Data Science

Lecture 11 - Association Rule Mining

Asst. Prof. Dr. Santitham Prom-on

Department of Computer Engineering, Faculty of Engineering
King Mongkut's University of Technology Thonburi

Overview

Learning Outcome

- Understand the basic concepts of association rule
- Analyze itemsets
- Perform association analysis

Agenda

- Itemsets
- Association rules
- Rule generation

https://colab.research.google.com/drive/1KofGyVoe-jSInBKQyU_gHEUfsqvPr9VX

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, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

{Diaper} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {Milk},

Implication means co-occurrence, not causality!

Frequent itemset

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

- **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

Frequent itemset

- **Association rule**
  - An implication expression of the form X → Y, where X and Y are itemsets
  - Example:

    {Milk, Diaper} → {Beer}
- **Rule evaluation metrics**
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measure 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

Example:

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

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

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

Association rule mining tasks

- 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 of each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds
  - Computationally prohibitive!!!

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

Example of Rules:

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

Observation:

- 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

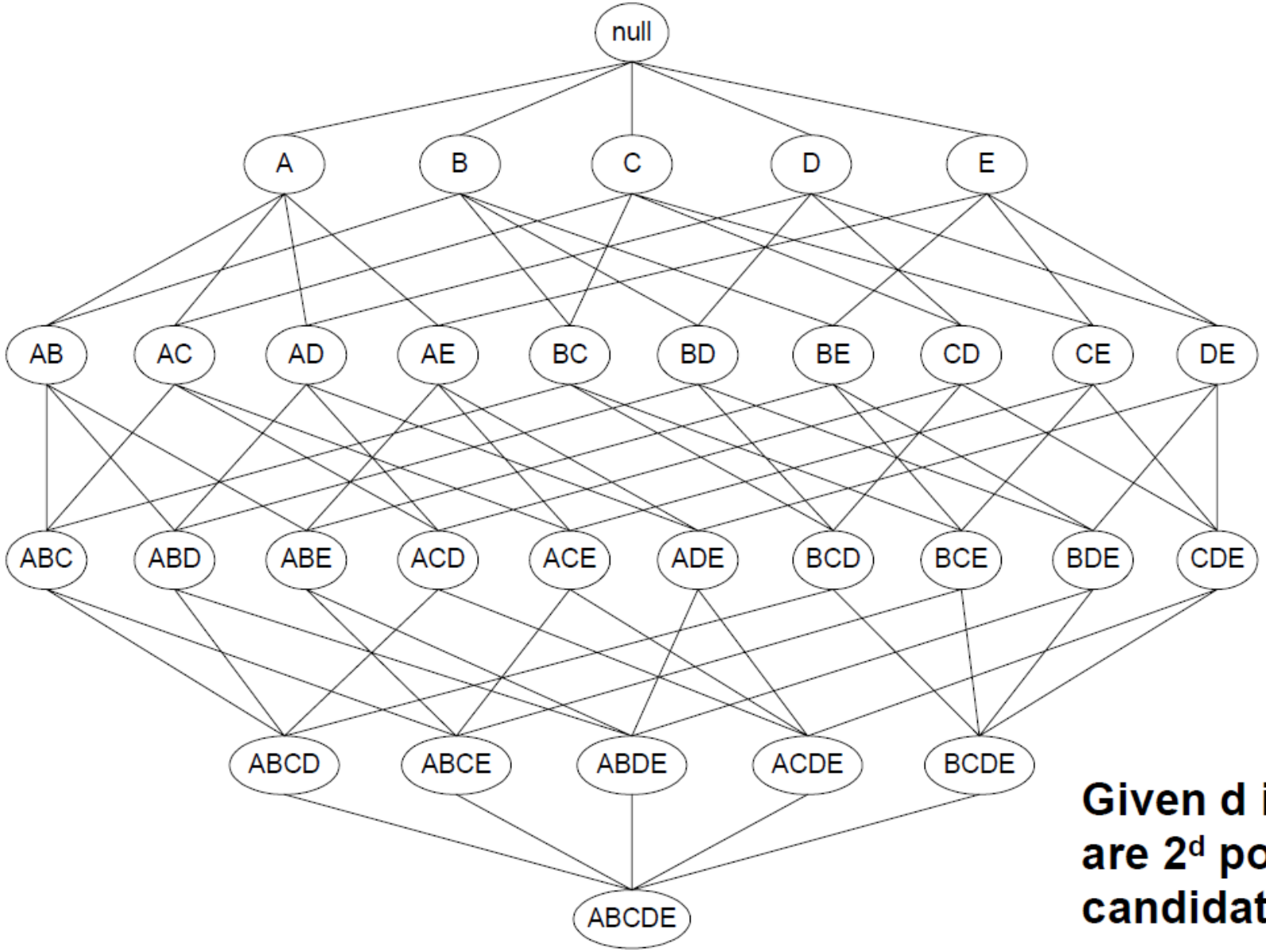
Mining association rules

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

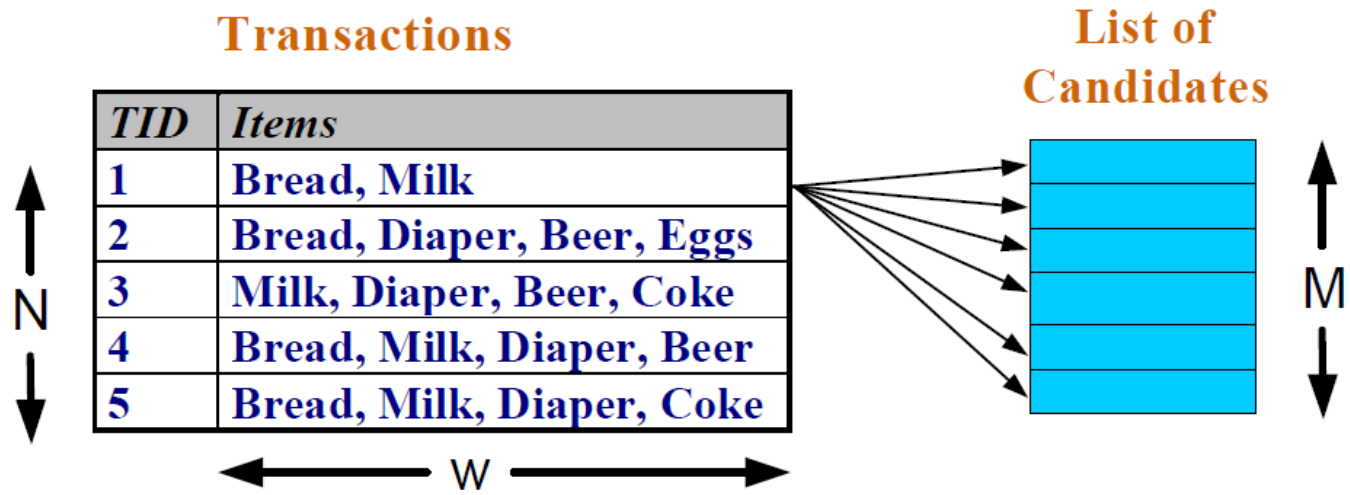


Given d items, there are 2^d possible candidate itemsets

Frequent itemset generation

Brute-force approach:

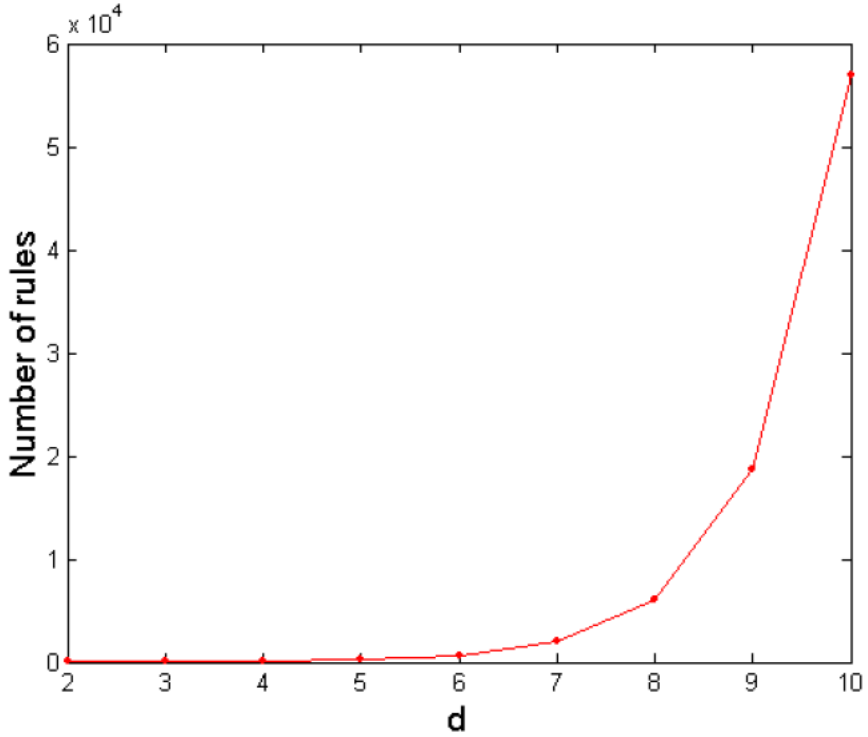
- Each itemset in the lattice is a candidate frequent itemset
- Count the support of each candidate by scanning the database



- 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:



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

$$= 3^d - 2^{d+1} + 1$$

If d=6, R = 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 the 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 number of candidates

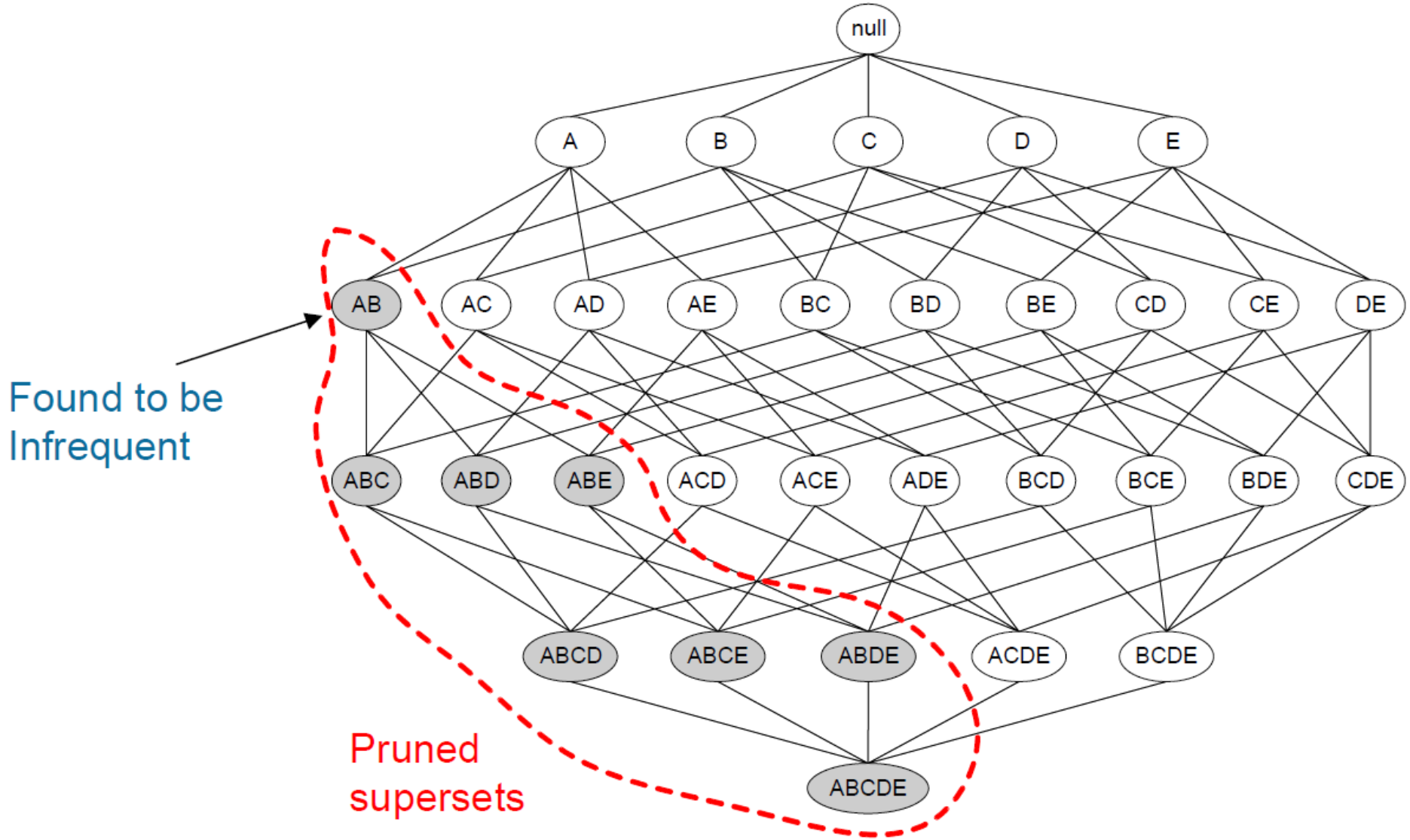
- Apriori principle
  - If an itemset is frequent, then all of its subsets must also be frequent

- 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



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,

$$6 + 6 + 1 = 13$$

Apriori algorithm – Method

- Let k = 1
- Generate frequent itemset 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

Closed itemset

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

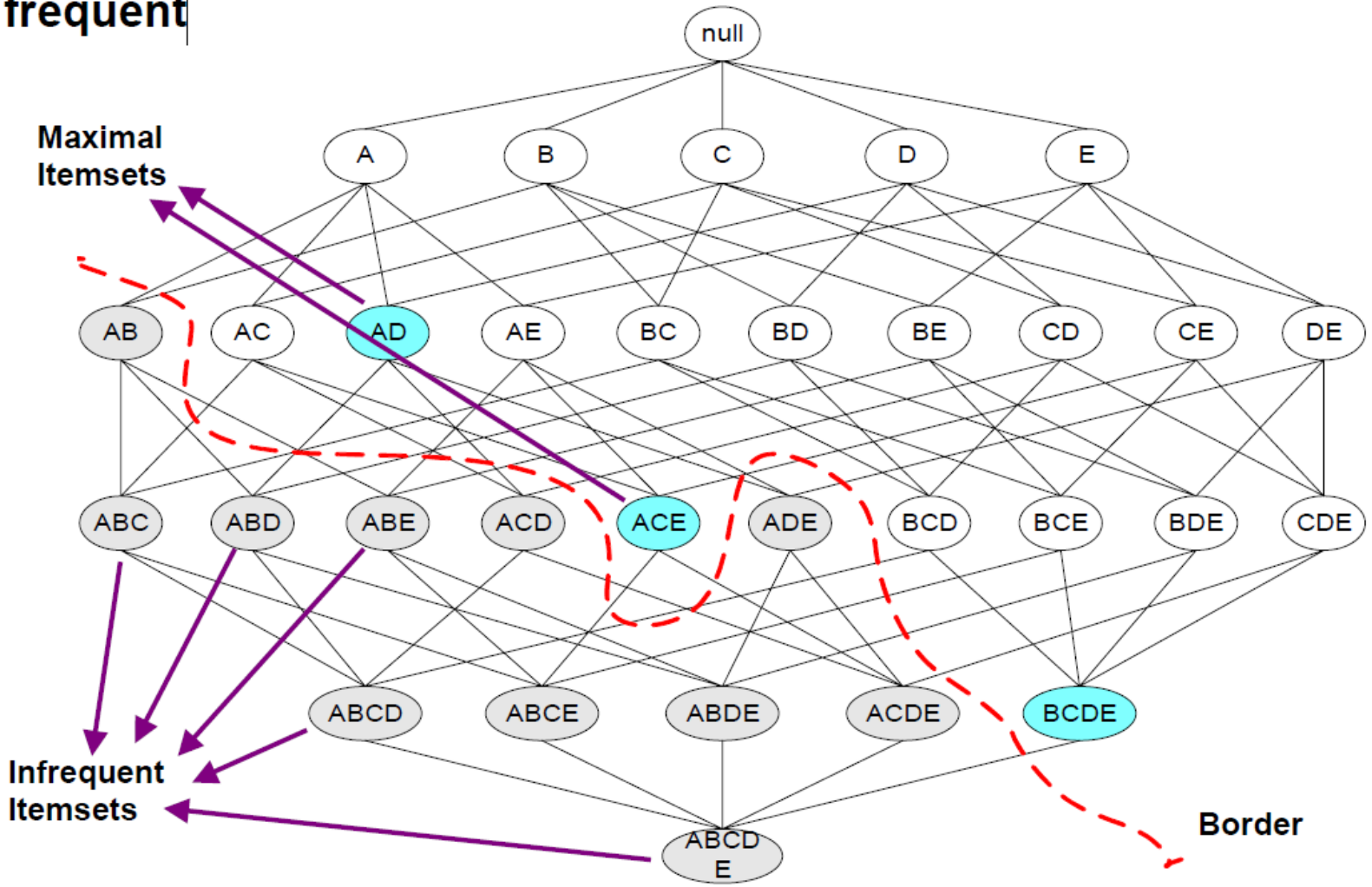
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

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

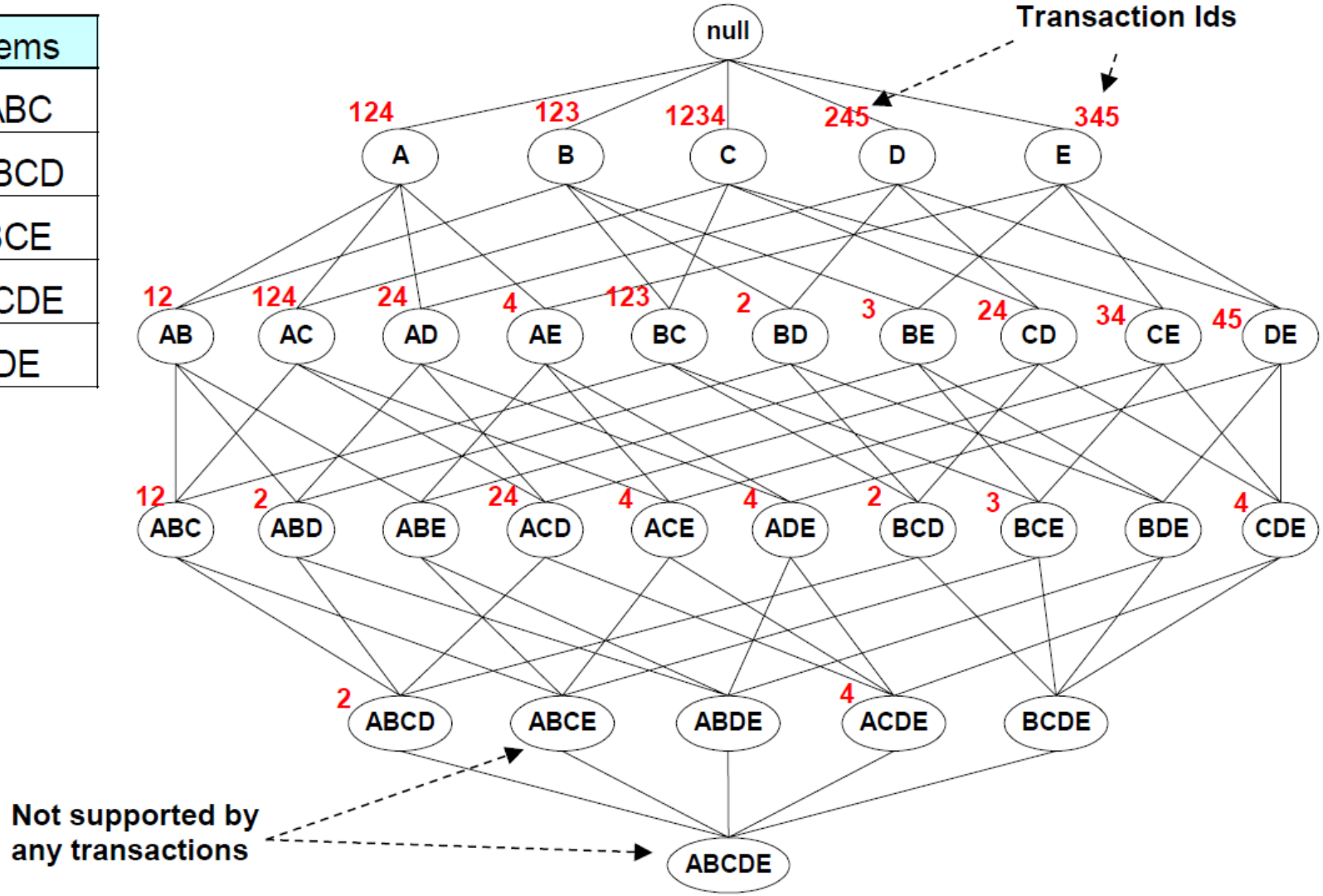
Maximal frequent itemset

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

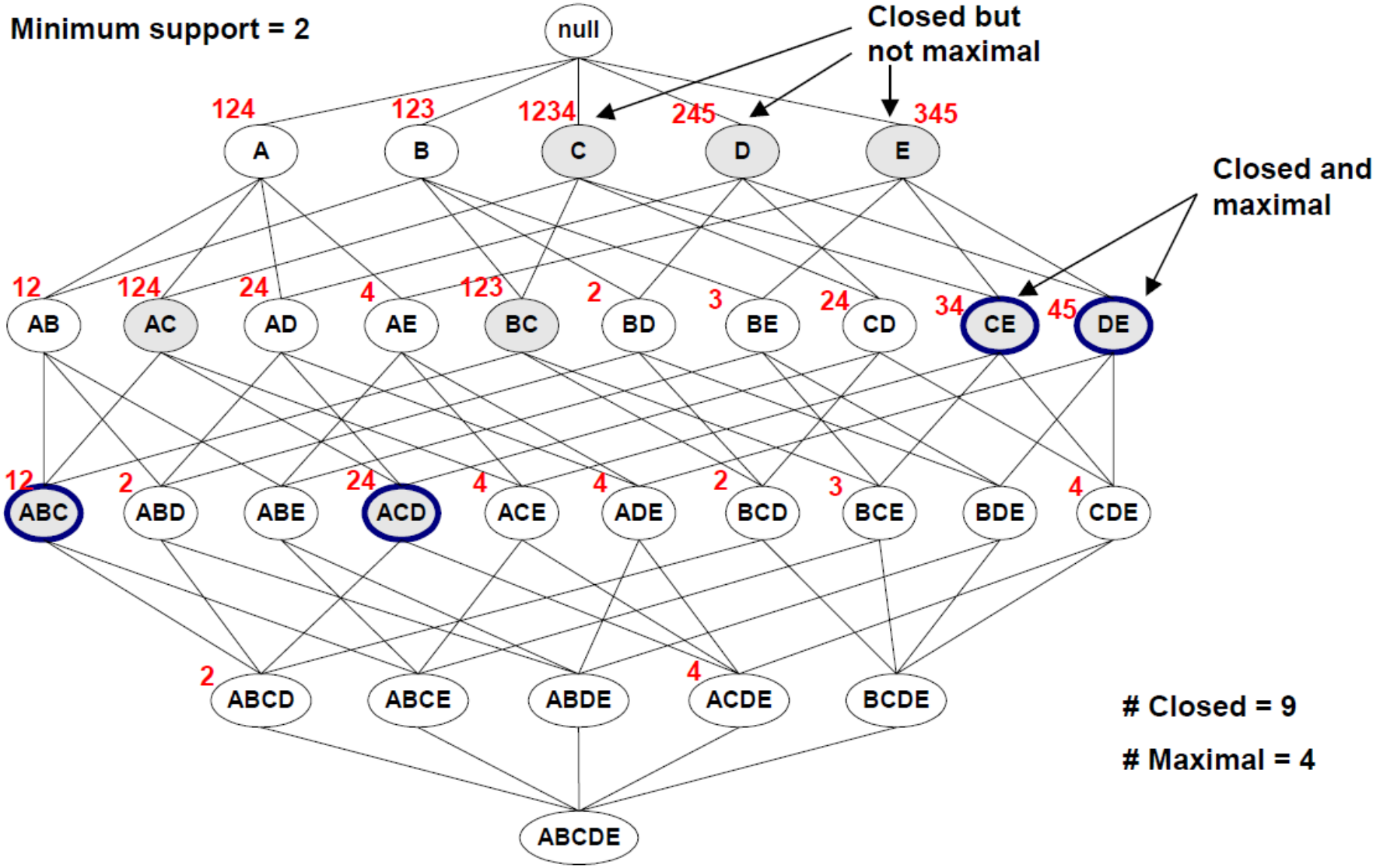


Maximal vs closed itemsets

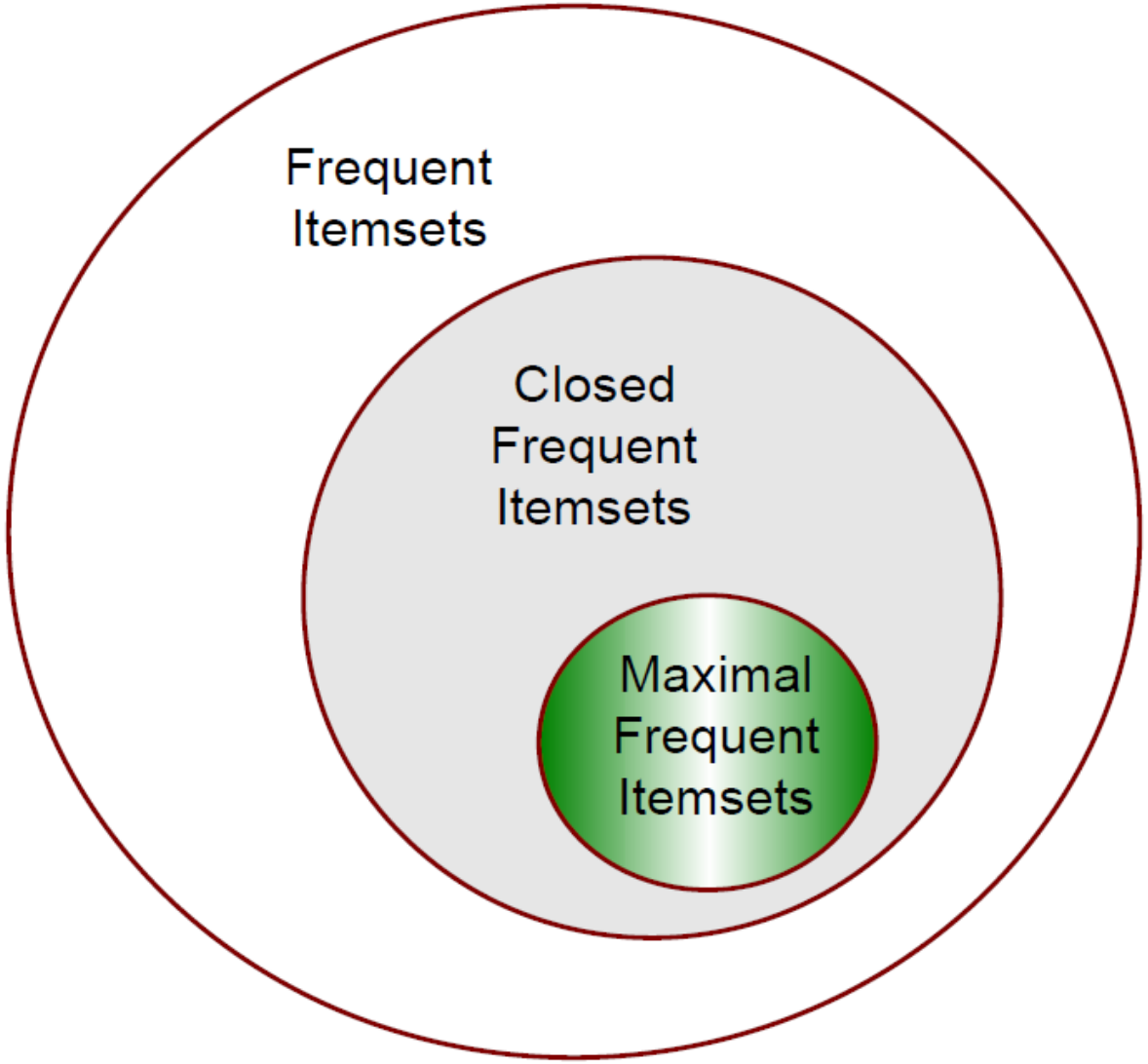
TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Maximal vs closed itemsets



Maximal vs closed itemsets



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:

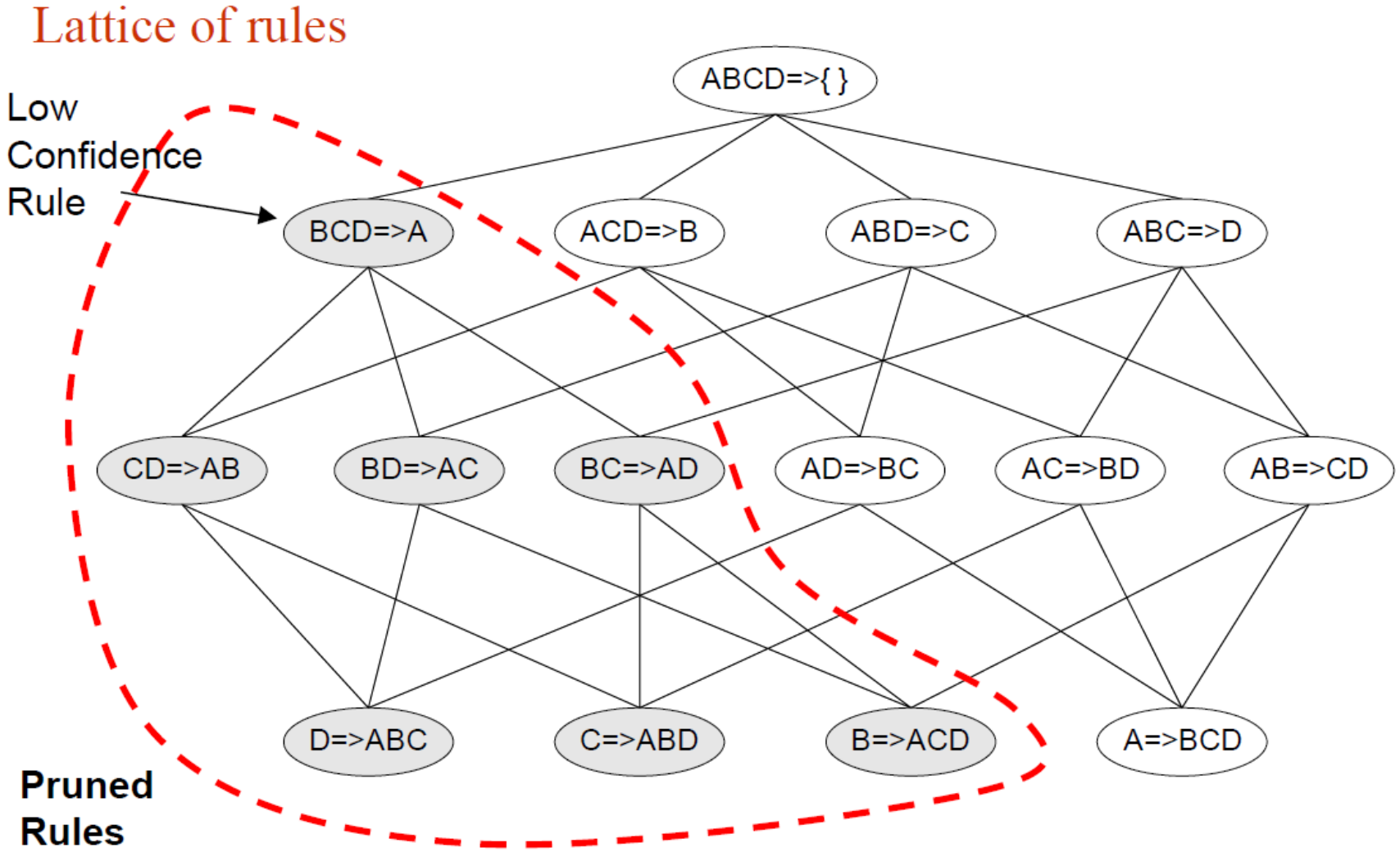
ABC →D,	ABD →C,	ACD →B,	BCD →A,
A →BCD,	B →ACD,	C →ABD,	D →ABC
AB →CD,	AC → BD,	AD → BC,	BC →AD,
BD →AC,	CD →AB,		

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

Rule generation

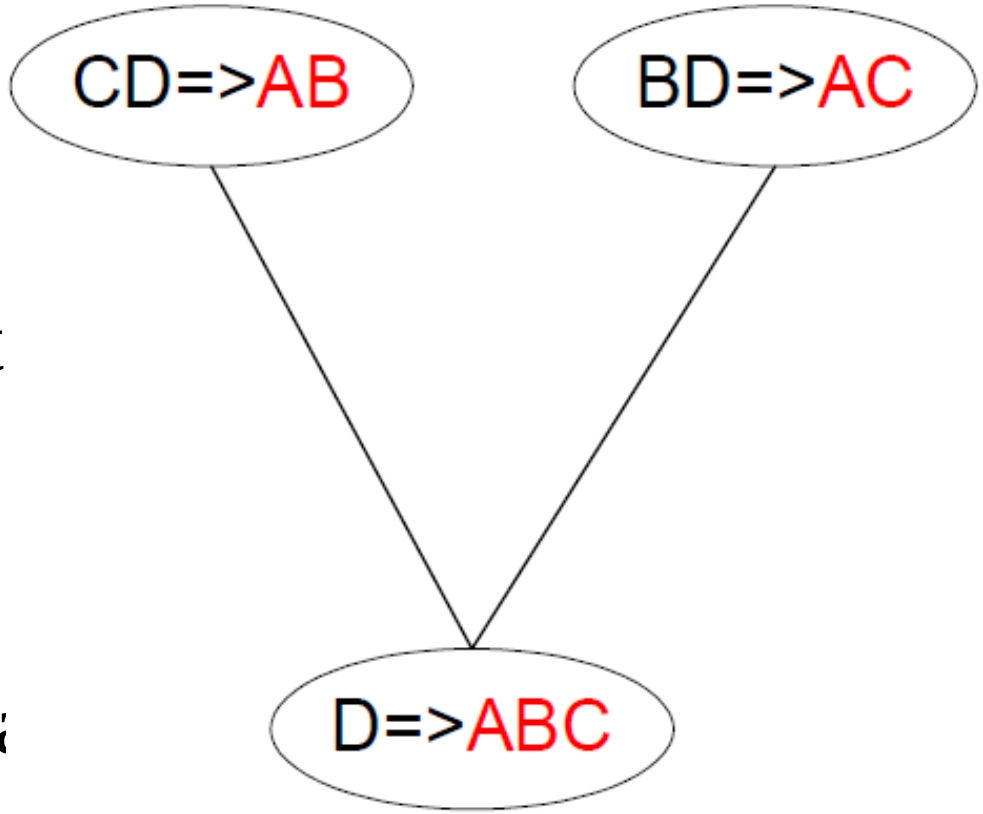
- How to efficiently generate rules from frequent itemsets?
  - In general, confidence does not have an antimonotone property
    $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$
  - But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., 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



Rule generation for apriori algorithm

- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent



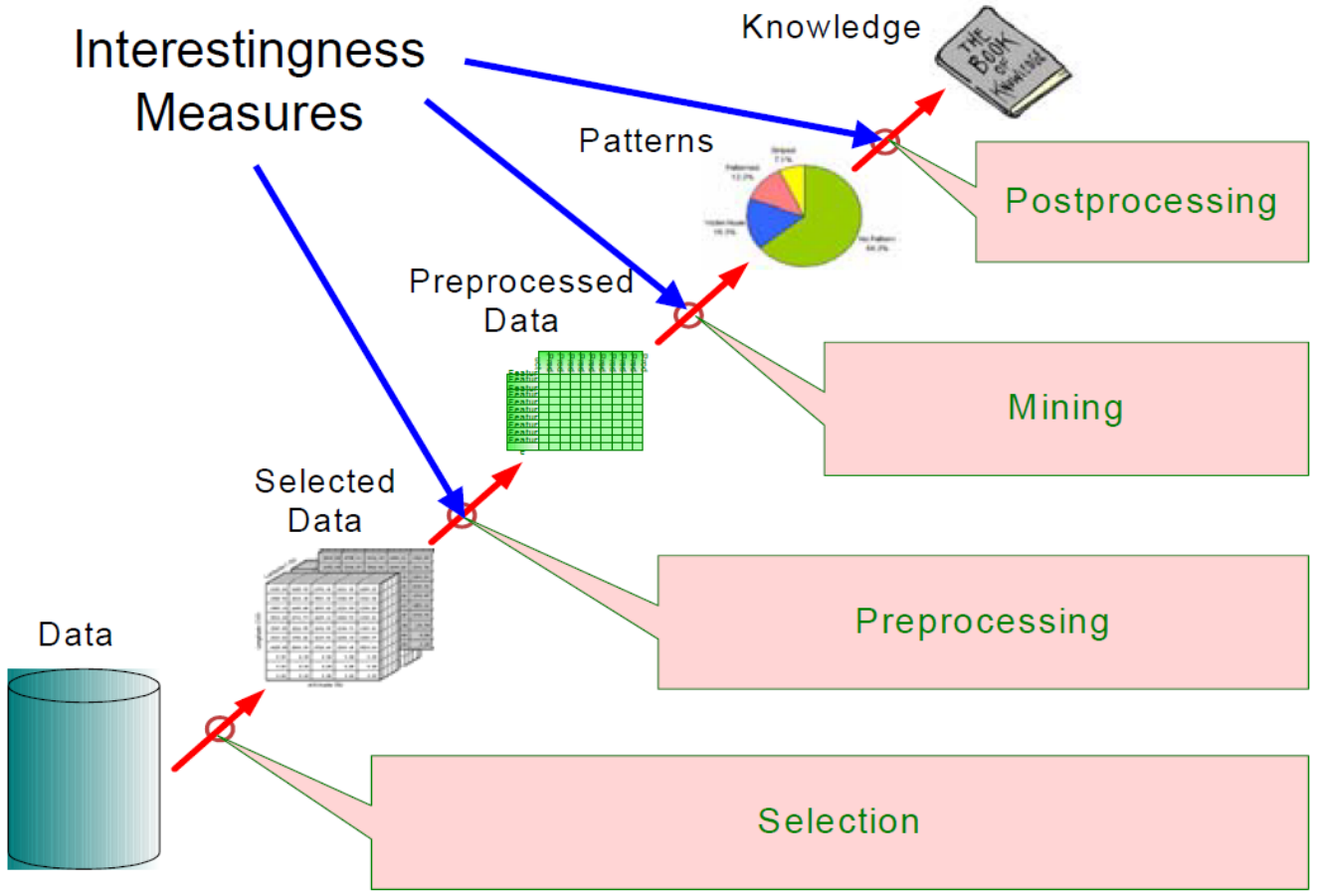
- join(CD=>AB,BD=>AC) would produce the candidat rule D => ABC

- Prune rule D=>ABC if its subset AD=>BC does not ha high confidence

Pattern evaluation

- Association rule algorithms tend to produce too many rules
  - many of them are uninteresting or redundant
  - Redundant if {A,B,C} → {D} and {A,B} → {D} have same support & confidence
- Interestingness measures can be used to prune/rank the derived patterns
- In the original formulation of association rules, support & confidence are the only measures used

Application of interesting measures



Computing interestingness measure

- Given a rule $X \rightarrow Y$, information needed to compute rule interestingness can be obtained from a contingency table

Contingency table for $X \rightarrow Y$

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	$\|T\|$

f_{11}: support of X and Y
f_{10}: support of X and $\overline{Y}$
f_{01}: support of $\overline{X}$ and Y
f_{00}: support of $\overline{X}$ and $\overline{Y}$

Used to define various measures

- support, confidence, lift, Gini, J-measure, etc.

Drawback of confidence

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.9

$\Rightarrow$ Although confidence is high, rule is misleading

$\Rightarrow$ P(Coffee|$\overline{\text{Tea}}$) = 0.9375

Statistical independence

Population of 1000 students

- 600 students know how to swim (S)
- 700 students know how to bike (B)
- 420 students know how to swim and bike (S,B)

- $P(S \wedge B) = 420/1000 = 0.42$
- $P(S) \times P(B) = 0.6 \times 0.7 = 0.42$

- $P(S \wedge B) = P(S) \times P(B)$ => Statistical independence
- $P(S \wedge B) > P(S) \times P(B)$ => Positively correlated
- $P(S \wedge B) < P(S) \times P(B)$ => Negatively correlated

Statistical-based measure

- Measures that take into account statistical dependence

$$Lift = \frac{P(Y \mid X)}{P(Y)}$$

Lift/Interest

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.9

$\Rightarrow$ Lift = 0.75/0.9= 0.8333 (< 1, therefore is negatively associated)

Drawback of lift/interest

	Y	$\bar{Y}$	
X	10	0	10
$\bar{X}$	0	90	90
	10	90	100

$$Lift = \frac{1}{0.1} = 10$$

	Y	$\bar{Y}$	
X	90	0	90
$\bar{X}$	0	10	10
	90	10	100

$$Lift = \frac{1}{0.9} = 1.11$$

Statistical independence:

If P(X,Y)=P(X)P(Y) => Lift = 1

Summary

- Association analysis allows us to derive frequent patterns from a huge amount of data
- Combining with statistical evaluation, interesting frequent patterns can be found and could be utilized further
- Further studies in this direction
  - Collaborative filtering
  - Content-based filtering

Install mlxtend

conda **install** mlxtend *--channel conda-forge*

```
Downloading and Extracting Packages
mlxtend-0.13.0       | 1.2 MB    | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

BIG DATA EXPERIENCE

Load data

```
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

df = pd.read_excel('retail.xlsx')
df.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

KMUTT

BIG DATA EXPERIENCE

Cleaning data

2. Clean the data

1. Remove extra space
2. Drop rows with no InvoiceNo
3. Remove credit card transactions (InvoiceNo with C)

```
df['Description'] = df['Description'].str.strip()
df.dropna(axis=0, subset=['InvoiceNo'], inplace=True)
df['InvoiceNo'] = df['InvoiceNo'].astype('str')
df = df[~df['InvoiceNo'].str.contains('C')]
```

KMUTT

Data preparation

```
basket = (df[df['Country'] =="France"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))
basket
```

.0	0.0	0.0	0.0	0.0	0.0	0.0
.0	0.0	0.0	0.0	0.0	0.0	0.0
.0	0.0	0.0	0.0	0.0	0.0	0.0
.0	0.0	0.0	0.0	0.0	0.0	0.0
.0	0.0	0.0	0.0	0.0	0.0	0.0

One hot encoding

```
def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1

basket_sets = basket.applymap(encode_units)
basket_sets.drop('POSTAGE', inplace=True, axis=1)
basket_sets
```

Description	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND
InvoiceNo					

Frequent itemsets

```
frequent_itemsets = apriori(basket_sets, min_support=0.07,
                            use_colnames=True, n_jobs=-1)
frequent_itemsets.sort_values(by = 'support', ascending=False)
```

	support	itemsets
22	0.188776	(RABBIT NIGHT LIGHT)
26	0.181122	(RED TOADSTOOL LED NIGHT LIGHT)
21	0.170918	(PLASTERS IN TIN WOODLAND ANIMALS)
18	0.168367	(PLASTERS IN TIN CIRCUS PARADE)
30	0.158163	(ROUND SNACK BOXES SET OF4 WOODLAND)

Association rule mining

```
rules = association_rules(frequent_itemsets,
                          metric="lift", min_threshold=1)
rules.sort_values(by='lift',ascending=False)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
2	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.096939	0.094388	0.079082	0.815789	8.642959
3	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.094388	0.096939	0.079082	0.837838	8.642959
4	(ALARM CLOCK BAKELIKE PINK)	(ALARM CLOCK BAKELIKE RED)	0.102041	0.094388	0.073980	0.725000	7.681081

Filter the rules

```
rules[ (rules['lift'] >= 6) &
       (rules['confidence'] >= 0.8) ]
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
2	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.096939	0.094388	0.079082	0.815789	8.642959
3	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.094388	0.096939	0.079082	0.837838	8.642959

Lab

- Use superstore data
- Find association rules

BIG DATA
EXPERIENCE

Thank you

Question?

KMUTT