Data Mining

# Association Analysis

# Example Applications in which Co-Occurrence Matters

- We are often interested in co-occurrence relationships
- Marketing
  1. identify items that are bought together by sufficiently many customers
  2. use this information for marketing or supermarket shelf management purposes

- Inventory Management
  1. identify parts that are often needed together for repairs
  2. use this information to equip your repair vehicles with the right parts

- Usage Mining
  1. identify words that frequently appear together in search queries
  2. use this information to offer auto-completion features to the user

# Outline

1. Correlation Analysis
2. Association Analysis
    1. Frequent Itemset Generation
    2. Rule Generation
    3. Handling Continuous and Categorical Attributes
    4. Interestingness Measures

# 1. Correlation Analysis

– Correlation analysis measures the degree of dependency between two variables

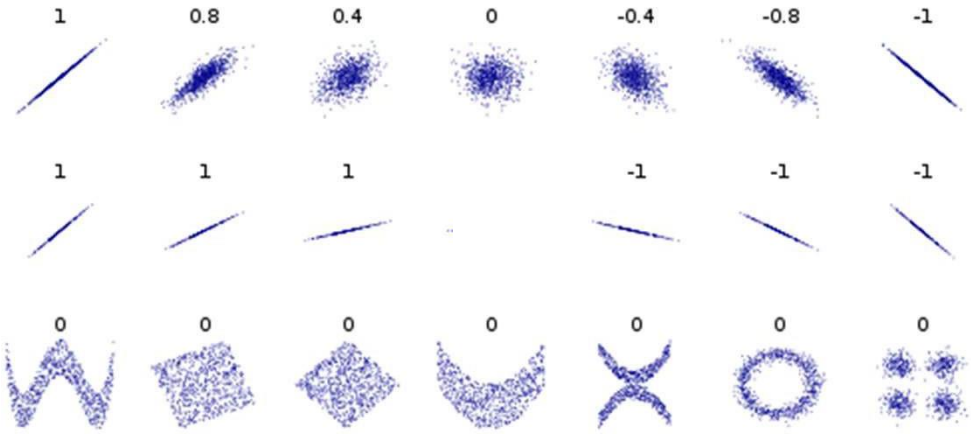
- Continuous variables: Pearson's correlation coefficient (PCC)
- Binary variables: **Phi coefficient**

$$PCC(x,y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2}\sqrt{\sum (y_i - \bar{y})^2}}$$

$$Phi(x,y) = \frac{f_{11} f_{00} - f_{01} f_{10}}{\sqrt{f_{1+} f_{+1} f_{0+} f_{+0}}}$$

– Value range [-1,1]

- 1 : positive correlation
- 0 : variables independent
- -1 : negative correlation



# Correlations between Products in Shopping Baskets

| | P1 | P2 | P3 | P4 | P5 |
|---|---|---|---|---|---|
| Basket 1 | 1 | 1 | 0 | 1 | 1 |
| Basket 2 | 1 | 0 | 0 | 1 | 1 |
| Basket 3 | 1 | 0 | 0 | 0 | 1 |

1 : always bought together
0 : sometimes bought together
-1 : never bought together

Correlations

| Feature 1 | Feature 2 | Correlation | uncorrected p | FDR |
|---|---|---|---|---|
| Cheese | Deli Salads | 0.035 | 3.4659e-18 | 2.00691e-16 |
| Pizza | Pot Scrubbers | 0.029 | 2.11864e-13 | 1.07641e-11 |
| Canned Fruit | Cottage Cheese | 0.029 | 3.62863e-13 | 1.77159e-11 |
| Deli Salads | TV Dinner | 0.029 | 9.22732e-13 | 4.37742e-11 |
| Fashion Magazi... | Pizza | 0.027 | 8.49238e-12 | 3.82157e-10 |
| Sour Cream | Tofu | 0.027 | 1.22618e-11 | 5.37504e-10 |
| Deli Salads | Hard Candy | 0.027 | 2.6884e-11 | 1.13823e-09 |
| Hot Dogs | Sunglasses | 0.026 | 3.42752e-11 | 1.39853e-09 |
| Fashion Magazi... | Pot Scrubbers | 0.026 | 4.25594e-11 | 1.6927e-09 |
| Bologna | Rice | 0.026 | 1.48627e-10 | 5.79425e-09 |

Shortcoming: Measures correlation only between two items but not between multiple items, e.g. {ThinkPad, Cover} → {Minimaus}

# 2. Association Analysis

- Association analysis can find multiple item co-occurrence relationships (descriptive method)
- focuses on occurring items, not absent items
- first algorithms developed in the early 90s at IBM by Agrawal & Srikant
- initially used for shopping basket analysis to find how items purchased by customers are related
- later extended to more complex data structures
  - sequential patterns
  - subgraph patterns
- and other application domains
  - web usage mining, social science, life science

# Association Analysis

**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.**

Shopping 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 |

Examples of Association Rules

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

**Implication means co-occurrence, not causality!**

# Definition: Support and Frequent Itemset

- Itemset
  - collection of one or more items
  - example: {Milk, Bread, Diaper}
  - k-itemset: An itemset that contains k items
- Support count ($\sigma$)
  - frequency of occurrence of an itemset
  - e.g. $\sigma$({Milk, Bread, Diaper}) = 2
- Support (s)
  - fraction of transactions that contain an itemset
  - e.g. s({Milk, Bread, Diaper}) = 2/5 = 0.4
- Frequent Itemset
  - an itemset whose support is greater than or equal to a minimal support (*minsup)* threshold specified by the user

| 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 |

# Definition: Association Rule

- Association Rule
  - an implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
  - an association rule states that when X occurs, Y occurs with certain probability.
  - Example:
    {Milk, Diaper} $\rightarrow$ {Beer}
    Condition Consequent

| 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

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

$$c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \qquad c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

# Main Challenges concerning Association Analysis

1. Mining associations from large amounts of data can be computationally expensive
   - algorithms need to apply smart pruning strategies
2. Algorithms often discover a large number of associations
   - many of them are uninteresting or redundant
   - the user needs to select the subset of the associations that is relevant given her task at hand

# The Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  1. support ≥ *minsup* threshold
  2. confidence ≥ *minconf* threshold
- *minsup* and *minconf* are provided by the user.
- Brute Force Approach:
  1. list all possible association rules
  2. compute the support and confidence for each rule
  3. remove rules that fail the *minsup* and *minconf* thresholds

  ⇒ Computationally prohibitive due to large number of candidates!

# 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 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)

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.

# Mining Association Rules

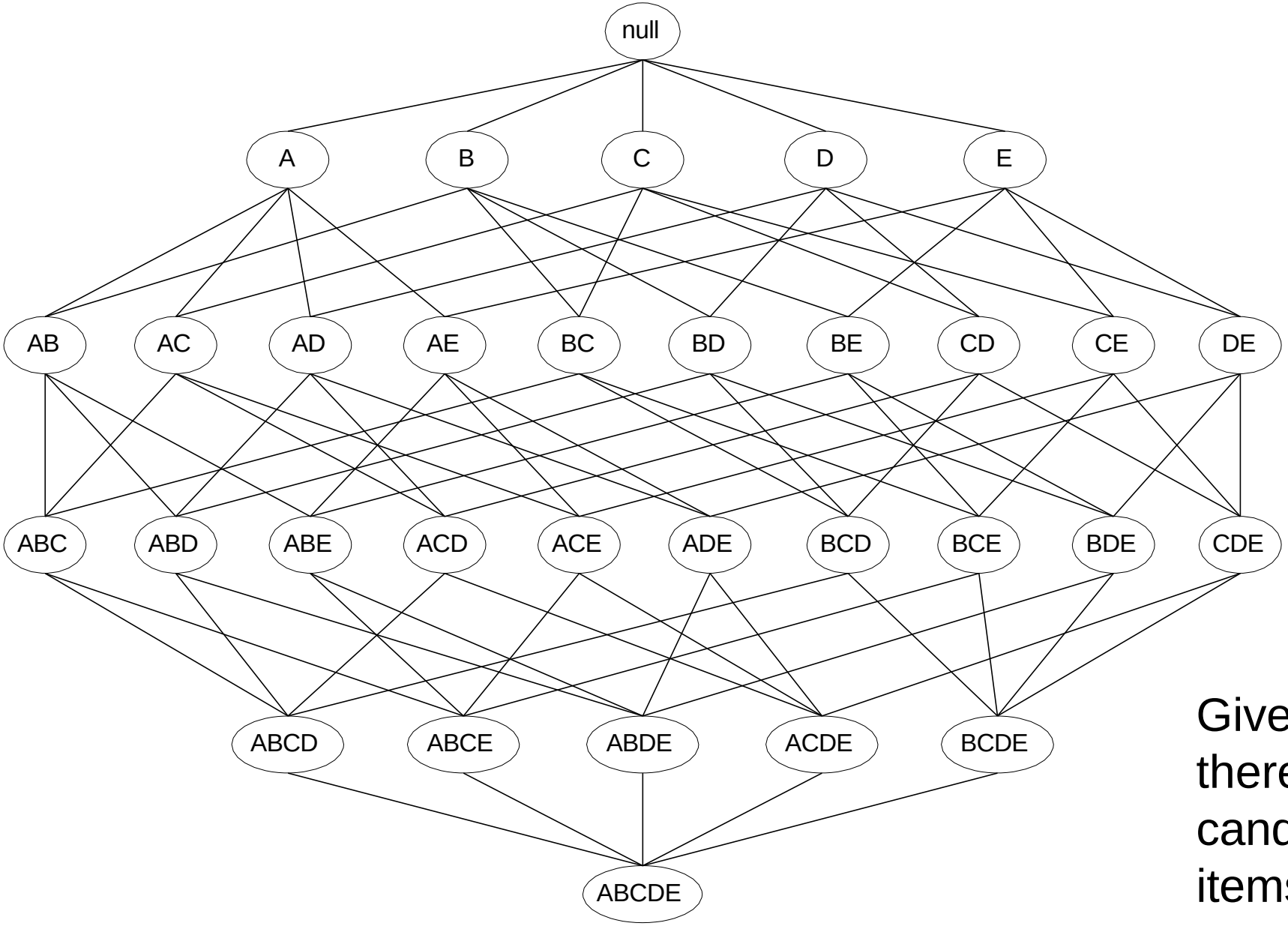
- Two-step approach:

  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

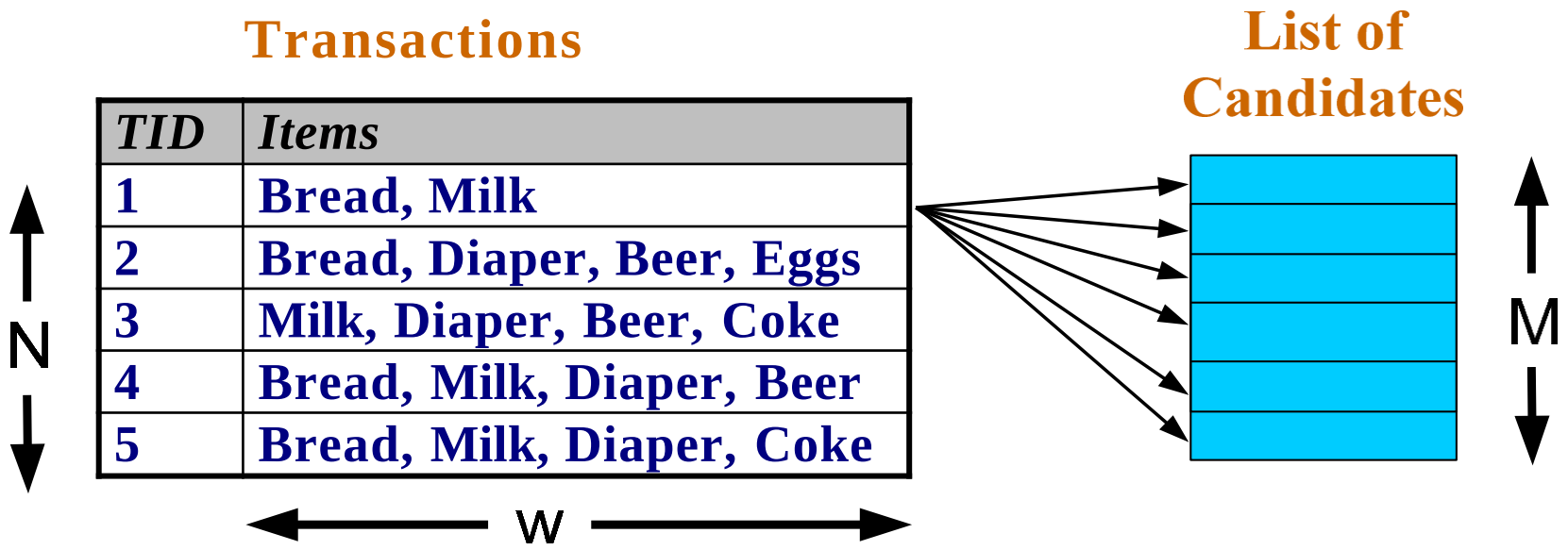
# 2.1 Frequent Itemset Generation




Given d items, there are $2^d$ candidate itemsets!

# Brute Force Approach

- Treat every itemset 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 = $2^d$
- A smarter algorithm is required!

# Example: Brute Force Approach

- Example:
  - Amazon has 10 million books (i.e., Amazon Germany, as of 2011)
- That is $2^{10.000.000}$ possible itemsets
- As a number:
  - $9.04981... \times 10^{3.010.299}$
  - that is: a number with 3 million digits!


- However:
  - most itemsets will not be important at all, e.g., books on Chinese calligraphy, Inuit cooking, and data mining bought together
  - thus, smarter algorithms should be possible
  - intuition for the algorithm: All itemsets containing Inuit cooking are likely infrequent

# Reducing the Number of Candidates

– Apriori Principle

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

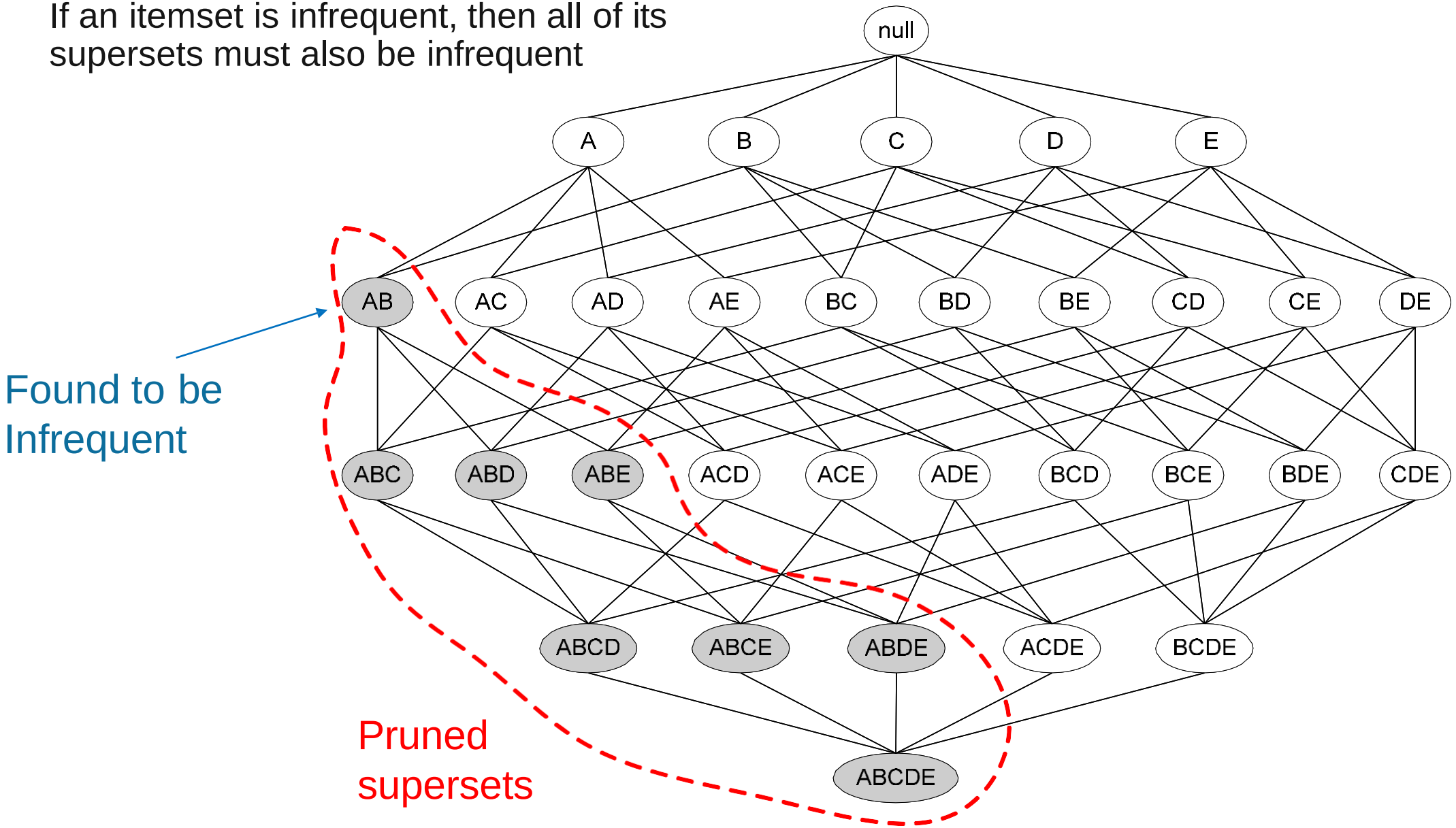
– Apriori principle holds due to the following property of the support measure:

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

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

# Using the Apriori Principle for Pruning

If an itemset is infrequent, then all of its supersets must also be infrequent



# Example: Using the Apriori Principle for Pruning

Minimum Support Count = 3

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

**Items** (1-itemsets)

No need to generate candidates involving Coke or Eggs

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

**Pairs** (2-itemsets)

**Triplets** (3-itemsets)

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

No need to generate candidate {Milk, Diaper, Beer} as count {Milk, Beer} = 2

# The Apriori Algorithm

1. Let k=1
2. Generate frequent itemsets of length 1
3. Repeat until no new frequent itemsets are identified
    1. Generate length (k+1) candidate itemsets from length k frequent itemsets
    2. Prune candidate itemsets that can not be frequent because they contain subsets of length k that are infrequent (Apriori Principle)
    3. Count the support of each candidate by scanning the dataset
    4. Eliminate candidates that are infrequent, leaving only those that are frequent

# Example Application of Frequent Itemsets

1. Take top-k frequent itemsets of size 2 containing item A
2. Rank second item according to
   - profit made by selling item
   - whether you want to reduce number of items B in stock
   - knowledge about customer preferences
3. Offer special price for combination with top-ranked second item

amazon

# 2.2 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.

Example Frequent Itemset:

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

Example Rule:

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

| 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 |

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

# Challenge: Large Number of Candidate Rules

- If {A,B,C,D} is a frequent itemset, then the candidate rules are:

| | | | |
|---|---|---|---|
| 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 $L \rightarrow \varnothing$ and $\varnothing \rightarrow L$)

# Rule Generation

– 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 with respect to the number of items on the right-hand side of the rule

# Explanation

Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

- i.e., “moving elements from left to right” cannot increase confidence

Reason:

$$c(AB \rightarrow C) := \frac{s(ABC)}{s(AB)} \qquad c(A \rightarrow BC) := \frac{s(ABC)}{s(A)}$$

- Due to anti-monotone property of support, we know

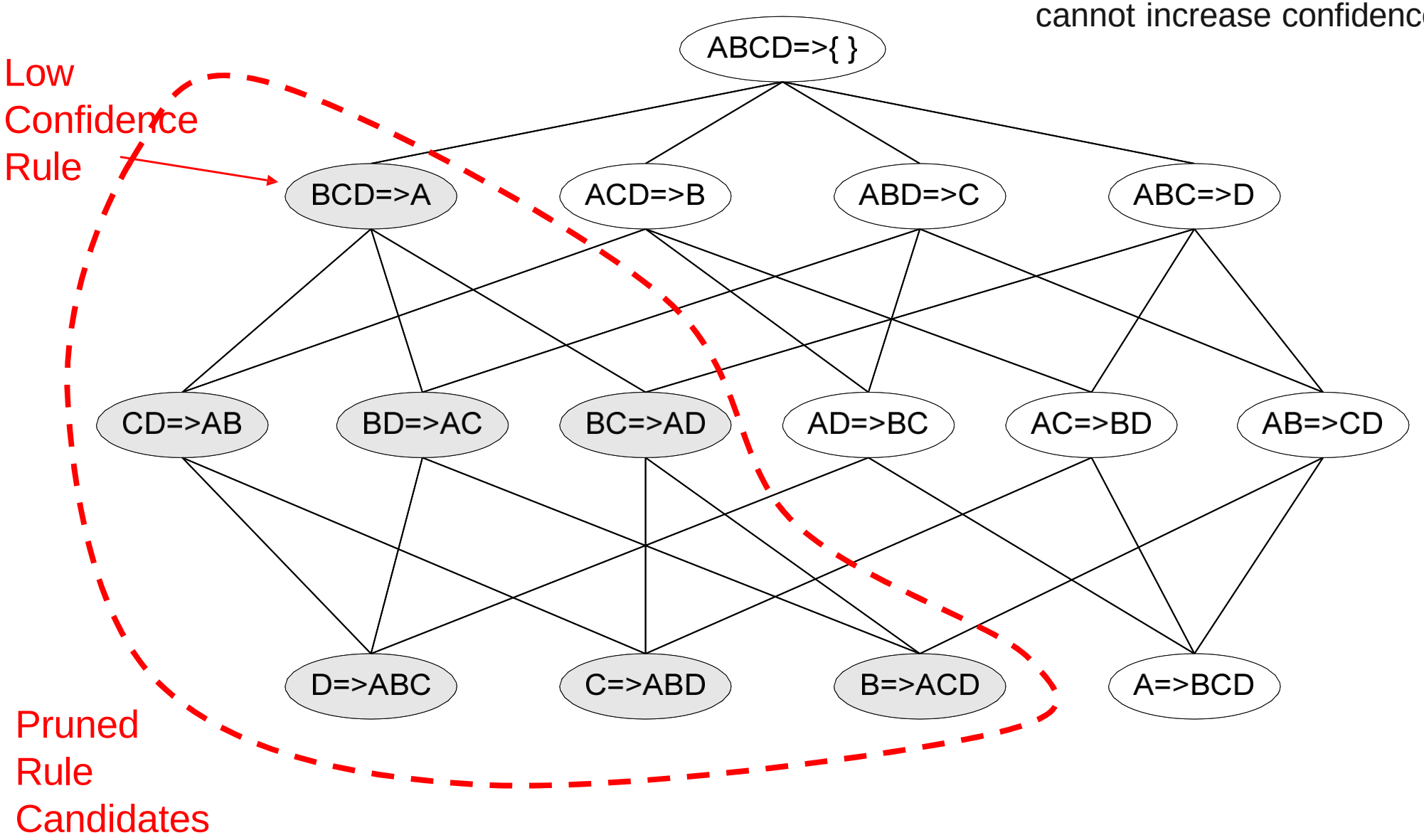
  $s(AB) \leq s(A)$

- Hence

  $c(AB \rightarrow C) \geq C(A \rightarrow BC)$

# Candidate Rule Pruning

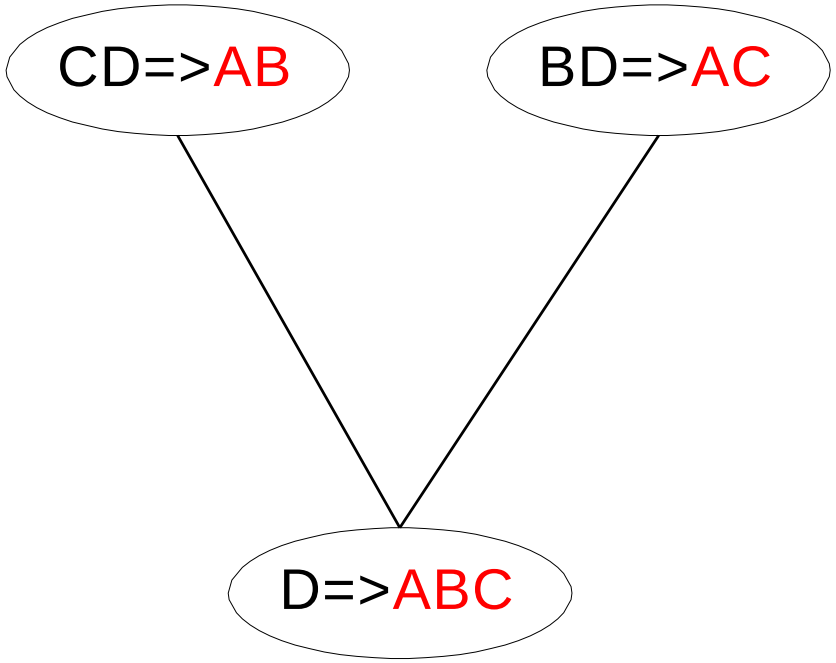
Moving elements from left to right
cannot increase confidence



# Candidate Rule Generation within Apriori Algorithm

– Candidate rule is generated by merging two rules that share the same prefix in the rule consequent (right hand side of rule)

1. join(CD $\rightarrow$ AB, BD $\rightarrow$ AC) would produce the candidate rule D $\rightarrow$ ABC




2. Prune rule D $\rightarrow$ ABC if one of its parent rules does not have high confidence (e.g. AD $\rightarrow$ BC)

– All the required information for confidence computation has already been recorded in itemset generation.

– Thus, there is no need to scan the transaction data *T* any more

# Drawback of Confidence

Contingency table

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

## Association Rule: Tea $\rightarrow$ Coffee

- confidence(Tea $\rightarrow$ Coffee) = 0.75
- but support(Coffee) = 0.9
- although confidence is high, rule is misleading as the fraction of coffee drinkers is higher than the confidence of the rule
- we want confidence(X $\rightarrow$ Y) > support(Y)
- otherwise rule is misleading as X reduces probability of Y

# Lift

- Lift is a measure of the strength of the association between two items, taking into account the frequency of both items in the dataset. It is calculated as the confidence of the association divided by the support of the second item. Lift is used to compare the strength of the association between two items to the expected strength of the association if the items were independent.

$$Lift(\{X\} \rightarrow \{Y\}) = \frac{(Transactions\ containing\ both\ X\ and\ Y)/(Transactions\ containing\ X)}{Fraction\ of\ transactions\ containing\ Y}$$

# Lift

– The lift of an association rule $X \rightarrow Y$ is defined as:

$$Lift = \frac{c(X \rightarrow Y)}{s(Y)}$$

– Confidence normalized by support of consequent

$$Lift(\{X\} \rightarrow \{Y\}) = \frac{(Transactions\ containing\ both\ X\ and\ Y)/(Transactions\ containing\ X)}{Fraction\ of\ transactions\ containing\ Y}$$

– Interpretation

- if lift > 1, then X and Y are positively correlated
- if lift = 1, then X and Y are independent
- if lift < 1, then X and Y are negatively correlated

# Example: Lift

Contingency table

| | Coffee | Coffee | |
|---|---|---|---|
| Tea | 15 | 5 | 20 |
| Tea | 75 | 5 | 80 |
| | 90 | 10 | 100 |

$$Lift = \frac{c(X \rightarrow Y)}{s(Y)}$$

Association Rule: Tea → Coffee

- confidence(Tea → Coffee) = 0.75
- but support(Coffee) = 0.9

Lift(Tea → Coffee) = 0.75/0.9= 0.8333

- lift < 1, therefore is negatively correlated