Machine Learning and Data Mining
(IT4242E)

Quang Nhat NGUYEN

quang.nguyennhat@hust.edu.vn

Hanoi University of Science and Technology
School of Information and Communication Technology

Academic year 2021-2022

The course’s content:

- Introduction
- Performance evaluation of the ML/DM system
- Regression problem
- Classification problem
- Clustering problem
- **Association rule mining problem**

Association rule mining – Introduction

- The problem of association rule mining:
  - 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

TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Beer, Eggs**
3	**Milk, Diaper, Beer, Coke**
4	**Bread, Milk, Diaper, Beer**
5	**Bread, Milk, Diaper, Coke**

Example of association rules:

{Diaper} $\rightarrow$ {Beer}

{Milk, Bread} $\rightarrow$ {Eggs,Coke}

{Beer, Bread} $\rightarrow$ {Milk}

Basic concepts (1)

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - *k*-itemset
    - An itemset that contains *k* items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g., σ({Milk, Bread,Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g., s({Milk, Bread, Diaper}) = 2/5
- **Frequent/large itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Basic concepts (2)

- **Association rule**
  - An implication expression of the form X $\rightarrow$ Y, where X and Y are itemsets
  - E.g., {Milk, Diaper} $\rightarrow$ {Beer}
- **Rule evaluation metrics**
  - **Support (s)**
    - Fraction of transactions that contain both X and Y
  - **Confidence (c)**
    - Measures how often items in Y appear in transactions that contain X

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

$$\{\text{Milk}, \text{Diaper}\} \rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Association rule mining task

- Given a set of transactions *T*, the goal of association rule mining is to find all rules having:
  - support ≥ *minsup* threshold, **<u>and</u>**
  - 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
- The brute-force approach has too high computational cost – not feasible for implementation in practice!

Mining association rules

TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Beer, Eggs**
3	**Milk, Diaper, Beer, Coke**
4	**Bread, Milk, Diaper, Beer**
5	**Bread, Milk, Diaper, Coke**

Các luật kết hợp:

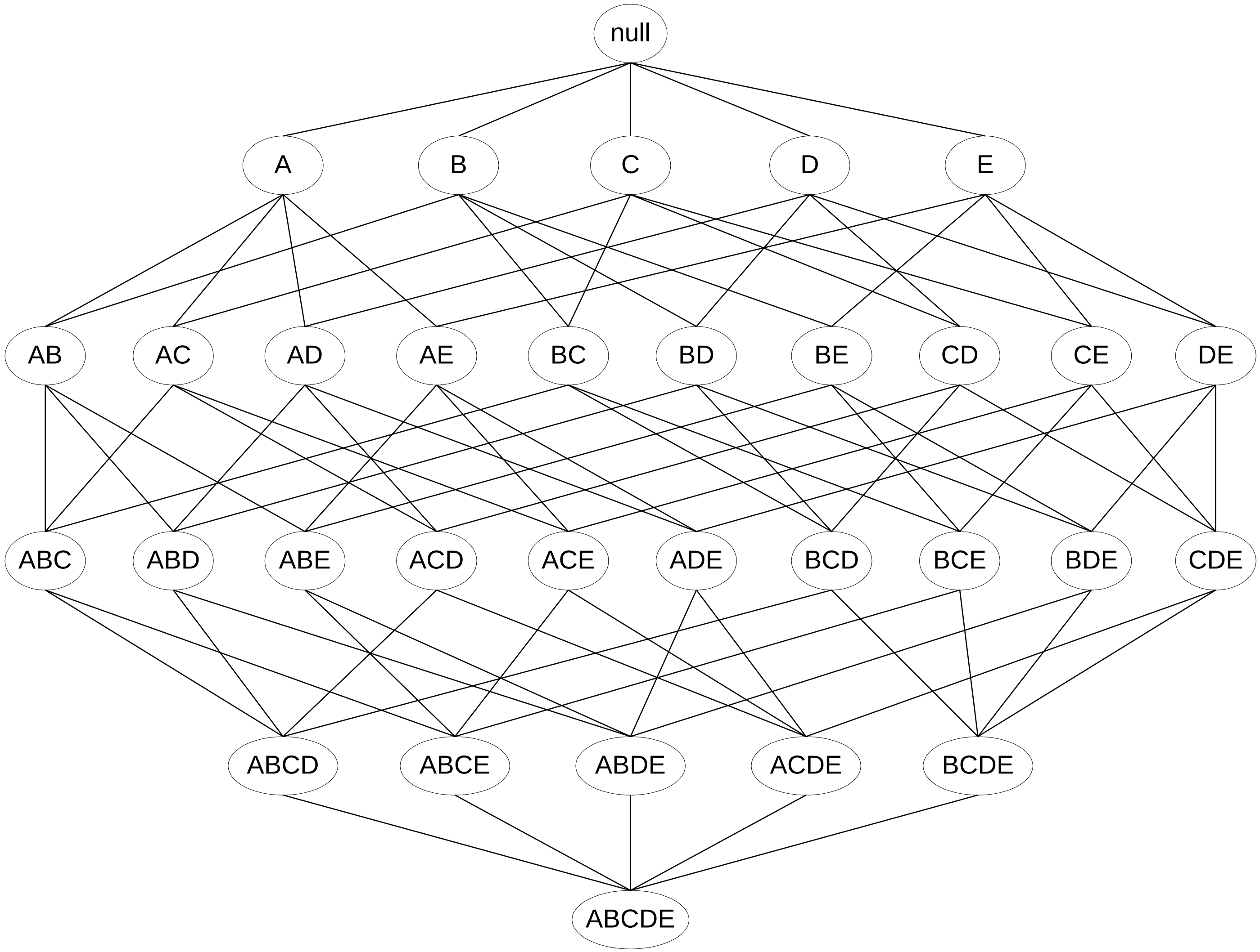
{Milk, Diaper} $\rightarrow$ {Beer} (s=0.4, c=0.67)
{Milk, Beer} $\rightarrow$ {Diaper} (s=0.4, c=1.0)
{Diaper, Beer} $\rightarrow$ {Milk} (s=0.4, c=0.67)
{Beer} $\rightarrow$ {Milk, Diaper} (s=0.4, c=0.67)
{Diaper} $\rightarrow$ {Milk, Beer} (s=0.4, c=0.5)
{Milk} $\rightarrow$ {Diaper, Beer} (s=0.4, c=0.5)

- All the above rules are binary partitions of the same itemset: {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

Mining association rules

- A process of association rule mining consists of two main phases (steps)
  - **Frequent/large itemset generation**
    - Generate all itemsets whose support ≥ minsup
  - **Rule generation**
    - From each frequent itemset, generate high confidence rules (their confidence ≥ *minconf*)
    - Each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

Frequent itemset generation



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

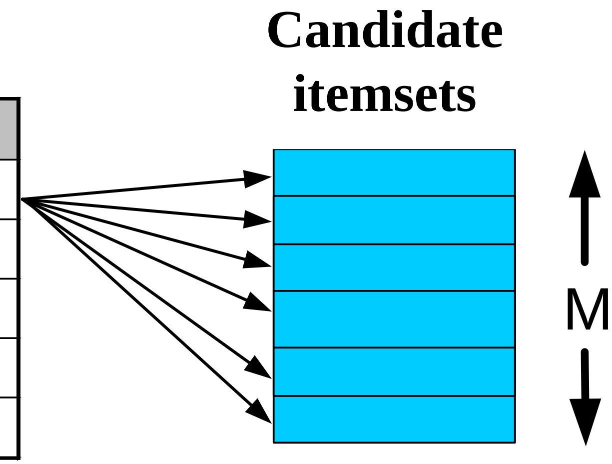
Frequent itemset generation

Transactions

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

N

w



- Brute-force approach
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database
  - Match each transaction against every candidate
  - Complexity ~ O(N.M.w)
    - Given $M = 2^d$, this cost is too high!

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 (i.e., the average number of items in an itemset - *w*) increases
- Reduce the **number of matchings/comparisons (NM)**
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

Reducing number of candidates

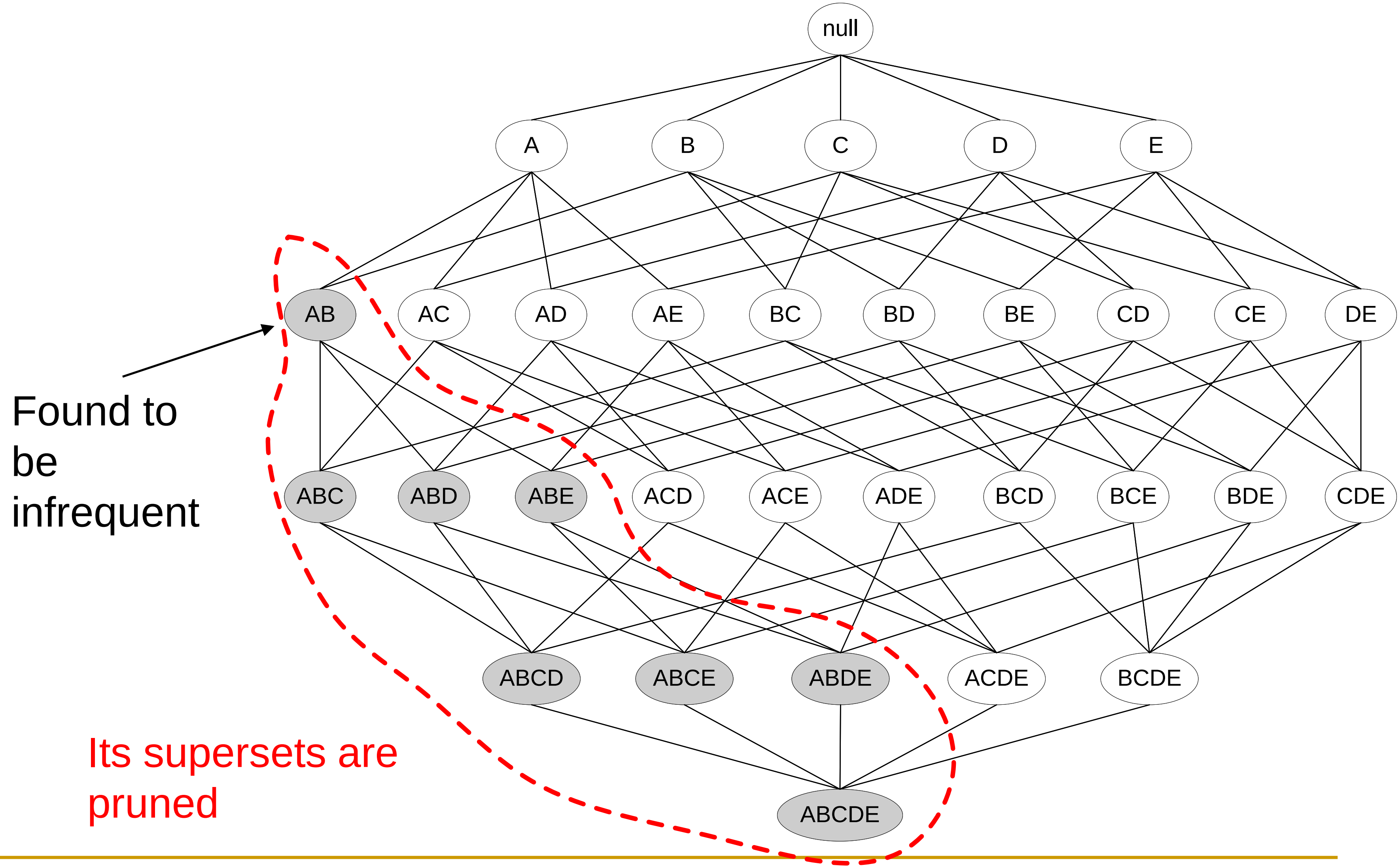
- **Apriori principle** – Prunning based on the support value
  - If an itemset is frequent, then all of *its subsets* must also be frequent
  - If an itemset is infrequent, then all of *its supersets* must also be infrequent

- Apriori principle holds due to **the anti-monotone 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

Apriori: Prunning based on support (1)



Apriori: Prunning based on support (2)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

1-itemsets (individual items)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Beer}	**2**
{Bread,Diaper}	**3**
{Milk,Beer}	**2**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

2-itemsets

(No need to generate candidates involving *Coke* or *Eggs*)

minsup = 3

- If every subset is considered:
  $^6C_1 + {}^6C_2 + {}^6C_3 = 41$
- With support-based pruning:
  $6 + 6 + 1 = 13$

3-itemsets

Itemset	Count
{Bread,Milk,Diaper}	**3**

Apriori algorithm

- Generate frequent itemsets of length 1 (i.e., frequent 1-itemsets)
- Set k=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 database
  - Eliminate candidates that are infrequent, leaving only those that are frequent

Apriori: Factors affecting complexity

- Choice of **minimum support threshold**
  - Lowering support threshold results in more frequent itemsets
  - This may increase number of candidates and maximum length of frequent itemsets
- **Dimensionality (number of items)** of the dataset
  - More space is needed to store support count of each item
  - If number of frequent items also increases, both computation and I/O costs may also increase
- **Size of the database**
  - Since Apriori makes multiple passes, run time of algorithm may increase with number of transactions
- **Average transaction width**
  - When the transaction width increases, the max length of frequent itemsets also increases

Rule generation (1)

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L \setminus f$ satisfies the minimum confidence requirement
  - E.g., 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 $\rightarrow \emptyset$ and $\emptyset \rightarrow$ L)

Rule generation (2)

- How to efficiently generate rules from frequent itemsets?
- In general, **confidence does not have an anti-monotone property**

  $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

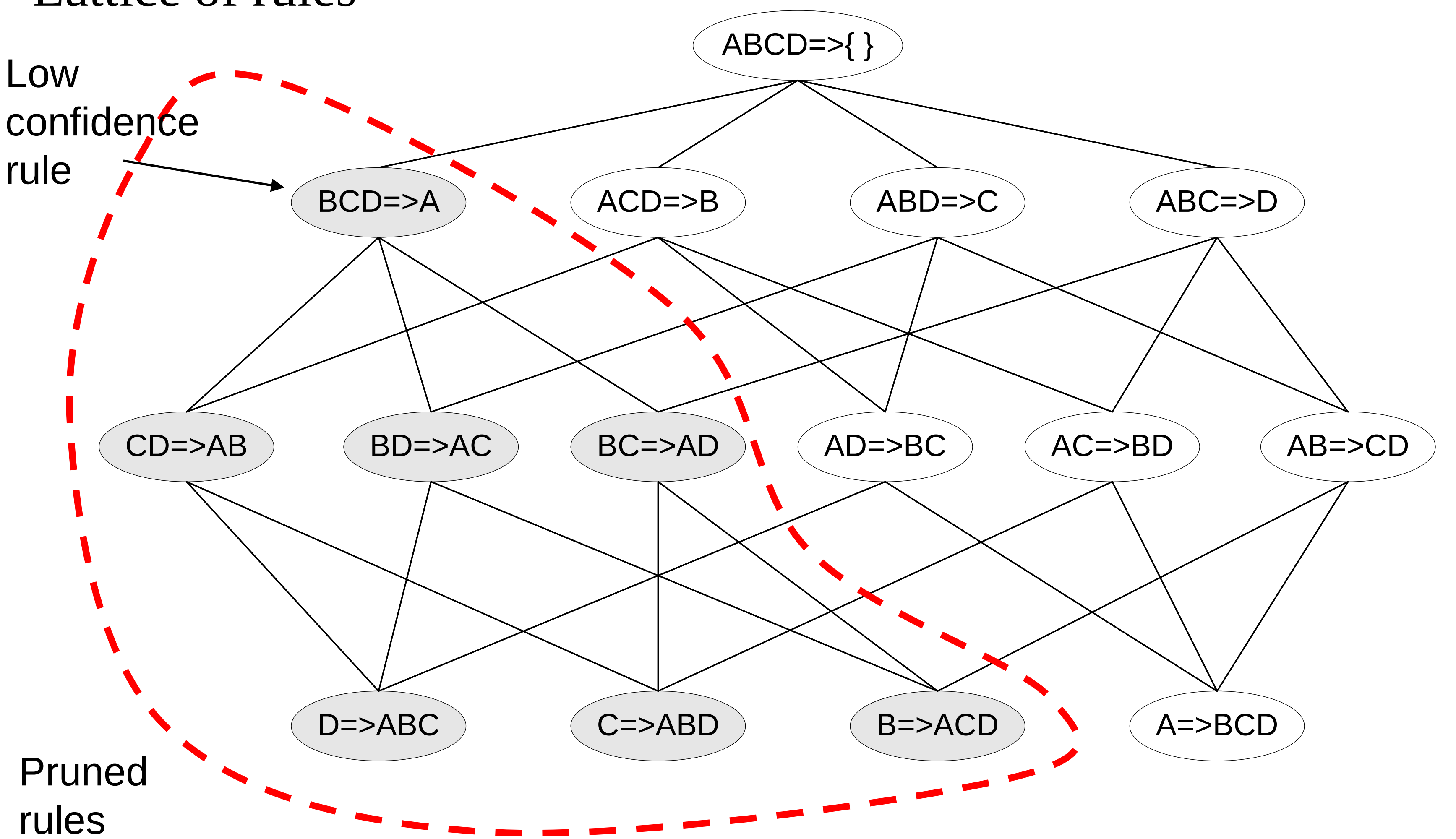
- But, **confidence of rules generated from the same itemset has the anti-monotone property**
  - E.g., For $L = \{A,B,C,D\}$:

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

  - Confidence is anti-monotone w.r.t. the number of items on the right hand side of the rule

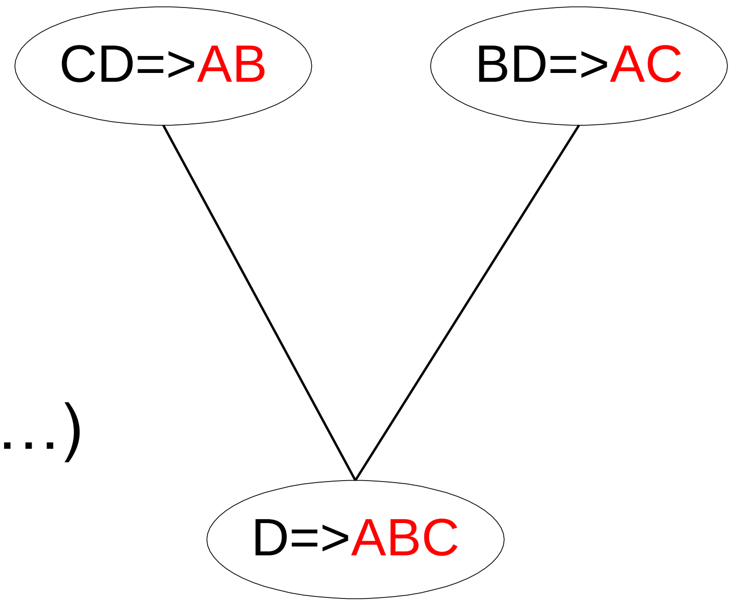
Apriori: Rule generation (1)

Lattice of rules



Apriori: Rule generation (2)

- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent
  - Example: join the two rules (CD => AB, BD => AC) would produce the new candidate rule (D => ABC)



- Prune rule D => ABC, if one of its subsets (AD=>BC, BCD=>A, …) does not have high confidence

References

- Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006.