BITS Pilani

Pilani | Dubai | Goa | Hyderabad

S2-20_DSECFZC415 Association Analysis

DISCLAIMERS

- *The slides presented here are obtained from the authors of the books and from various other contributors. I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified a few slides to suit the requirements of the course.*

Association Analysis Basics

Association Analysis

Association analysis measures the strength of co-occurrence between one item and another.

- The objective of this class of data mining algorithms is not to predict an occurrence of an item, like classification or regression do, but to find usable patterns in the co-occurrences of the items.
- Association rules learning is a branch of an unsupervised learning process that discovers hidden patterns in data, in the form of easily recognizable *rules*

Association algorithms are widely used in retail analysis of transactions, recommendation engines, and online clickstream analysis across web pages.

- One of the popular applications of this technique is called *market basket analysis*, which finds co-occurrences of one retail item with another item within the same retail purchase transaction

Retailer can take advantage of this association for bundle pricing, product placement, and even shelf space optimization within the store layout.

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

Market-Basket transactions

TID	Items
1	Bread, Milk
2	Bread, Diaper, Butter, Beans
3	Milk, Diaper, Butter, Coke
4	Bread, Milk, Diaper, Butter
5	Bread, Milk, Diaper, Coke

Example of Association Rules

$\{Diaper\} \rightarrow \{Butter\}$,
$\{Milk, Bread\} \rightarrow \{Beans, Coke\}$,
$\{Butter, Bread\} \rightarrow \{Milk\}$,

Implication means co-occurrence, not causality!

Definition: Frequent Itemset

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 Itemset

- An itemset whose support is greater than or equal to a *minsup* threshold

TID	Items
1	Bread, Milk
2	Bread, Diaper, Butter, Beans
3	Milk, Diaper, Butter, Coke
4	Bread, Milk, Diaper, Butter
5	Bread, Milk, Diaper, Coke

Definition: Association Rule

- Association Rule
  - An implication expression of the form X → Y, where X and Y are itemsets
  - Example: {Milk, Diaper} → {Butter}

TID	Items
1	Bread, Milk
2	Bread, Diaper, Butter, Beans
3	Milk, Diaper, Butter, Coke
4	Bread, Milk, Diaper, Butter
5	Bread, Milk, Diaper, Coke

- 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

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Butter}$$

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

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

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
- 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

⇒ Computationally prohibitive!

Mining Association Rules

TID	Items
1	Bread, Milk
2	Bread, Diaper, Butter, Beans
3	Milk, Diaper, Butter, Coke
4	Bread, Milk, Diaper, Butter
5	Bread, Milk, Diaper, Coke

Example of Rules:

{Milk,Diaper} → {Butter} (s=0.4, c=0.67)
{Milk,Butter} → {Diaper} (s=0.4, c=1.0)
{Diaper,Butter} → {Milk} (s=0.4, c=0.67)
{Butter} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Butter} (s=0.4, c=0.5)
{Milk} → {Diaper,Butter} (s=0.4, c=0.5)

Observations:

- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Butter}
- 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

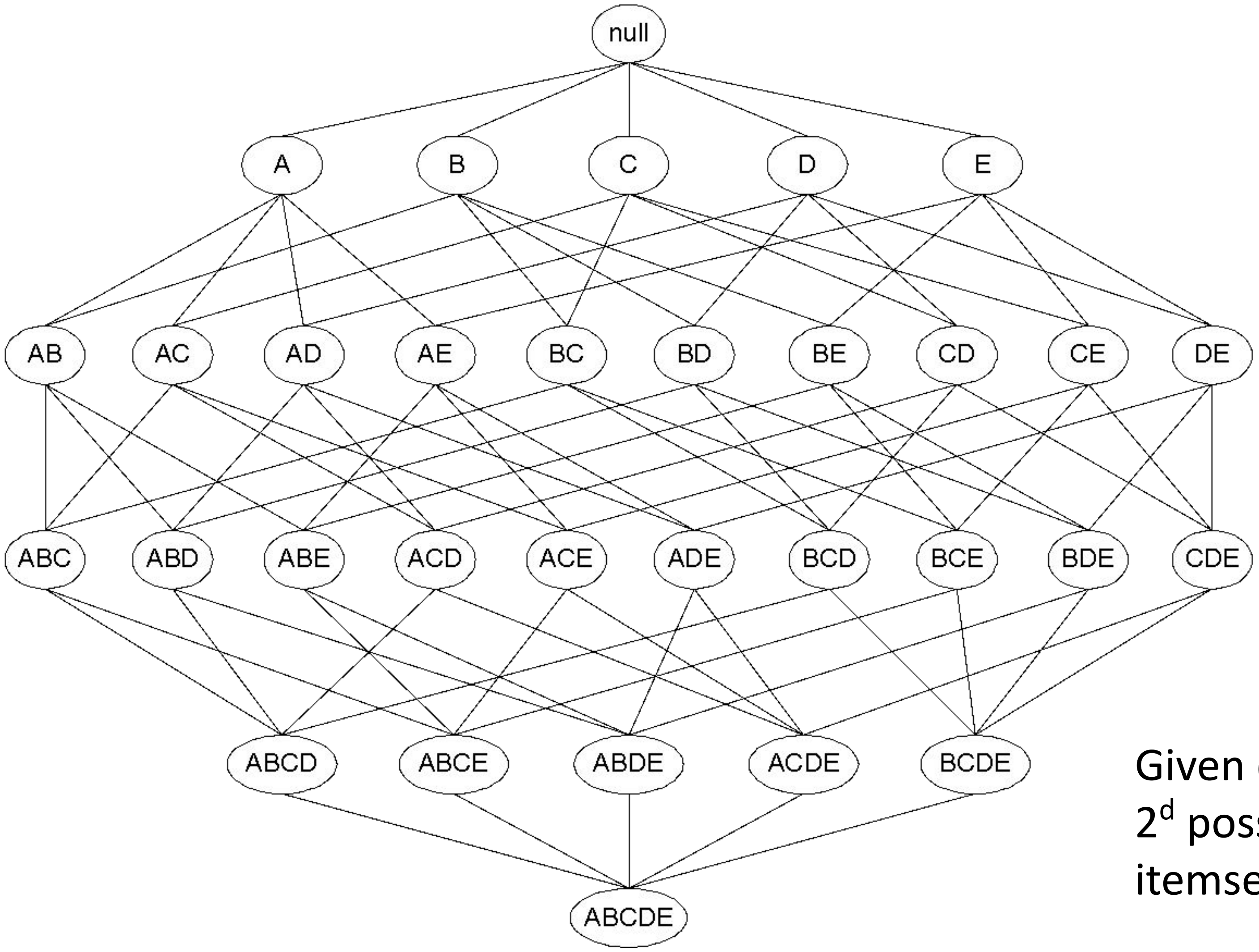
Two-step approach:

1. Frequent Itemset Generation
   – Generate all itemsets whose support ≥ minsup

2. Rule Generation
   – Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

Frequent itemset generation is still computationally expensive

Frequent Itemset Generation

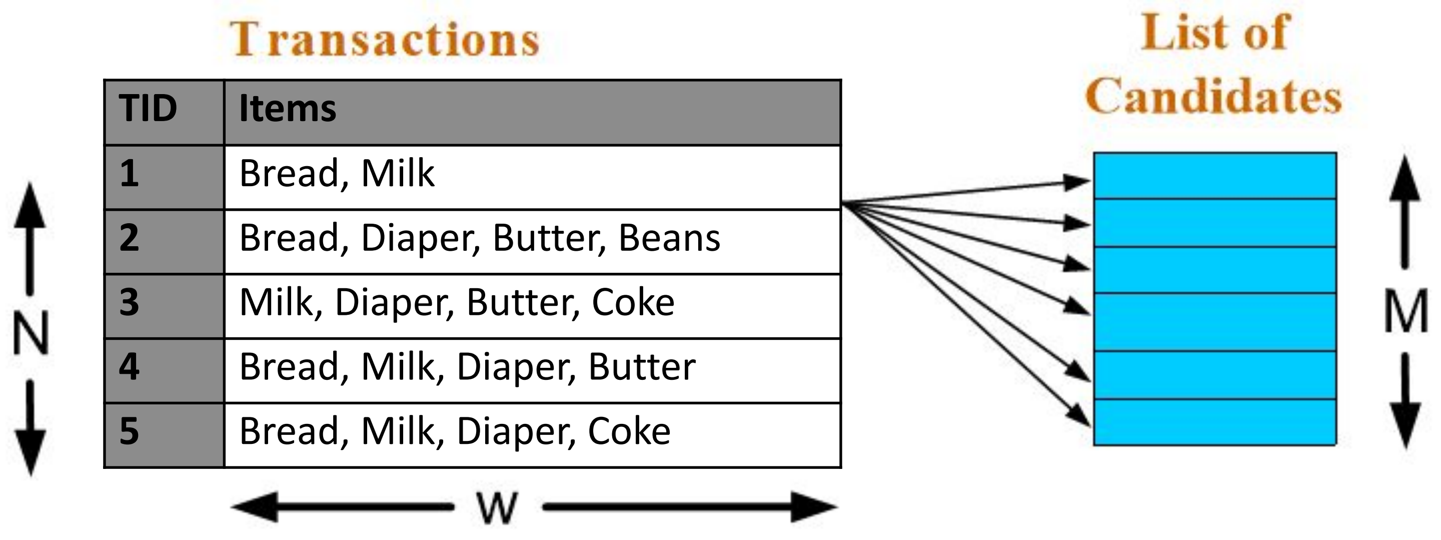


Given d items, there are 2^d possible candidate itemsets

Frequent Itemset Generation

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, Milk
2	Bread, Diaper, Butter, Beans
3	Milk, Diaper, Butter, Coke
4	Bread, Milk, Diaper, Butter
5	Bread, Milk, Diaper, Coke

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

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
- Used by DHP and vertical-based mining algorithms

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

Apriori Algorithm

Mining Association Rules

Two-step approach:

1. Frequent Itemset Generation
   – Generate all itemsets whose support ≥ minsup

2. Rule Generation
   – Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

Frequent itemset generation is computationally expensive

– Apriori principle can be used to reduce computations

Mining Association Rules

Two-step approach:

1. Frequent Itemset Generation
   - Generate all itemsets whose support ≥ minsup

2. Rule Generation
   - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

Frequent itemset generation is still computationally expensive

- ***Apriori*** principle can be used to reduce computations

Reducing Number of Candidates

Apriori principle:

- If an itemset is frequent, then all of its subsets must also be frequent

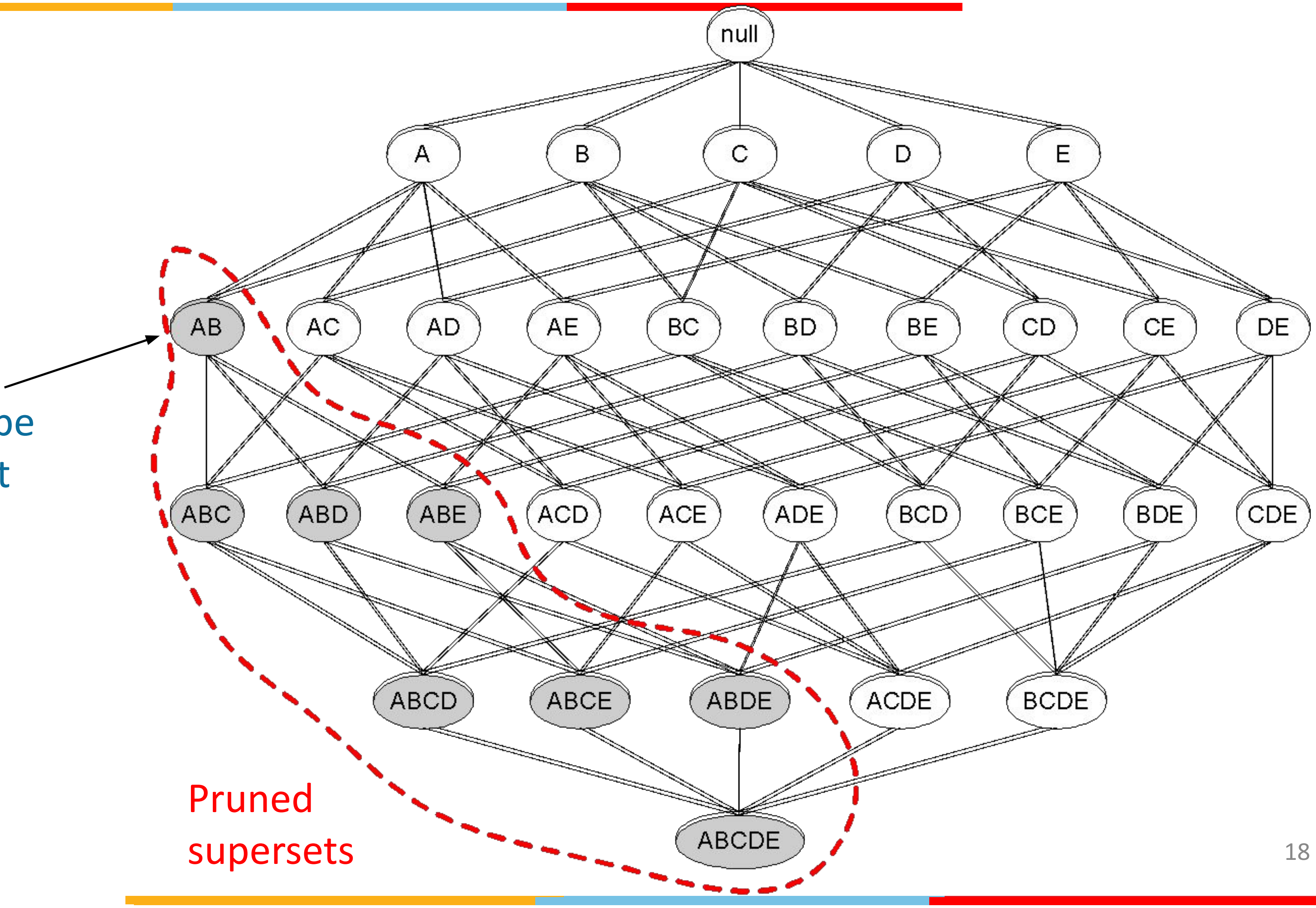
Apriori principle holds due to the following property of the support measure:

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

- Support of an itemset never exceeds the support of its subsets
- This is known as the anti-monotone property of support

The Apriori algorithm was proposed by Rakesh Agrawal and Ramakrishnan Srikant in 1994

Illustrating Apriori Principle



Apriori: A Candidate Generation-and-Test Approach

Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!

Method:

Initially, scan DB once to get frequent 1-itemset

Generate length (k+1) candidate itemsets from length k frequent itemsets

Test the candidates against DB

Terminate when no frequent or candidate set can be generated

TID	Items
1	Bread, Milk
2	Bread, Diaper, Butter, Beans
3	Milk, Diaper, Butter, Coke
4	Bread, Milk, Diaper, Butter
5	Bread, Milk, Diaper, Coke

Apriori Algorithm

Method:

- 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

Important Details of Apriori

How to generate candidates?

- Step 1: self-joining L_k
- Step 2: pruning

Example of Candidate-generation

- L_3={*abc, abd, acd, ace, bcd*}
- Self-joining: $L_3 * L_3$
  - *abcd* from *abc* and *abd*
  - *acde* from *acd* and *ace*
- Pruning:
  - *acde* is removed because *ade* is not in L_3
- C_4={*abcd*}

How to count supports of candidates?

Illustrating Apriori Principle

Item	Count
Bread	4
Coke	2
Milk	4
Butter	3
Diaper	4
Beans	1

Items (1-itemsets)

Minimum Support = 3

TID	Items
1	Bread, Milk
2	Bread, Diaper, Butter, Beans
3	Milk, Diaper, Butter, Coke
4	Bread, Milk, Diaper, Butter
5	Bread, Milk, Diaper, Coke

Illustrating Apriori Principle

Item	Count
Bread	4
Coke	2
Milk	4
Butter	3
Diaper	4
Beans	1

Items (1-itemsets)

Minimum Support = 3

Itemset	Count
{Bread,Milk}	3
{Bread,Butter}	2
{Bread,Diaper}	3
{Milk,Butter}	2
{Milk,Diaper}	3
{Butter,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Beans)

TID	Items
1	Bread, Milk
2	Bread, Diaper, Butter, Beans
3	Milk, Diaper, Butter, Coke
4	Bread, Milk, Diaper, Butter
5	Bread, Milk, Diaper, Coke

Illustrating Apriori Principle

Item	Count
Bread	4
Coke	2
Milk	4
Butter	3
Diaper	4
Beans	1

Items (1-itemsets)

Minimum Support = 3

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

Itemset	Count
{Bread,Milk}	3
{Bread,Butter}	2
{Bread,Diaper}	3
{Milk,Butter}	2
{Milk,Diaper}	3
{Butter,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Beans)

TID	Items
1	Bread, Milk
2	Bread, Diaper, Butter, Beans
3	Milk, Diaper, Butter, Coke
4	Bread, Milk, Diaper, Butter
5	Bread, Milk, Diaper, Coke

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	2

Factors Affecting Complexity

Choice of minimum support threshold

- lowering support threshold results in more frequent itemsets
- this may increase number of candidates and max length of frequent itemsets

Dimensionality (number of items) of the data set

- 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 database

- since Apriori makes multiple passes, run time of algorithm may increase with number of transactions

Average transaction width

- transaction width increases with denser data sets
- This may increase max length of frequent itemsets and traversals of hash tree (number of subsets in a transaction increases with its width)

Can we improve Apriori Efficiency?

Hash-based technique

Transaction reduction

Partitioning

Sampling

Dynamic itemset counting

Hash-based technique

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Create hash table using hash function
h(x, y)=((order of x)*10 + (order of y)) mod 7

bucket address	0	1	2	3	4	5	6
bucket count	2	2	4	2	2	4	4
bucket contents	{I1, I4} {I3, I5}	{I1, I5} {I1, I5}	{I2, I3} {I2, I3} {I2, I3} {I2, I3}	{I2, I4} {I2, I4}	{I2, I5} {I2, I5}	{I1, I2} {I1, I2} {I1, I2} {I1, I2}	{I1, I3} {I1, I3} {I1, I3} {I1, I3}

A 2-itemset with a corresponding bucket count in the hash table that is below the support threshold cannot be frequent

Efficiency Techniques

Transaction reduction

- A transaction that does not contain any frequent *k*-itemsets cannot contain any frequent (*k* + 1)-itemsets. Such a transaction can be removed from further consideration

Partitioning

- It has two phases. In phase I, divide the transactions of *D* into *n* partitions. Each partition has proportionally lower threshold. For each partition, all the *local frequent itemsets* are found.
- Any itemset that is potentially frequent with respect to D must be a frequent itemset in at least one of the partitions. Therefore, all local frequent itemsets are candidate itemsets with respect to D

Dynamic Itemset Counting

- Instead of counting for the entire database, promote a candidate to frequent itemset if it passes a (lower) threshold after partial counting. Afterwards, generate larger patterns using the itemset, so promoted.

Closed Patterns and Max-Patterns

A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$ contains
$^{100}C_1 + {}^{100}C_2 + \ldots + {}^{100}C_{100} = 2^{100} - 1 = 1.27*10^{30}$ sub-patterns!

Solution: *Mine closed frequent patterns and maximal frequent patterns instead*

- An itemset X is closed if X is *frequent* and there exists *no super-pattern* Y ɔ X, *with the same support* as X
- An itemset X is a maximal pattern if X is frequent and there exists no frequent super-pattern Y ɔ X

Closed pattern is a lossless compression of freq. patterns

- Reducing the # of patterns and rules

Closed Patterns and Max-Patterns

Example

- DB = {<a_1 …, a_{100}>, < a_1 …, a_{100}>, < a_1, …, a_{50}>}
- Min_sup = 2

What is the set of closed itemset?

- <a_1, …, a_{100}>: 2
- < a_1, …, a_{50}>: 3

What is the set of maximal pattern?

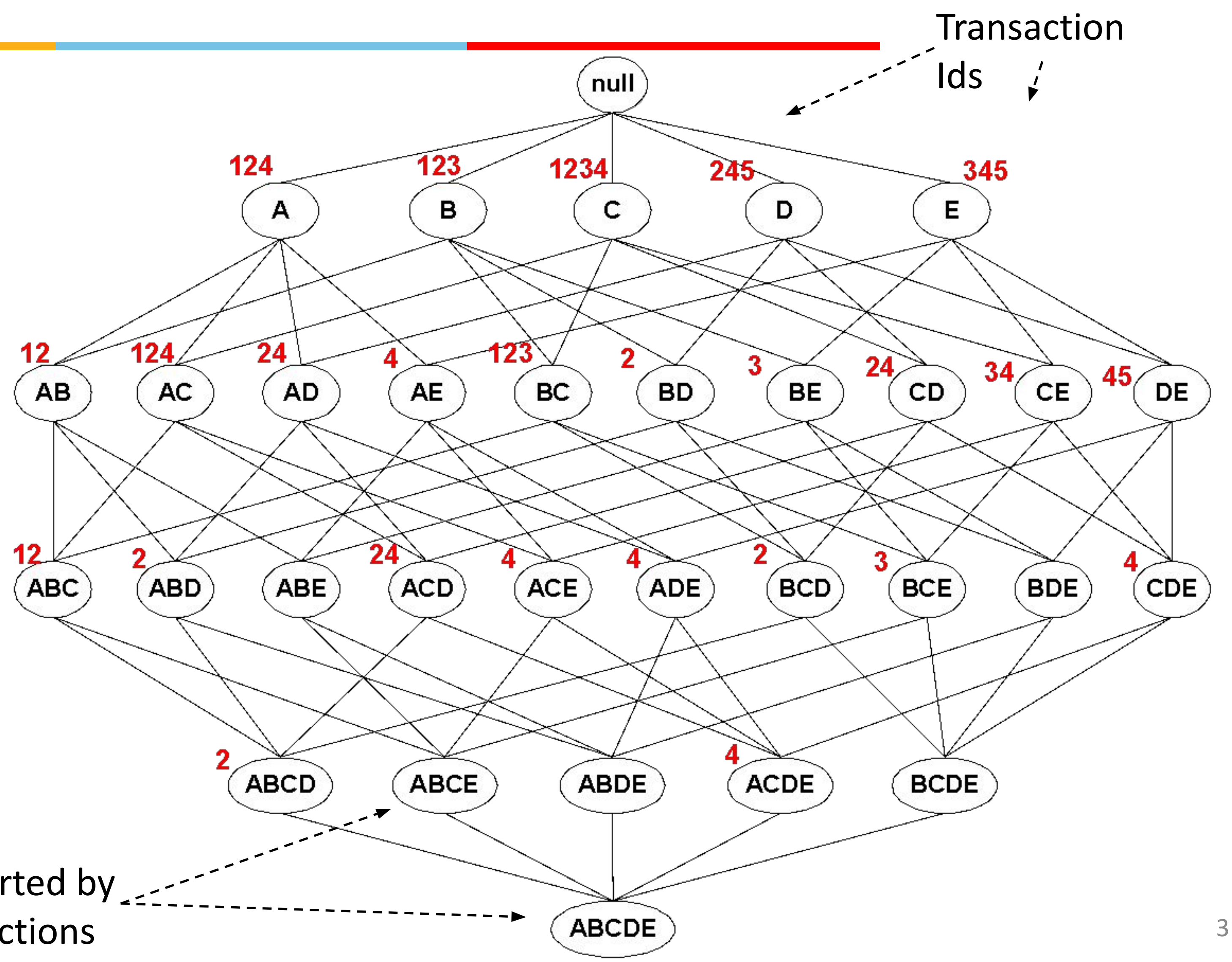
- <a_1, …, a_{100}>: 2

What is the set of all patterns?

- $1.27*10^{30}$

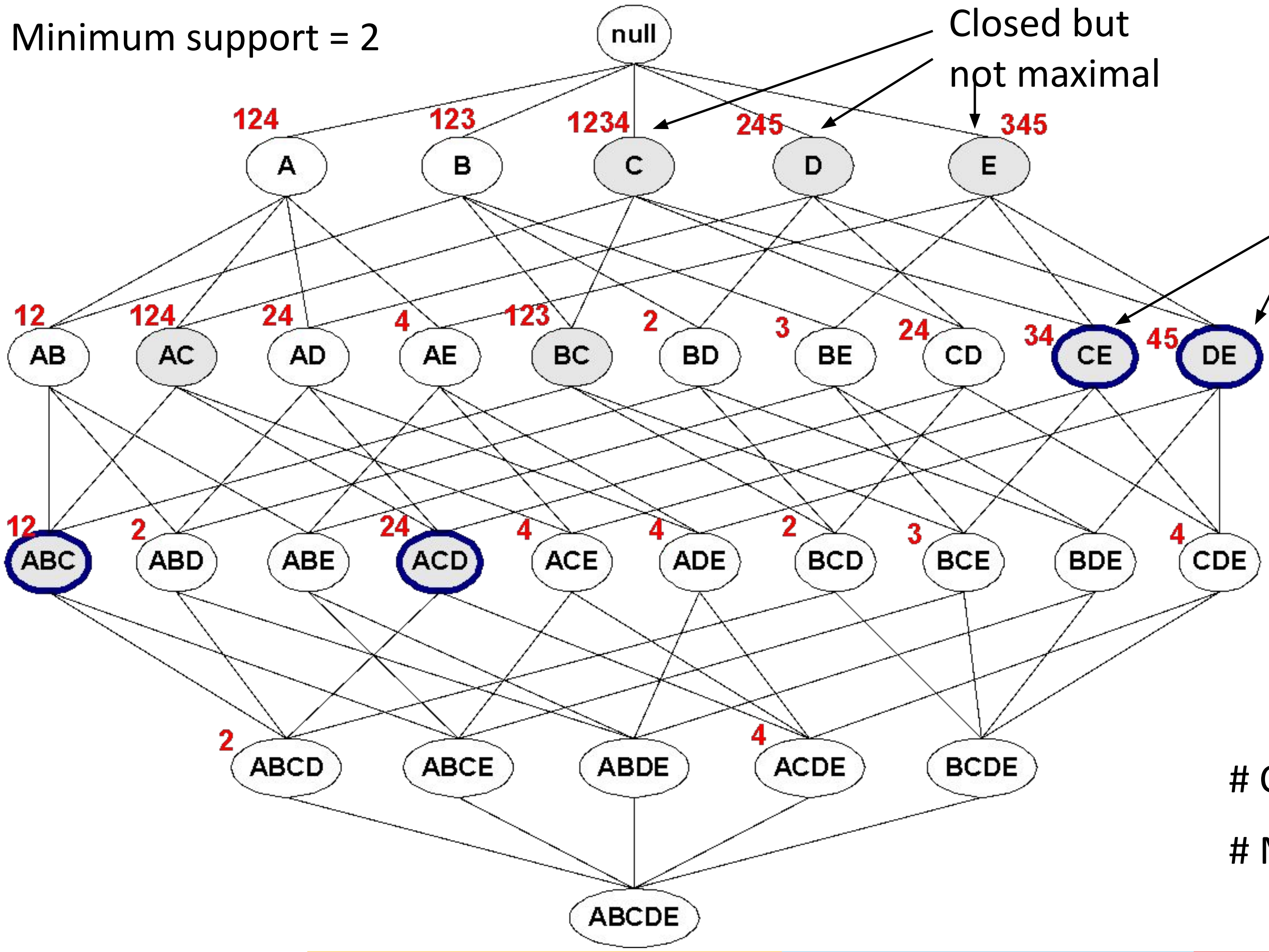
Maximal vs Closed Itemsets

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Maximal vs Closed Frequent Itemsets

Minimum support = 2



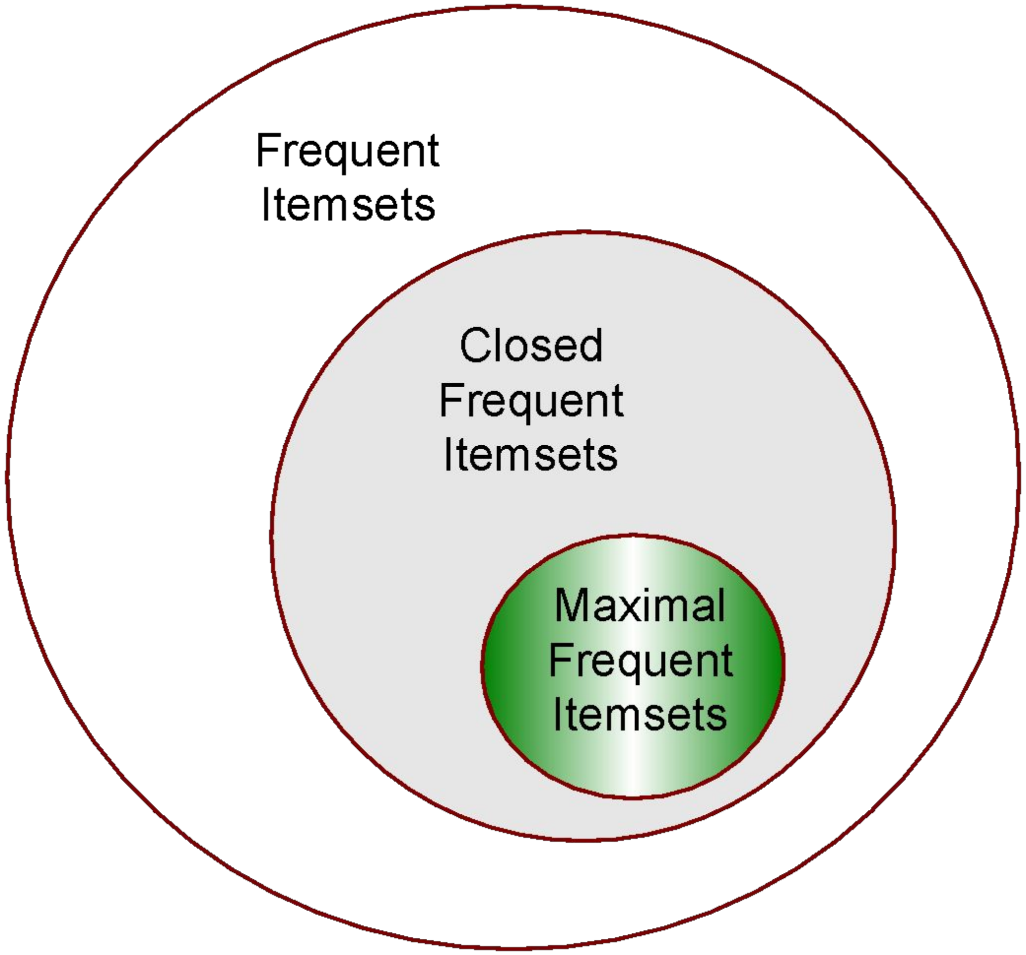
Closed but not maximal

Closed and maximal

# Closed = 9

# Maximal = 4

Maximal vs Closed Itemsets



Prescribed Text Books

	Author(s), Title, Edition, Publishing House
T1	Tan P. N., Steinbach M & Kumar V. “Introduction to Data Mining” Pearson Education
T2	Data Mining: Concepts and Techniques, Third Edition by Jiawei Han, Micheline Kamber and Jian Pei Morgan Kaufmann Publishers
R1	Predictive Analytics and Data Mining: Concepts and Practice with RapidMiner by Vijay Kotu and Bala Deshpande Morgan Kaufmann Publishers

innovate achieve lead

Thank You