PES UNIVERSITY ONLINE

MACHINE INTELLIGENCE

Dr. N MEHALA
Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Module 4 [Unsupervised Learning]

Dr. N MEHALA
Department of Computer Science and Engineering

- 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, 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},

Implication means co-occurrence, not causality!

Association Rule Mining : Definition

- **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

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

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

Association Rule Mining : Definition

- **Association Rule**
  - An implication expression of the form X $\rightarrow$ Y, where X and Y are itemsets
  - Example: {Milk, Diaper} $\rightarrow$ {Beer}

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

- **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:

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

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

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

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
  - 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

  $\Rightarrow$ Computationally prohibitive!

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 Rules:

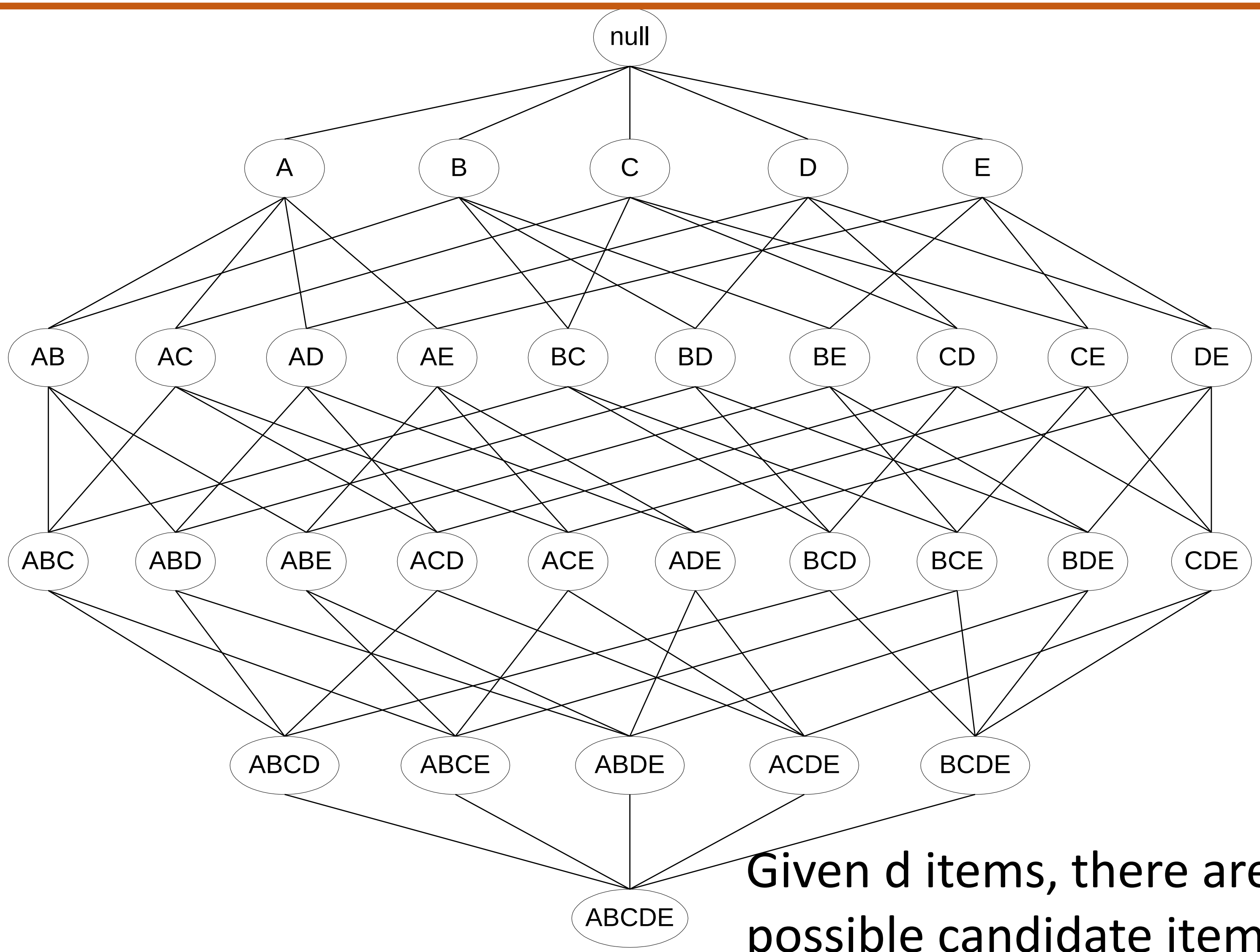
{Milk,Diaper} $\rightarrow$ {Beer} (s=0.4, c=0.67)
{Milk,Beer} $\rightarrow$ {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} $\rightarrow$ {Milk} (s=0.4, c=0.67)
{Beer} $\rightarrow$ {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} $\rightarrow$ {Milk,Beer} (s=0.4, c=0.5)
{Milk} $\rightarrow$ {Diaper,Beer} (s=0.4, c=0.5)

Observations:

- 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

- **Two-step approach:**
  1. Frequent Itemset Generation
     – Generate all itemsets whose support $\geq$ 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

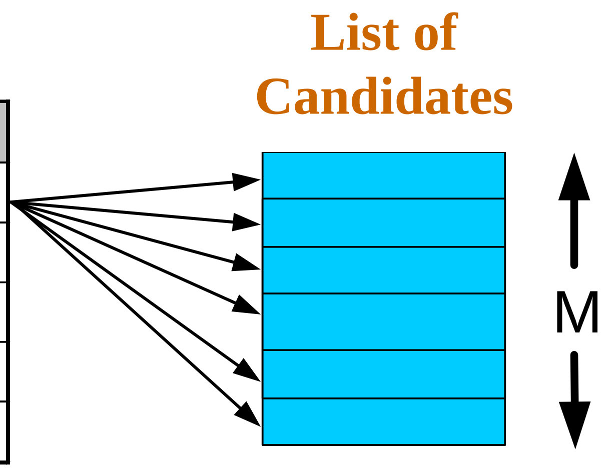
- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database

Transactions

N

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

w



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

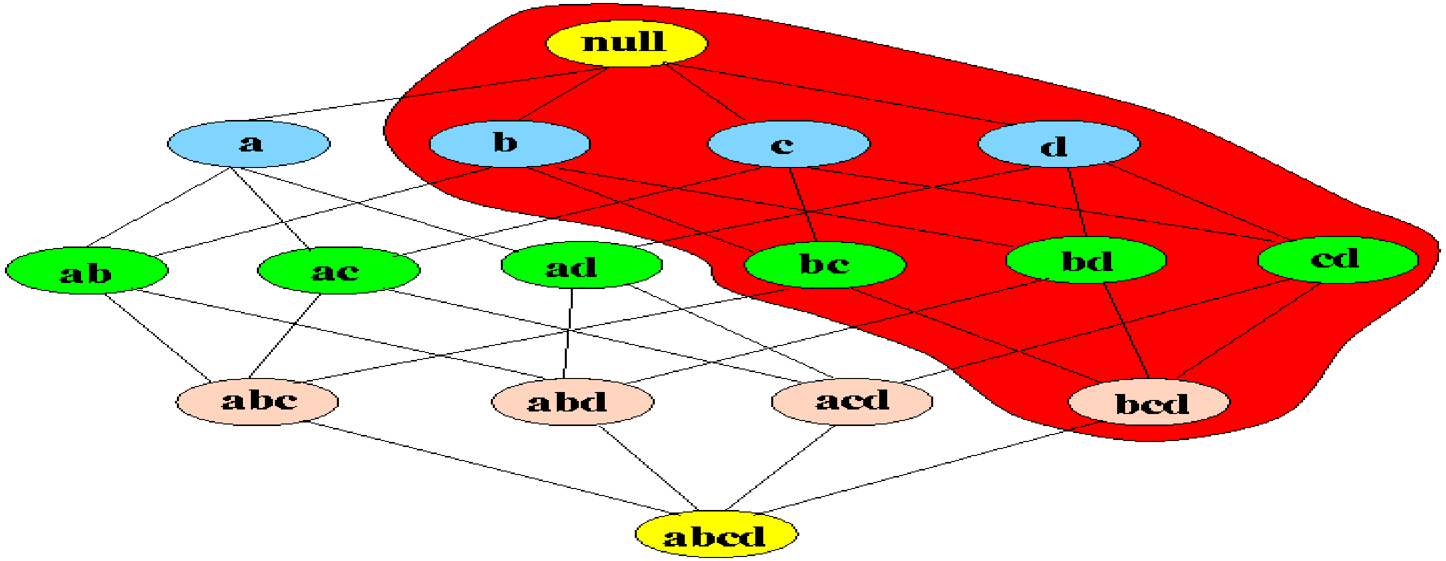
Frequent Itemset Generation Strategies

- 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 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 principle:
  - If an itemset is frequent, then all of its subsets must also be frequent
  - Example: if **{b,c,d}** is **frequent**, then ***all* subsets** of **{b,c,d}** are also **frequent**

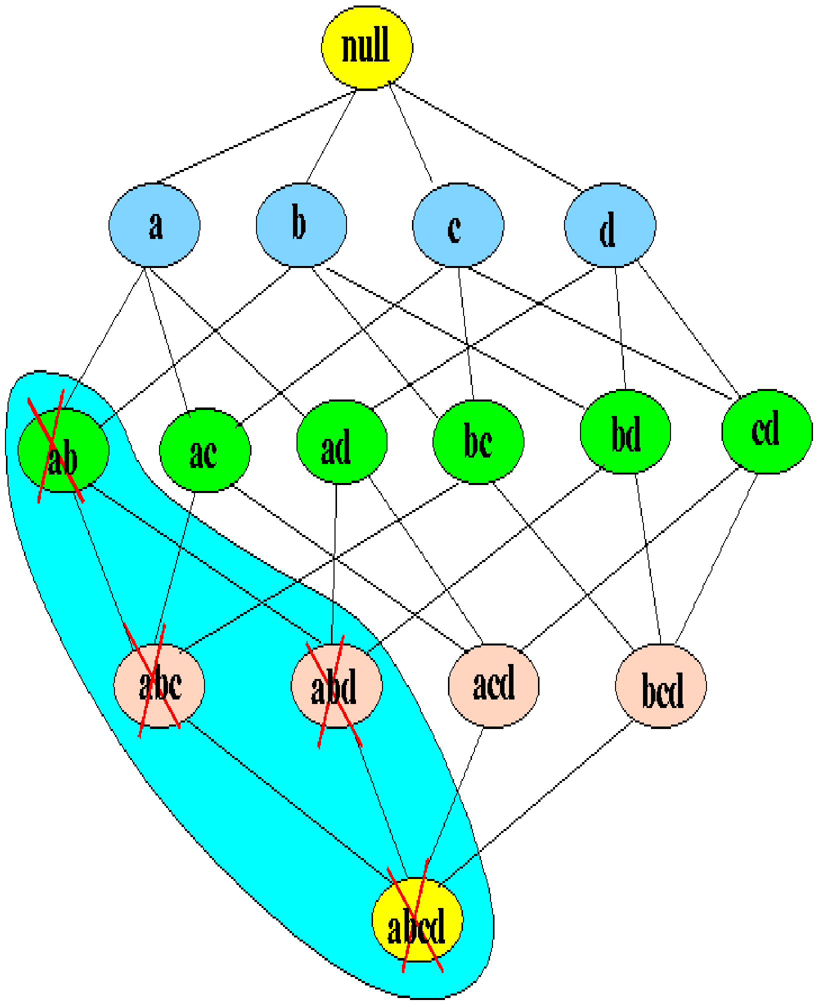


$$\forall X,Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

Applying the Apriori Principle to Eliminate Candidate Sets

Converse of the Apriori Principle:

- If an **itemset *x*** is ***not* frequent** then:
- **all *super* sets of *x*** are also **not frequent**
- **Example:**
- if **{a,b}** is **infrequent**, then all its **super sets** are also **infrequent**:



THANK YOU

Dr. N MEHALA
Department of Computer Science and Engineering
mehala@pes.edu

MACHINE INTELLIGENCE

Dr. N MEHALA
Department of Computer
Science and Engineering

MACHINE INTELLIGENCE

Module 4 [Unsupervised Learning]

Dr. N MEHALA
Department of Computer Science and Engineering

- **Two-step approach:**
  1. Frequent Itemset Generation
     – Generate all itemsets whose support $\geq$ 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

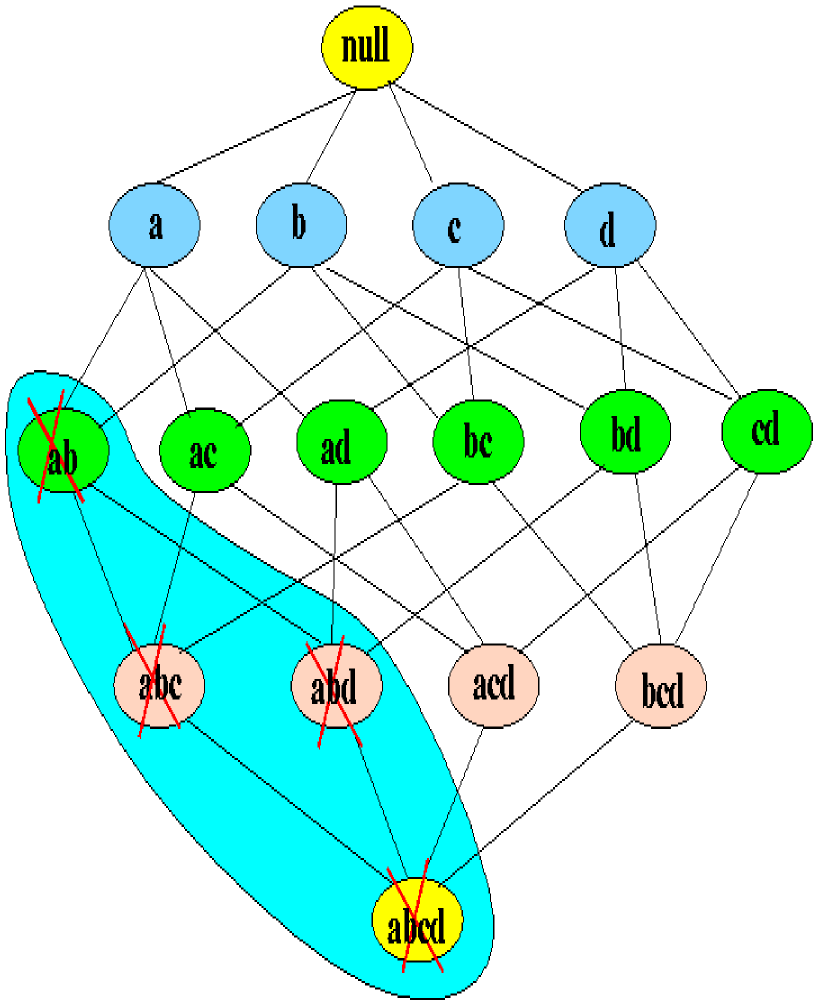
– Generate all itemsets whose support $\geq$ minsup

1. **Apriori Algorithm**
2. **FP-Growth Algorithm**
3. H-Mine
4. CLOSET
5. CHARM

Applying the Apriori Principle to Eliminate Candidate Sets

Converse of the Apriori Principle:

- If an **itemset *x*** is ***not* frequent** then:
- **all *super* sets of *x*** are also **not frequent**
- **Example:**
- if **{a,b}** is **infrequent**, then all its **super sets** are also **infrequent**:



The Apriori Algorithm - Example

Min support =50%

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

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

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
{2 3 5}	2

Illustrating Apriori Principle

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

Items (1-itemsets)

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

Pairs (2-itemsets)
(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

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

- 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

THANK YOU

Dr. N MEHALA
Department of Computer Science and Engineering
mehala@pes.edu

MACHINE INTELLIGENCE

Dr. N MEHALA
Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Module 4 [Unsupervised Learning]

Dr. N MEHALA

Department of Computer Science and Engineering

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 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

Reducing Number of comparisons

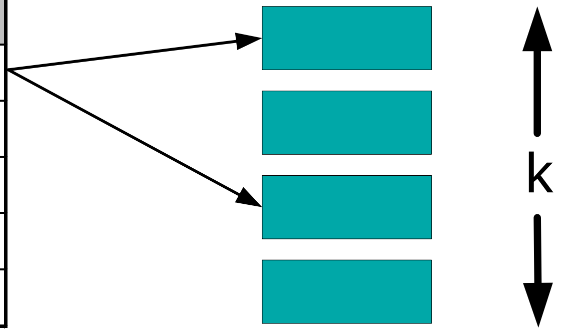
- Candidate counting:
  - Scan the database of transactions to determine the support of each candidate itemset
  - To reduce the number of comparisons, store the candidates in a hash structure
    - Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets

Transactions

N

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

Hash Structure



Buckets

Reducing Number of comparisons

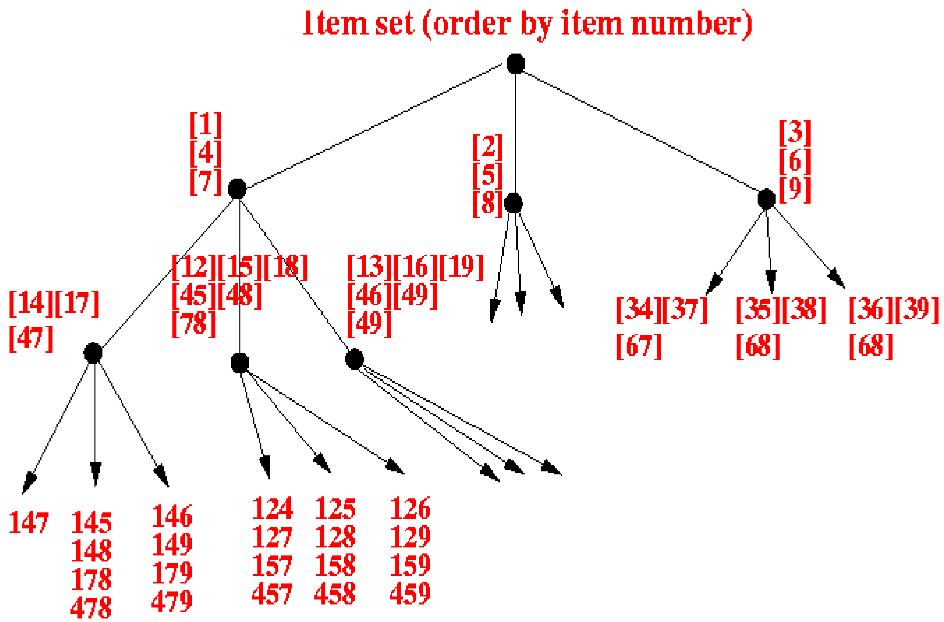
- In the **Apriori** algorithm, the **counters** for the **candidate itemsets** are **partitioned** into **different buckets** and **stored** in a **hash tree** - this speeds up the search for an item set

Reducing Number of comparisons

- **Example: 3-item hash tree** for transactions containing items

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

3-item set hash tree using h(x) = x mod 3

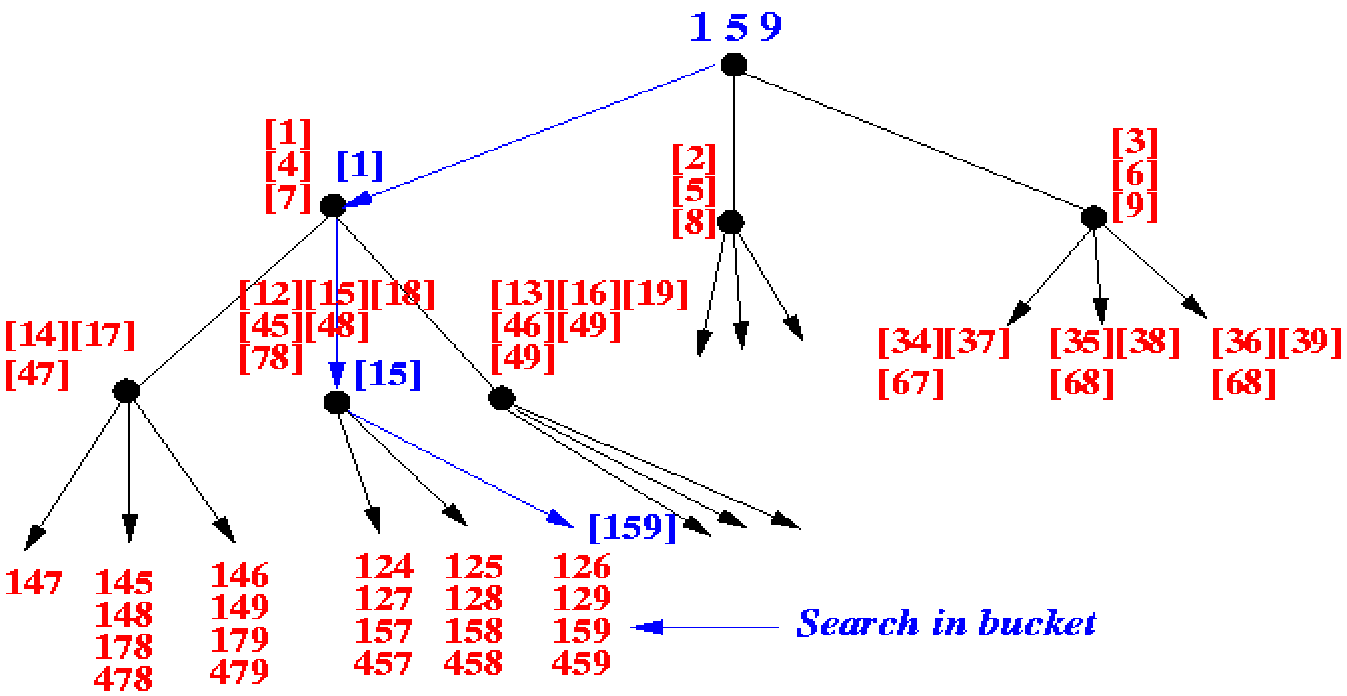


- The **leaves** of the tree contains the **counters** for the different **3-item item sets**
- The **items** in a transaction is first **sorted**
- We then form ***all*** **3 item itemsets** from the **items** in the **transaction.**
- The **3-item itemset** is hashed using **hash(x) = x mod 3** to **locate the counter** for the itemset

- **Concrete example:**

finding the **counter** for itemset **1 5 9**

3-item set hash tree using h(x) = x mod 3

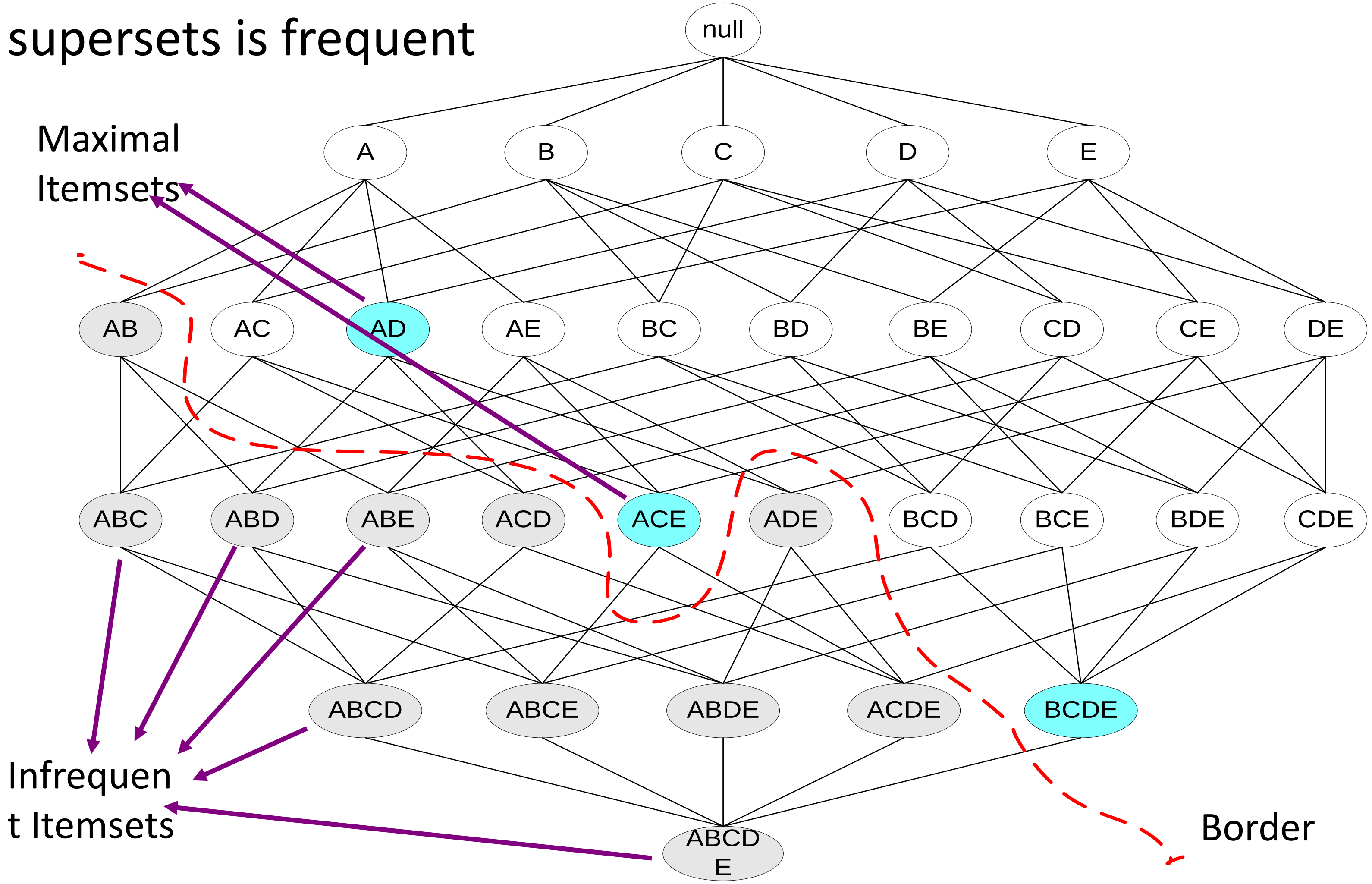


Factors Affecting Complexity

- Choice of minimum support threshold
  - lowering support threshold results in more frequent IS
  - 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)

Maximal Frequent Itemset

An itemset is maximal frequent if none of its immediate supersets is frequent



Closed ItemSet

- An itemset is closed if none of its immediate supersets has the same support as the itemset

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

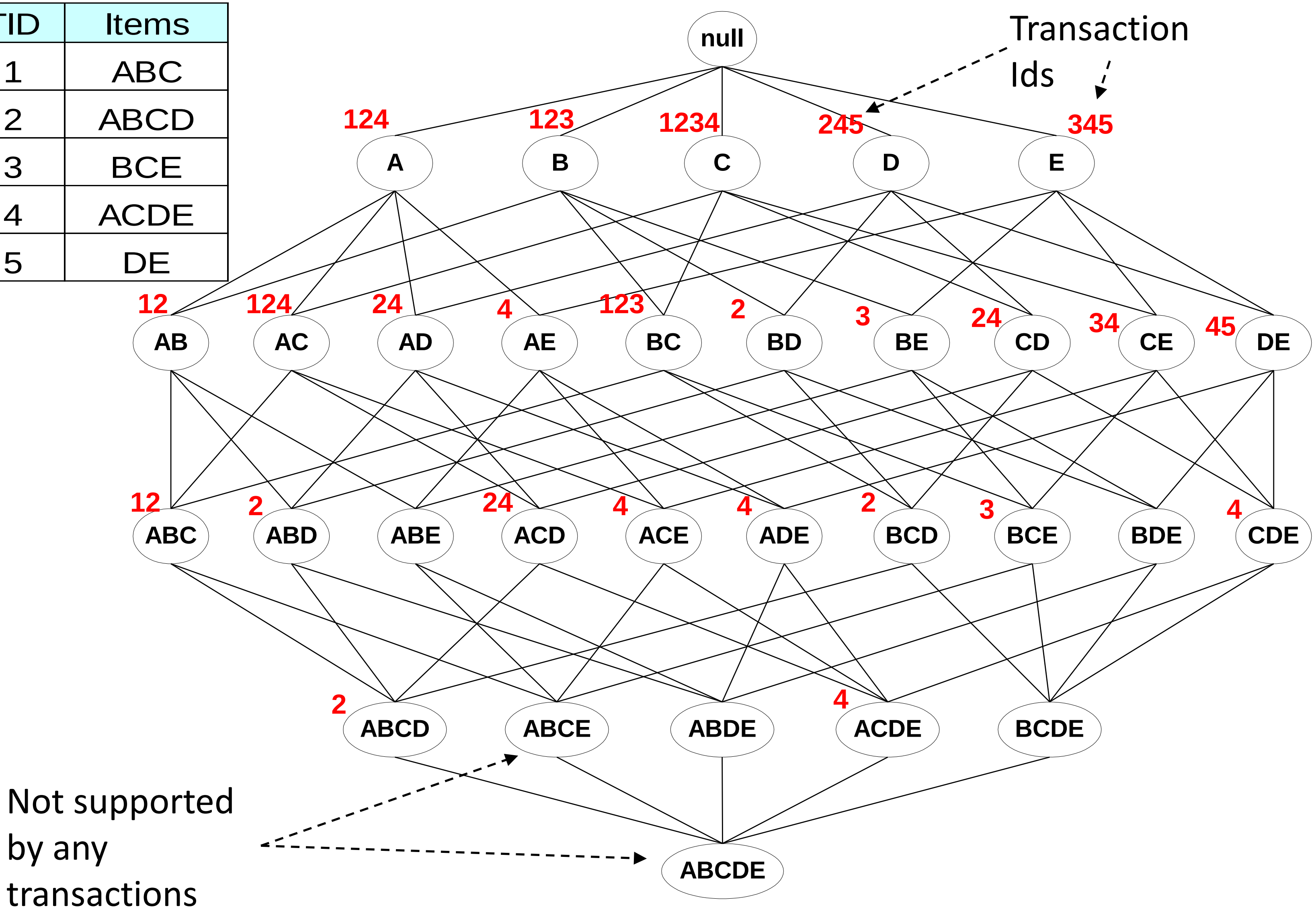
Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

MACHINE INTELLIGENCE

Maximal Vs Closed ItemSets

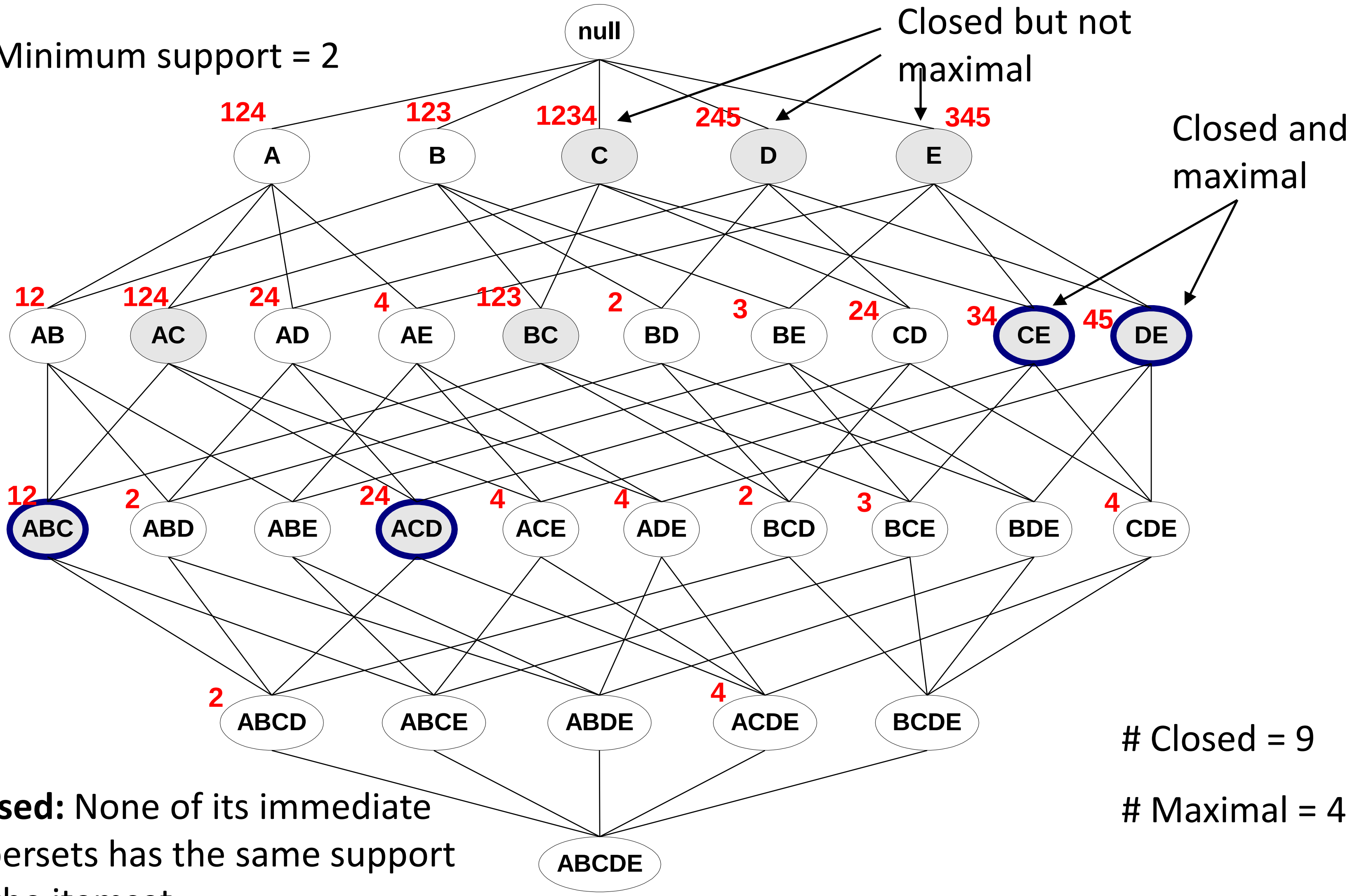
TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Maximal Vs Closed Frequent ItemSets

Maximal: None of its immediate supersets is frequent

Minimum support = 2

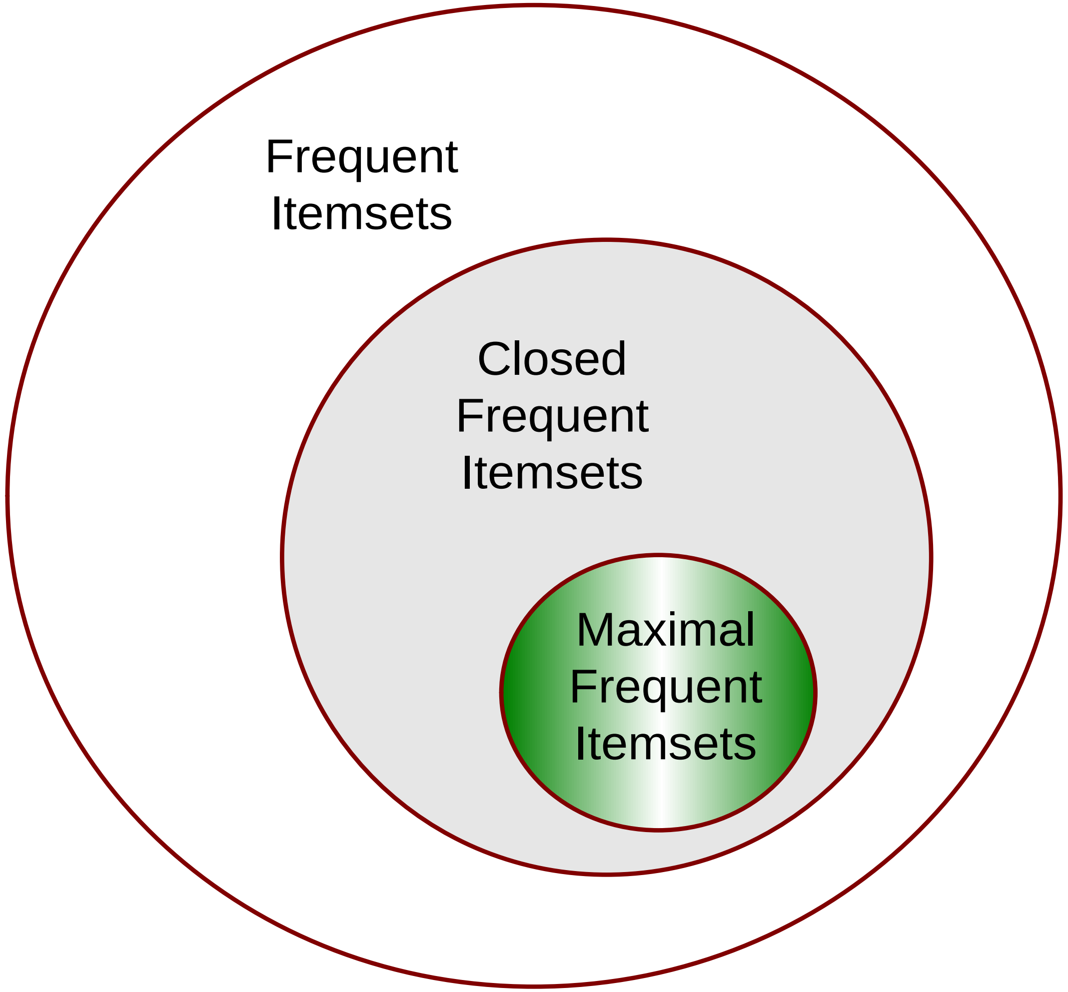


Closed = 9

Maximal = 4

Closed: None of its immediate supersets has the same support as the itemset

Maximal Vs Closed ItemSets



- **Two-step approach:**
  1. Frequent Itemset Generation
     – Generate all itemsets whose support $\geq$ minsup
  2. Rule Generation
     – Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

- 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 $L \rightarrow \emptyset$ and $\emptyset \rightarrow L$)

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

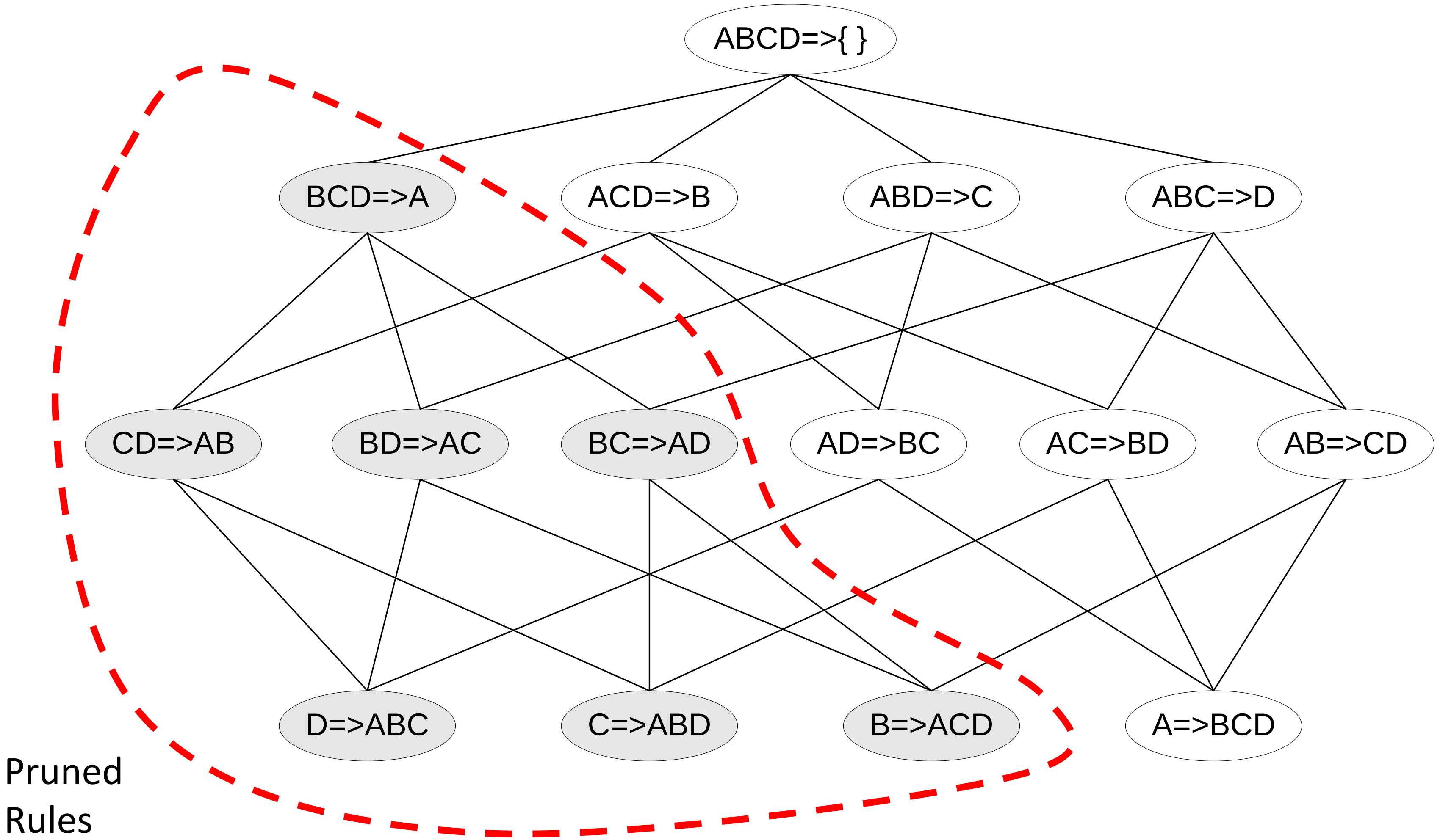
    $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

  - But confidence of rules generated from the same itemset has 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 w.r.t. number of items on the RHS of the rule

Rule Generation for Apriori Algorithm



Summary

- Association Rule Mining Task
- Frequent Item Set Generation : Apriori Algorithm
- Factors Affecting Complexity

Resources

- http://www2.ift.ulaval.ca/~chaib/IFT-4102-7025/public_html/Fichiers/Machine_Learning_in_Action.pdf
- http://wwwusers.cs.umn.edu/~kumar/dmbook/.
- ftp://ftp.aw.com/cseng/authors/tan
- http://web.ccsu.edu/datamining/resources.html

THANK YOU

Dr. N MEHALA

Department of Computer Science and Engineering

mehala@pes.edu