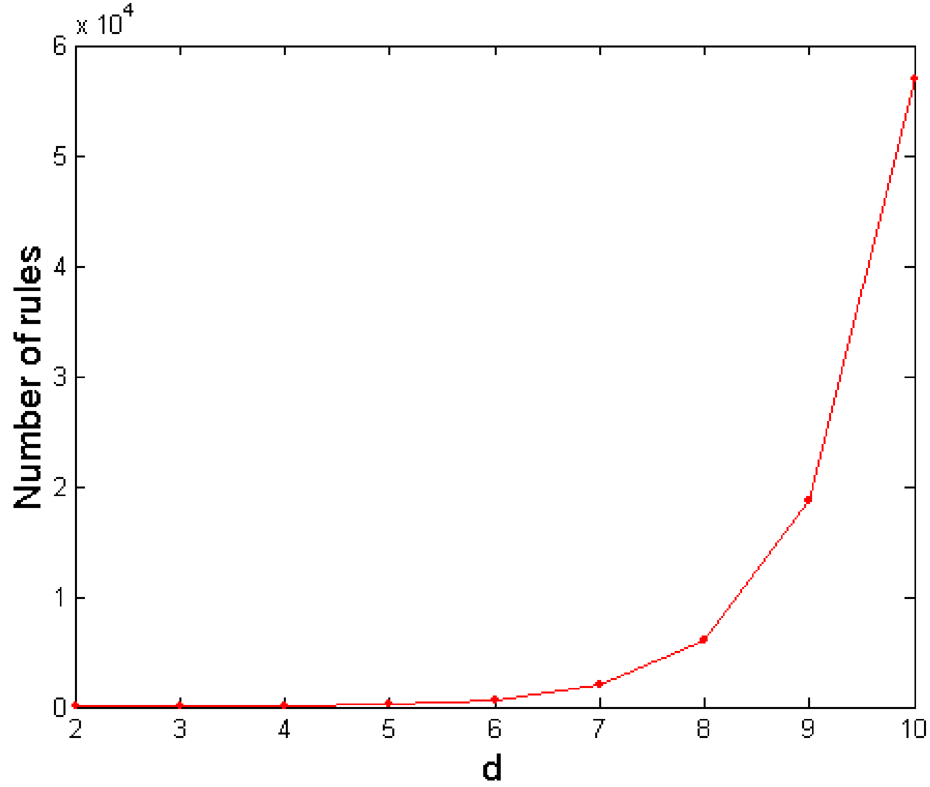
- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database



TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

- Match each transaction against every candidate
- Complexity ~ O(NMw) => Expensive since $M = 2^d$

Computational Complexity



- Given d unique items:
  - Total number of itemsets = 2^d
  - Total number of possible association rules:

$$\sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right]$$
$$= 3^d - 2^{d+1} + 1$$

  - For d=6, there are 602 rules

Frequent Itemset Generation Strategies

- Reduce the number of candidates (M)
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

Reducing the Number of Candidates

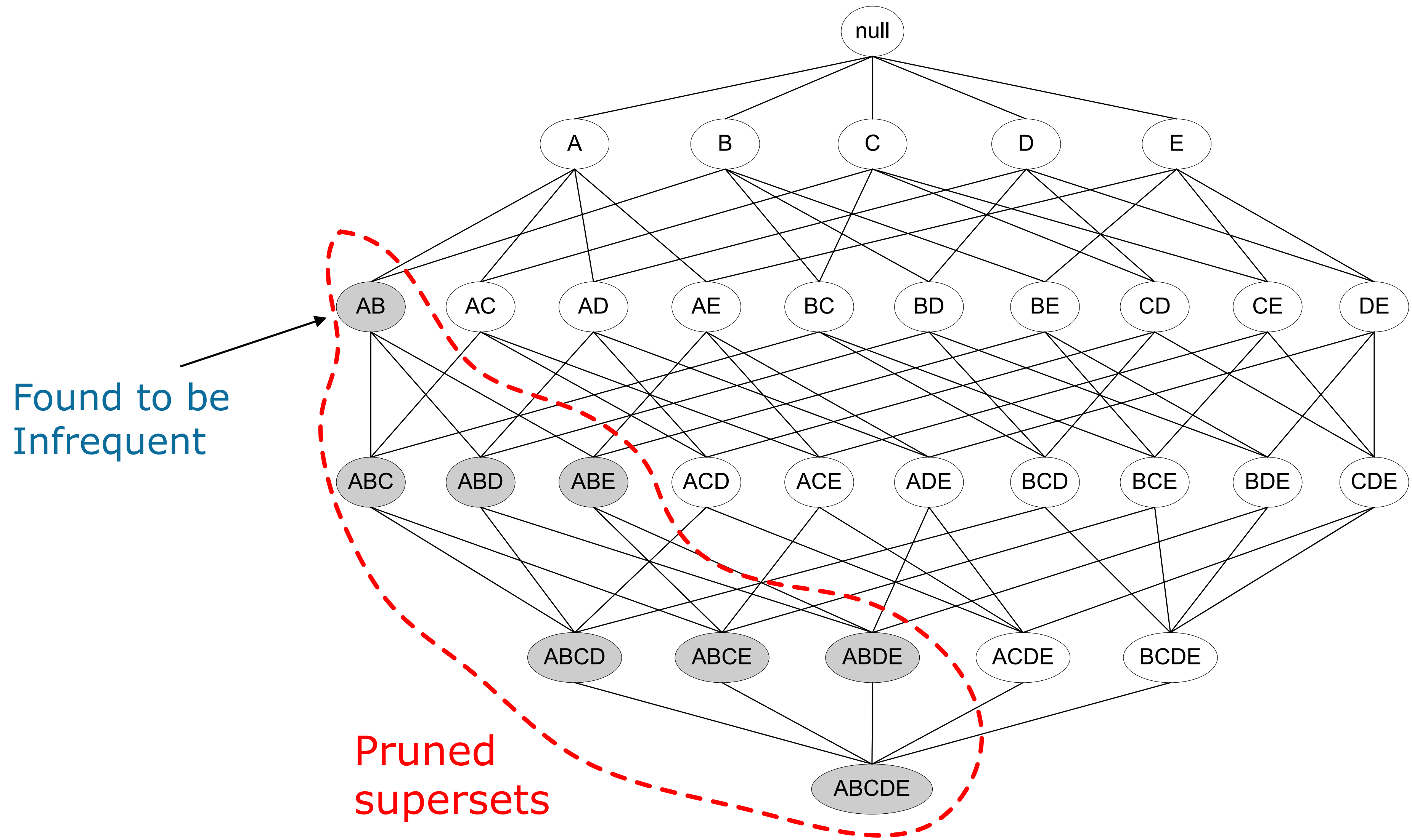
- Apriori principle
  - If an itemset is frequent, then all of its subsets must also be frequent
- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

Support of an itemset never exceeds the support of its subsets

This is known as the anti-monotone property of support

Illustrating Apriori Principle



How does the Apriori principle work?

Items (1-itemsets)

Item	Count
Bread	4
Peanuts	4
Milk	6
Fruit	6
Jam	5
Soda	6
Chips	4
Steak	1
Cheese	1
Yogurt	1

Minimum Support = 4

2-itemsets

2-Itemset	Count
Bread, Jam	4
Peanuts, Fruit	4
Milk, Fruit	5
Milk, Jam	4
Milk, Soda	5
Fruit, Soda	4
Jam, Soda	4
Soda, Chips	4

3-itemsets

3-Itemset	Count
Milk, Fruit, Soda	4

Apriori Algorithm

- Let k=1
- Generate frequent itemsets of length 1
- Repeat until no new frequent itemsets are identified
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Prune candidate itemsets containing subsets of length k that are infrequent
  - Count the support of each candidate by scanning the DB
  - Eliminate candidates that are infrequent, leaving only those that are frequent

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

L_1 = {frequent items};
for (k = 1; L_k !=∅; k++) **do begin**
  C_{k+1} = candidates generated from L_k;
  for each transaction t in database do
    increment the count of all candidates in C_{k+1}
    that are contained in t
  L_{k+1} = candidates in C_{k+1} with min_support
  end
return $\cup_k L_k$;

- Join Step
  - C_k is generated by joining L_{k-1} with itself
- Prune Step
  - Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset

Efficient Implementation of Apriori in SQL

- Hard to get good performance out of pure SQL (SQL-92) based approaches alone
- Make use of object-relational extensions like UDFs, BLOBs, Table functions etc.
- Get orders of magnitude improvement
- S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. In SIGMOD'98

How to Improve Apriori's Efficiency

- Hash-based itemset counting: A k-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- Transaction reduction: A transaction that does not contain any frequent k-itemset is useless in subsequent scans
- Partitioning: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
- Sampling: mining on a subset of given data, lower support threshold + a method to determine the completeness
- Dynamic itemset counting: add new candidate itemsets only when all of their subsets are estimated to be frequent

Rule Generation

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
- If {A,B,C,D} is a frequent itemset, candidate rules:

  $ABC \rightarrow D$, $ABD \rightarrow C$, $ACD \rightarrow B$, $BCD \rightarrow A$, $A \rightarrow BCD$, $B \rightarrow ACD$, $C \rightarrow ABD$, $D \rightarrow ABC$, $AB \rightarrow CD$, $AC \rightarrow BD$, $AD \rightarrow BC$, $BC \rightarrow AD$, $BD \rightarrow AC$, $CD \rightarrow AB$

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \varnothing$ and $\varnothing \rightarrow L$)

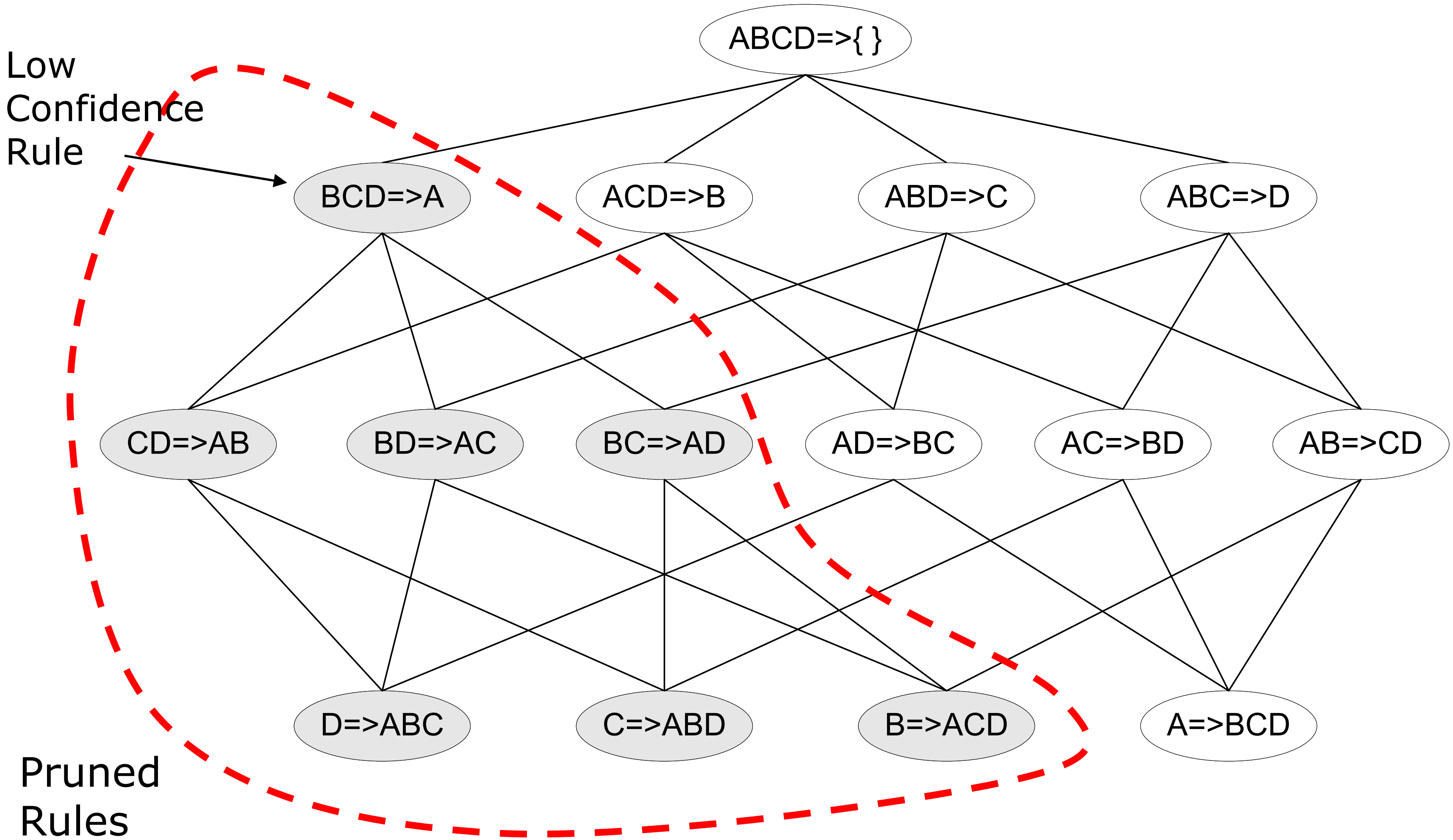
How to efficiently generate rules from frequent itemsets?

- Confidence does not have an anti-monotone property
- $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$
- But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., $L = \{A,B,C,D\}$:

$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

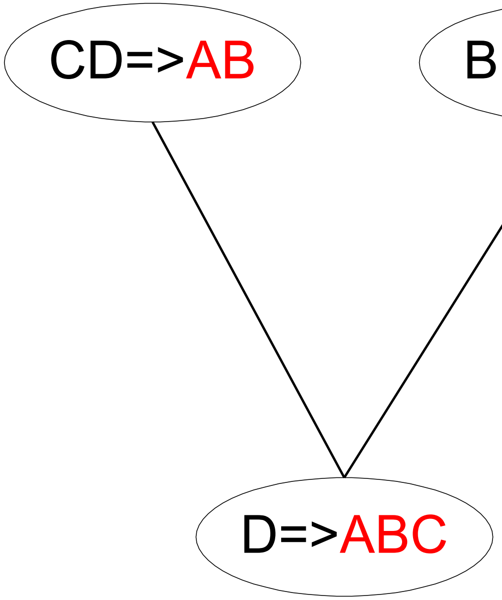
- Confidence is anti-monotone with respect to the number of items on the right hand side of the rule

Rule Generation for Apriori Algorithm



Rule Generation for Apriori Algorithm

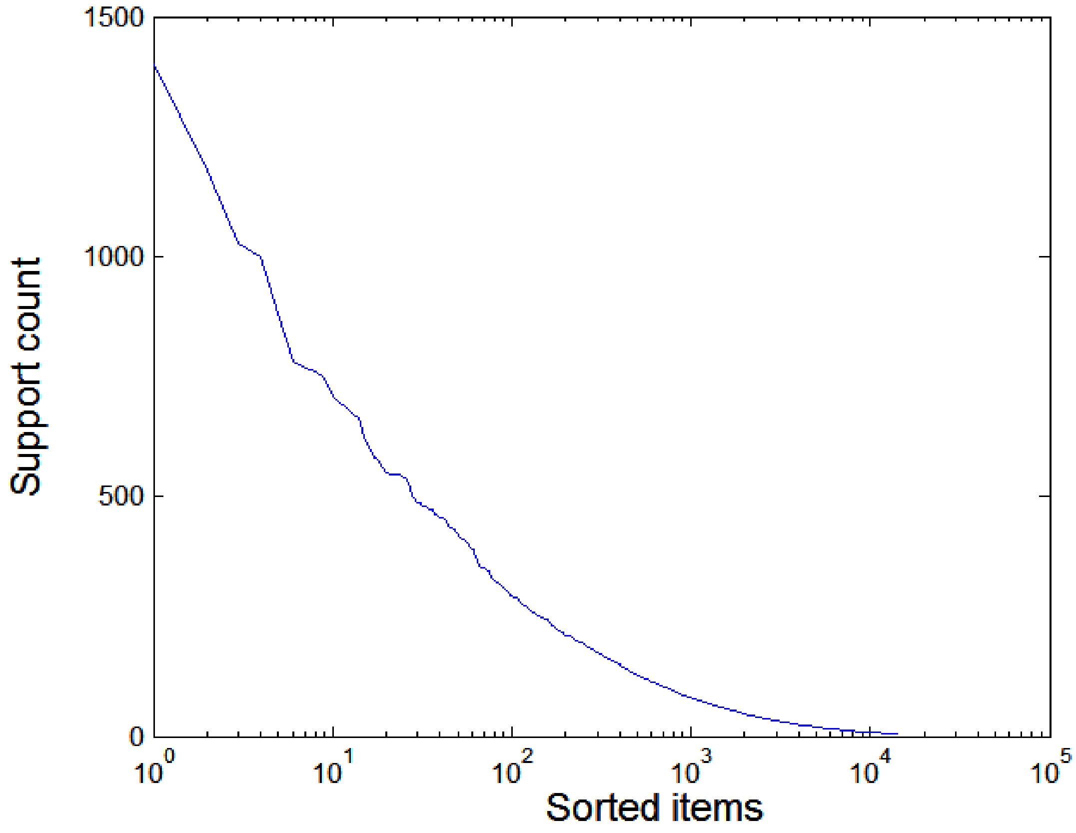
- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent
- join(CD=>AB,BD=>AC) would produce the candidate rule D => ABC
- Prune rule D=>ABC if its subset AD=>BC does not have high confidence



Effect of Support Distribution

- Many real data sets have skewed support distribution

Support distribution of a retail data set



How to set the appropriate minsup?

- If minsup is set too high, we could miss itemsets involving interesting rare items (e.g., expensive products)

- If minsup is set too low, it is computationally expensive and the number of itemsets is very large

- A single minimum support threshold may not be effective

Summary

- Association rules are implication of the form X $\Rightarrow$ Y, where X and Y are itemsets (e.g., {bread} $\Rightarrow$ {milk})
- An itemset is a collection of one or more items
- Support is the frequency of occurrence of an itemset
- The confidence of an association rule X $\Rightarrow$ Y measures how often items in Y appear in transactions that contain X
- Mining association rules find all the rules having
  - support $\geq$ minsup threshold
  - confidence $\geq$ minconf threshold
- Apriori exploits the anti-monotone property of support to reduce the number of candidate solutions