POLITECNICO DI MILANO

Association Rules

Information Retrieval and Data Mining

Prof. Matteo Matteucci

Learning Unsupervised Rules !?!

Market-Basket Transactions

Bread
Peanuts
Milk
Fruit
Jam

Bread
Jam
Soda
Chips
Milk
Fruit

Steak
Jam
Soda
Chips
Bread

Jam
Soda
Peanuts
Milk
Fruit

Jam
Soda
Chips
Milk
Bread

Fruit
Soda
Chips
Milk

Fruit
Soda
Peanuts
Milk

Fruit
Peanuts
Cheese
Yogurt

What is Frequent Pattern Mining?

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- Motivation: Finding inherent regularities in data
  - What products were often purchased together? Beer and diapers?!
  - What are the subsequent purchases after a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, DNA sequence analysis, etc.

What is Association Rule Mining?

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

Examples

$\{bread\} \Rightarrow \{milk\}$

$\{soda\} \Rightarrow \{chips\}$

$\{bread\} \Rightarrow \{jam\}$

- 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

Frequent Itemset

- **Itemset**
  - A collection of one or more items, e.g., {milk, bread, jam}
  - k-itemset, an itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - σ({Milk, Bread}) = 3
    σ({Soda, Chips}) = 4
- **Support**
  - Fraction of transactions that contain an itemset
  - s({Milk, Bread}) = 3/8
    s({Soda, Chips}) = 4/8
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a minsup threshold

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

What is An Association Rule?

- Implication of the form X ⇒ Y, where X and Y are itemsets
  - Example: {bread} ⇒ {milk}
- Rule Evaluation Metrics, Suppor & Confidence
  - Support (s)
    - Fraction of transactions that contain both X and Y

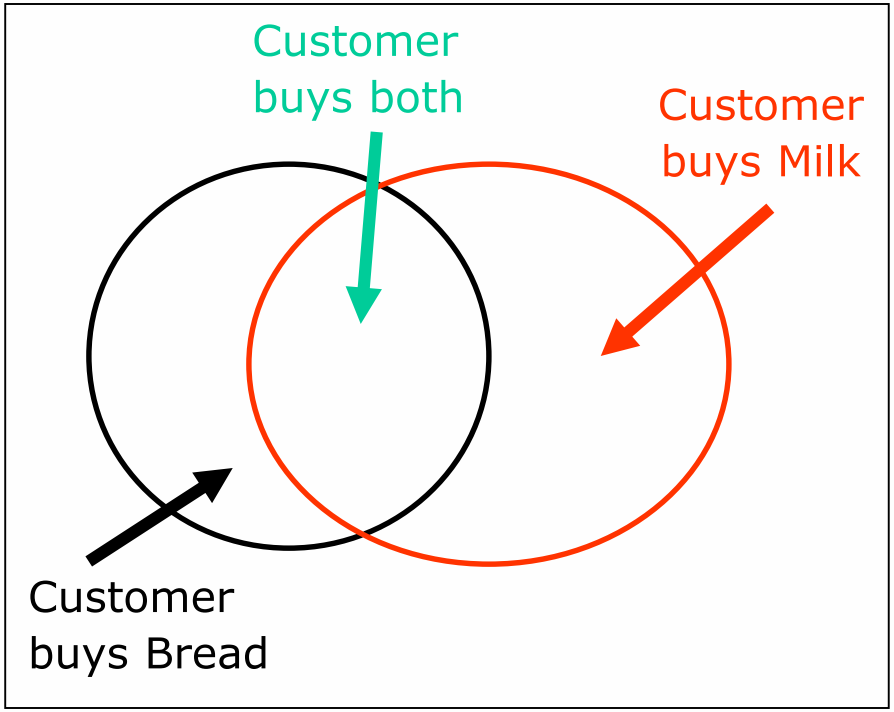
$$s = \frac{\sigma(\{\text{Bread}, \text{Milk}\})}{\#\text{ of transactions}} = 0.38$$

  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

$$c = \frac{\sigma(\{\text{Bread}, \text{Milk}\})}{\sigma(\{\text{Bread}\})} = 0.75$$

Support and Confidence Meaning

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt



Support (s) = P(X,Y)

$$s = \frac{\sigma(\{\text{Bread}, \text{Milk}\})}{\#\text{ of transactions}} = 0.38$$

Confidence (c) = P(X,Y)/P(X)
= P(Y|X)

$$c = \frac{\sigma(\{\text{Bread}, \text{Milk}\})}{\sigma(\{\text{Bread}\})} = 0.75$$

What is the Goal?

- 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
- Brute-force approach is computationally prohibitive!

Mining Association Rules

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

{Bread, Jam} $\Rightarrow$ {Milk}: s=3/8 c=3/4
{Bread, Milk} $\Rightarrow$ {Jam}: s=3/8 c=3/3
{Milk, Jam} $\Rightarrow$ {Bread}: s=3/8 c=3/3
{Bread} $\Rightarrow$ {Milk, Jam}: s=3/8 c=3/4
{Jam} $\Rightarrow$ {Bread, Milk}: s=3/8 c=3/5
{Milk} $\Rightarrow$ {Bread, Jam}: s=3/8 c=3/6

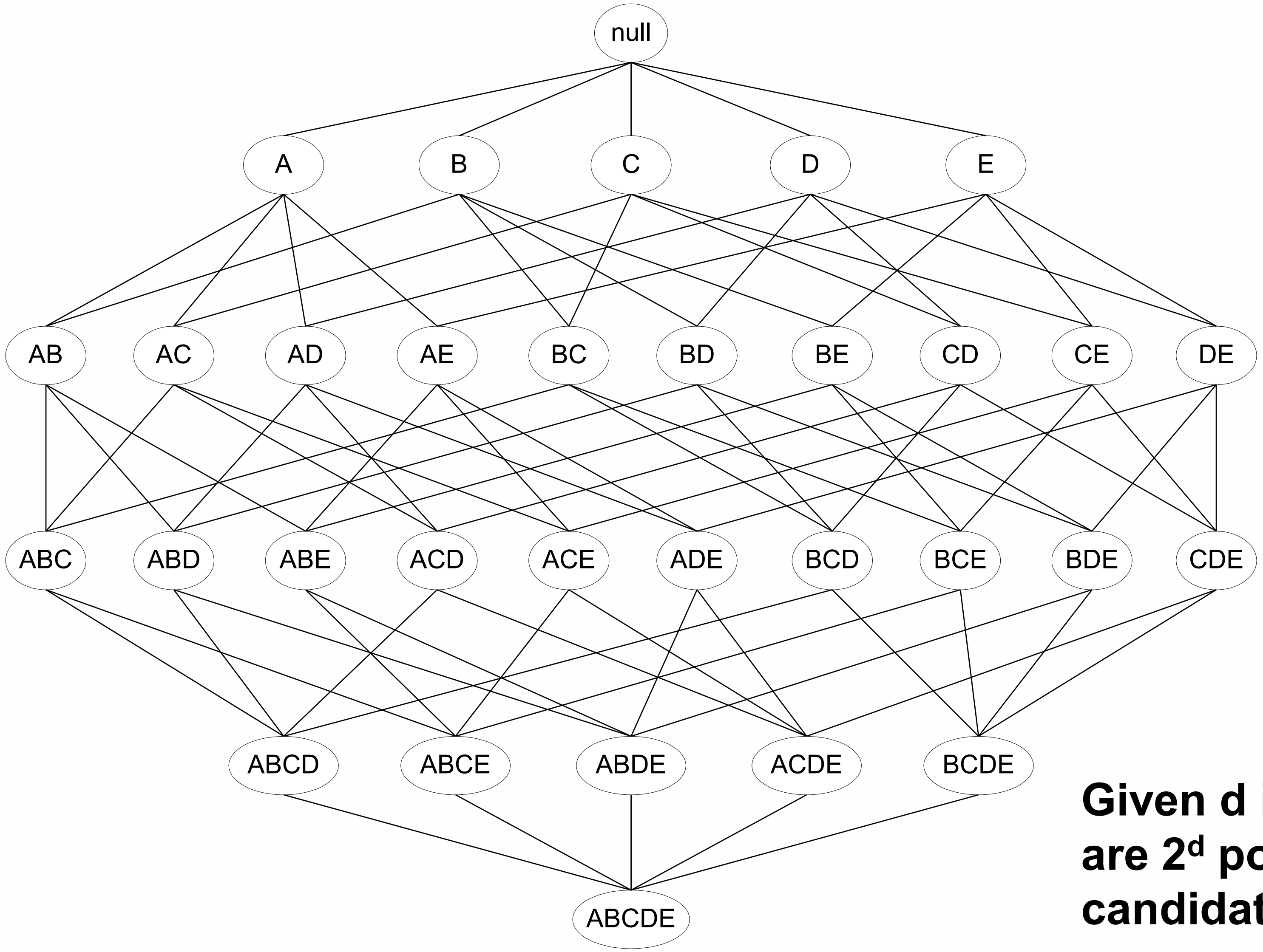
- All bove rules are binary partitions of the same itemset: {Milk, Bread, Jam}
- Rules originating from the same itemset have identical support but can have different confidence
- Decouple the support and confidence requirements!

Mining Association Rules in Two Steps

1. Frequent Itemset Generation
   - Generate all itemsets whose support $\geq$ minsup
2. Rule Generation
   - Generate high confidence rules from frequent itemset
   - Each rule is a binary partitioning of a frequent itemset

However frequent itemset generation is computationally expensive!

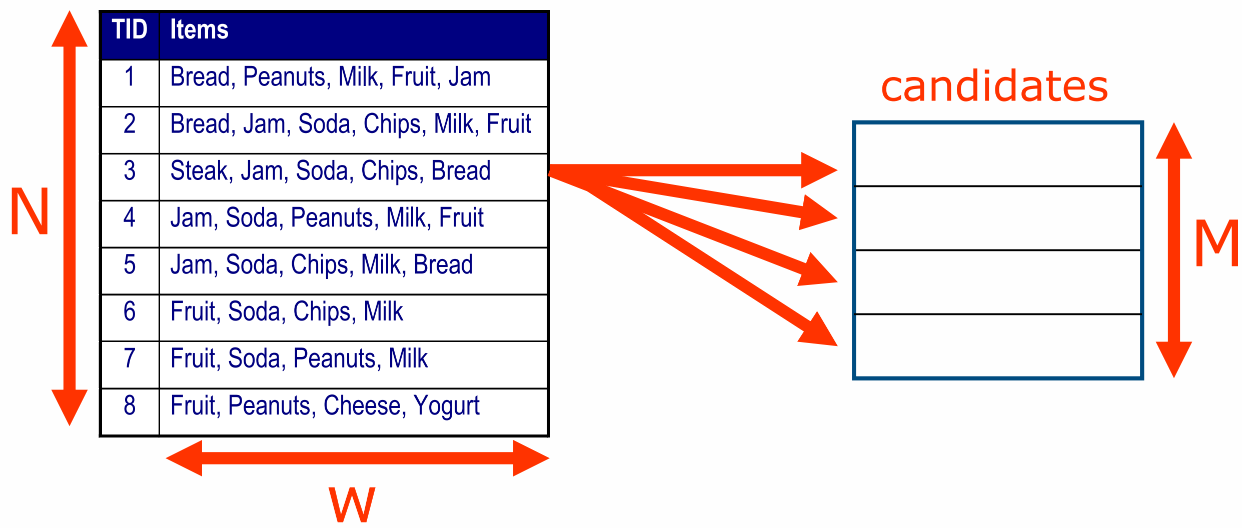
Frequent Itemset Generation



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

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

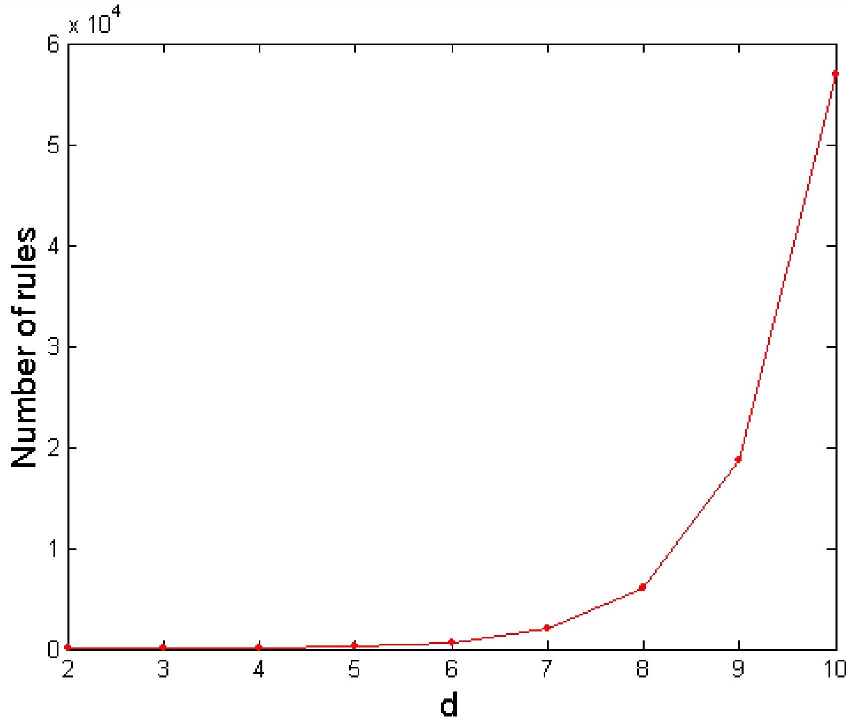


  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$

Computational Complexity

- Given d unique items:
  - Number of itemsets: 2^d
  - Number of possible association rules: $\sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right] = 3^d - 2^{d+1} + 1$

- For d=6, there are 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 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 the Number of Candidates

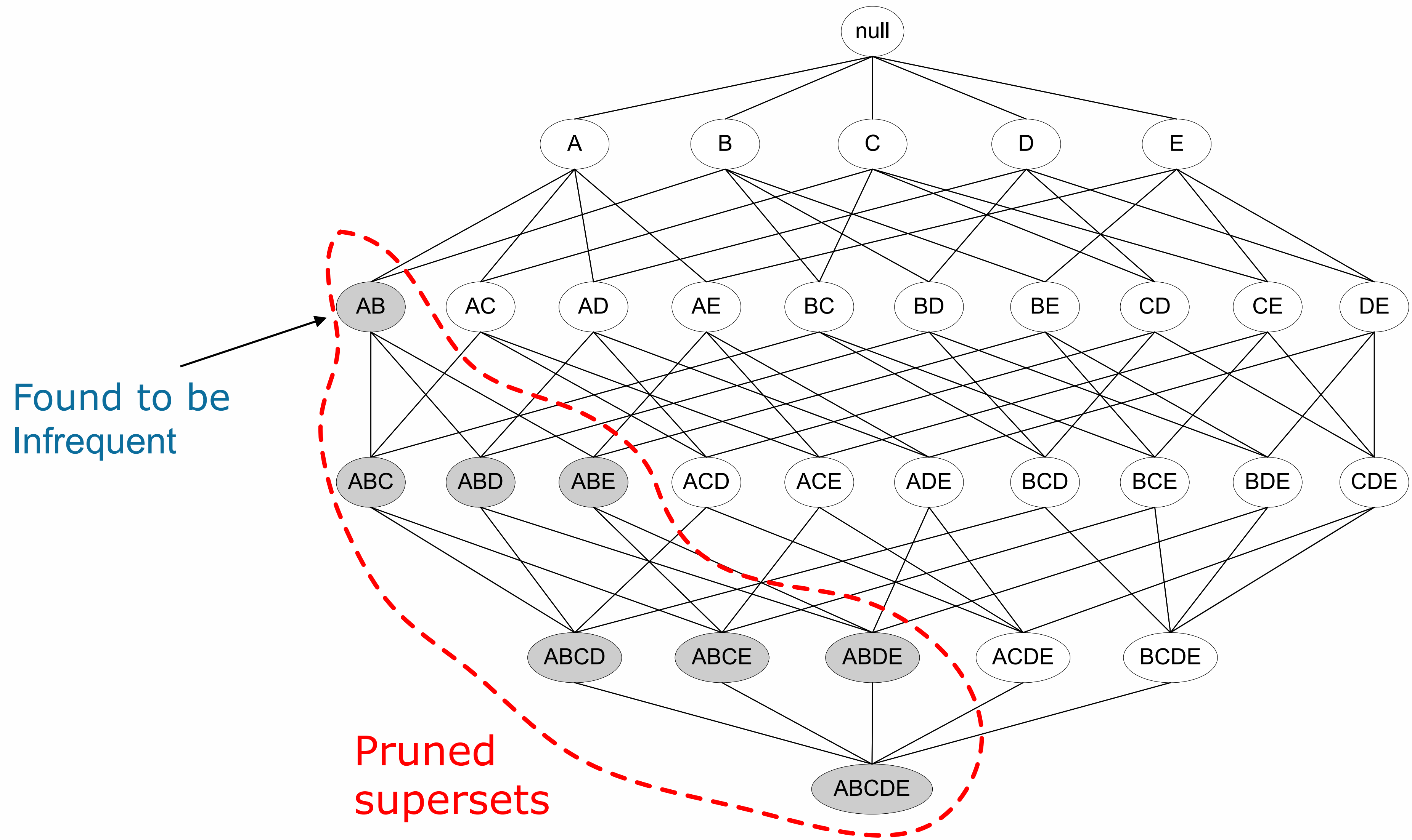
- Apriori principle
  - If an itemset is frequent, then all of its subsets must also be frequent

- 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

Illustrating Apriori Principle



How does the Apriori principle work?

Items (1-itemsets)

Item	Count
Bread	4
Peanuts	4
Milk	6
Fruit	6
Jam	5
Soda	6
Chips	4
~~Steak~~	~~1~~
~~Cheese~~	~~1~~
~~Yogurt~~	~~1~~

Minimum Support = 4

2-itemsets

2-Itemset	Count
Bread, Jam	4
Peanuts, Fruit	4
Milk, Fruit	5
Milk, Jam	4
Milk, Soda	5
Fruit, Soda	4
Jam, Soda	4
Soda, Chips	4

3-itemsets

3-Itemset	Count
Milk, Fruit, Soda	4

The Apriori Algorithm Idea

- 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 each candidate support by scanning the DB
  - Eliminate candidates that are infrequent, leaving only those that are frequent

Important Details on Apriori Algorithm

- 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*}

C_k: Candidate itemset of size k

L_k : frequent itemset of size k

L_1 = {frequent items};

for (k = 1; L_k !=$\emptyset$; k++) **do begin**

    C_{k+1} = candidates generated from L_k;

    for each transaction t in database do

        increment the count of all candidates in C_{k+1}

        that are contained in t

    L_{k+1} = candidates in C_{k+1} with min_support

    end

return $\cup_k L_k$;

An Example of Frequent Itemset

$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1st scan →

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

→ L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

← 2nd scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

← L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3rd scan → L_3

Itemset	sup
{B, C, E}	2

Strategies to Improve Apriori Efficiency

- Hash-based itemset counting:
  - A k-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- Transaction reduction:
  - A transaction that does not contain any frequent k-itemset is useless in subsequent scans
- Partitioning:
  - Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
- Sampling:
  - mining on a subset of given data, lower support threshold + a method to determine the completeness
- Dynamic itemset counting:
  - add new candidate itemsets only when all of their subsets are estimated to be frequent

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 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 $\varnothing \rightarrow L$)

How to efficiently generate rules from frequent itemsets?

- Confidence does not have an anti-monotone property

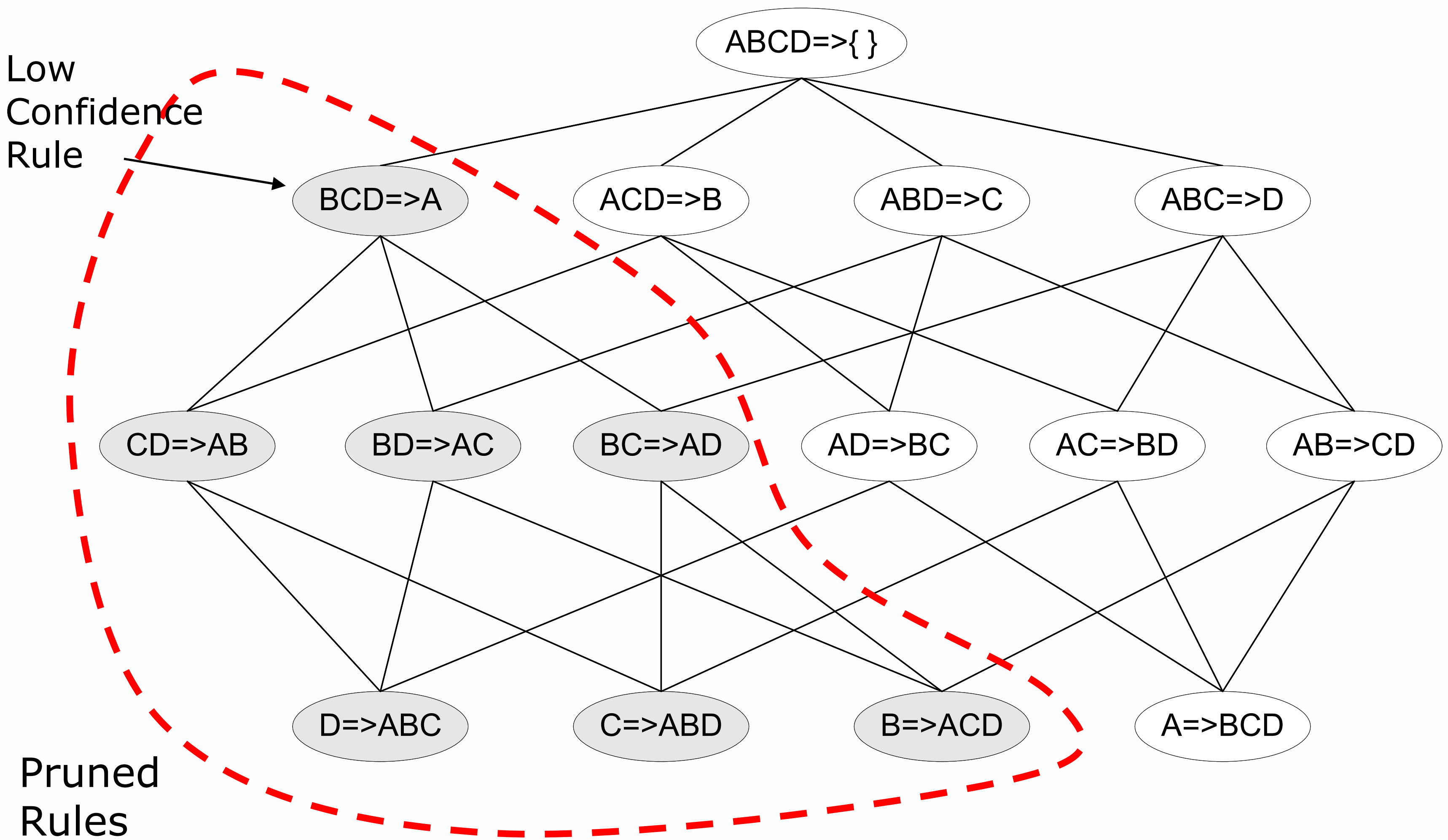
  $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

- But confidence of rules generated from the same itemset has an anti-monotone property

  $L = \{A,B,C,D\}$: $c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$

- Confidence is anti-monotone with respect to the number of items on the right hand side of the rule

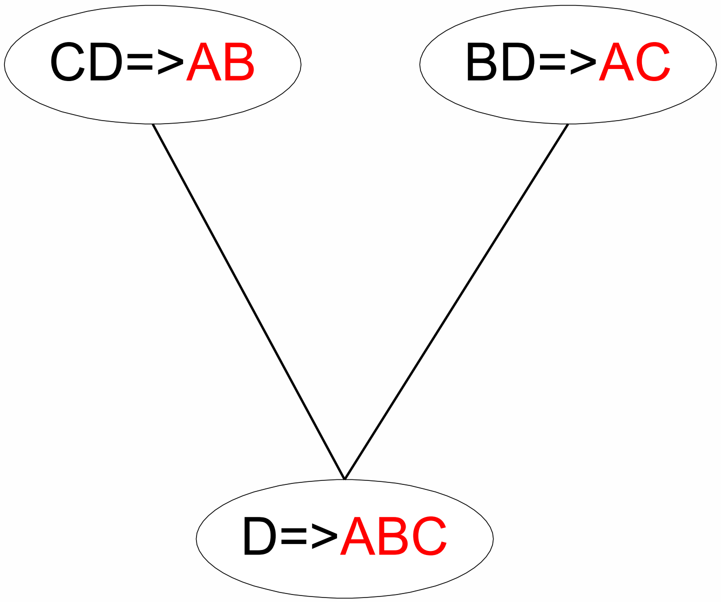
Rule Generation for Apriori Algorithm



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



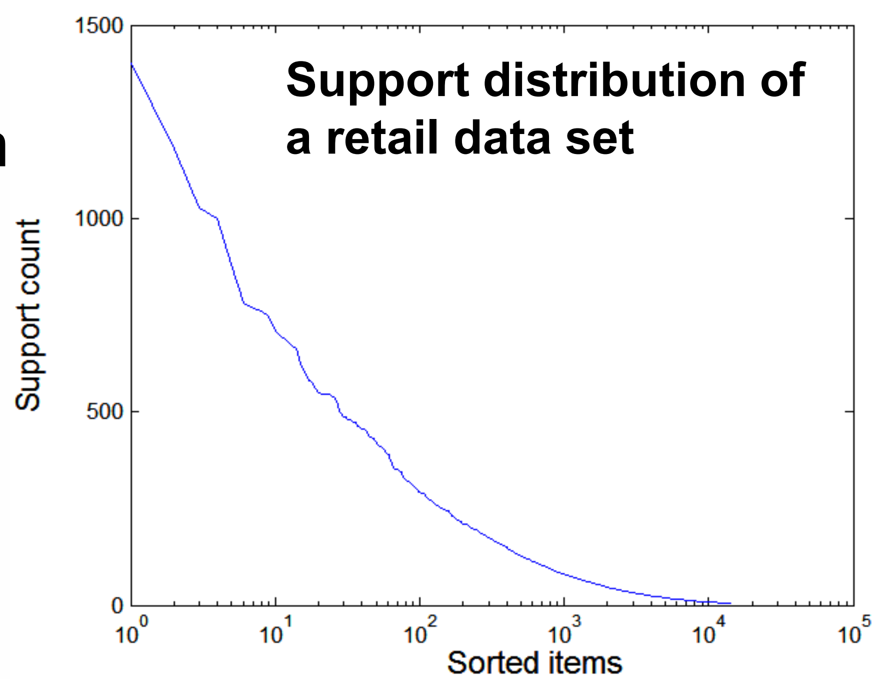
Example with Weka: Formatting the data

```
%
% Example of market basket data
%
@relation 'basket'
@attribute Bread {1}
@attribute Peanuts {1}
@attribute Milk {1}
@attribute Fruit {1}
@attribute Jam  {1}
@attribute Soda  {1}
@attribute Chips {1}
@attribute Steak {1}
@attribute Cheese {1}
@attribute Yogurt {1}
```

```
@data
1,1,1,1,1,?,?,?,?,?
1,?,1,1,1,1,1,?,?,?
1,?,?,?,1,1,1,1,?,?
?,1,1,1,1,1,?,?,?,?
1,?,1,?,1,1,1,?,?,?
?,?,1,1,?,1,1,?,?,?
?,1,1,1,?,1,?,?,?,?
?,1,?,1,?,?,?,?,1,1
```

Effect of Support Distribution

- Many real data sets have skewed support distribution



- If minsup is set too high, we could miss itemsets involving interesting rare (expensive)items
- If minsup is set too low, apriori becomes computationally expensive and the number of itemsets very large
- A single minimum support threshold may not be effective

Are there Interesting/Useful Rules?

Anything that is interesting happens significantly more than you would expect by chance.

Example: basic statistical analysis of basket data may show that 10% of baskets contain bread, and 4% of baskets contain washing-up powder.

What is the probability of a basket containing both bread *and* washing-up powder? The laws of probability say:

- if you choose a basket at random:
  - There is a probability 0.1 that it contains bread.
  - There is a probability 0.04 that it contains washing-up powder.
- If these two are independent:
  - There is a probability 0.1 * 0.04 = 0.004 it contains both

Are there Interesting/Useful Rules?

Anything that is <u>interesting</u> happens significantly more than you would expect by chance.

<u>Example:</u> basic statistical analysis of basket data may show that 10% of baskets contain bread, and 4% of baskets contain washing-up powder.

We have a prior expectation that just 4 baskets in 1000 should contain **both** bread and washing up powder:

- If we discover that really it is 20 in 1000 baskets, then we will be very surprised.
- Something is going on in shoppers' minds: bread and washing-up powder are connected in some way.
- There may be ways to exploit this discovery …

Another Example

ID	apples	beer	cheese	dates	eggs	fish	glue	honey	cream
1	1	1		1			1	1	
2			1	1	1				
3		1	1			1			
4		1				1			1
5					1		1		
6						1			1
7	1			1				1	
8						1			1
9			1		1				
10		1					1		
11					1		1		
12	1								
13			1			1			
14			1			1			
15								1	1
16				1					
17	1					1			
18	1	1	1	1				1	
19	1	1		1			1	1	
20					1				

Sequential Pattern Mining

- Association rules do not consider order of transactions
- In many applications such orderings are significant
  - In market basket analysis, it is interesting to know whether people buy some items in sequence, e.g., buying bed first and then bed sheets later
  - In Web usage mining, it is useful to find navigational patterns of users in a Web site from sequences of page visits of users
  - …

Examples of Sequence

- Typical Web sequence
  - < {Homepage} {Electronics} {Digital Cameras} {Canon Digital Camera} {Shopping Cart} {Order Confirmation} {Return to Shopping} >
- Sequence of events causing the accident at 3-mile Island:
  - <{clogged resin} {outlet valve closure} {loss of feedwater} {condenser polisher outlet valve shut} {booster pumps trip} {main waterpump trips} {main turbine trips} {reactor pressure increases}>
- Sequence of books checked out at a library:
  - <{Fellowship of the Ring} {The Two Towers} {Return of the King}>

Basic concepts

- Let I = {i1, i2, …, im} be a set of items.
  - A sequence is an ordered list of **itemsets**.
  - We denote a sequence S by ⟨A1, A2, …, Ar⟩, where Ai is an itemset, also called an element of S.
  - An element (or an itemset) of a sequence is denoted by {a1, a2, …, ak}, where aj $\in$ I is an item.
- We assume without loss of generality that items in an element of a sequence are in lexicographic order

Basic concepts … continued

- The size of a sequence is the number of elements (***itemsets***) in the sequence
- The length of a sequence is the number of items in the sequence (a sequence of length k is called k-sequence)

- Given s1 = ⟨a1 a2 … ar⟩ and s2 = ⟨b1 b2 … bv⟩
  - s1 is a subsequence of s2, or s2 is a supersequence of s1, if there exist integers

    $1 \leq j_1 < j_2 < \ldots < j_{r-1} < jr \leq v$ such that

    $a1 \subseteq bj1, a2 \subseteq bj2, \ldots, ar \subseteq bjr.$
  - We also say that s2 contains s1.

Example

Let I = {1, 2, 3, 4, 5, 6, 7, 8, 9}.

- The size of the sequence $\langle\{3\}\{4, 5\}\{8\}\rangle$ is 3, and the length of the sequence is 4.

- The sequence
  - $\langle\{3\}\{4, 5\}\{8\}\rangle$ is contained in (or is a subsequence of)
  - $\langle\{6\}\ \{3, 7\}\{9\}\{4, 5, 8\}\{3, 8\}\rangle$

  - In fact, $\{3\} \subseteq \{3, 7\}$, $\{4, 5\} \subseteq \{4, 5, 8\}$, and $\{8\} \subseteq \{3, 8\}$

- However, $\langle\{3\}\{8\}\rangle$ is not contained in $\langle\{3, 8\}\rangle$ or vice versa

Goal of Sequential Pattern Mining

- The input is a set S of sequences (or sequence database)
- The problem is to mine all sequences that have a user-specified minimum support
  - Each such sequence is called a frequent sequence, or a sequential pattern
  - The support for a sequence is the fraction of total data sequences in S that contains this sequence
- Apriori property for sequential patterns
  - If a sequence S is not frequent, then none of the super-sequences of S is frequent
  - For instance, if <h b> is infrequent so do <h a b> and <{ah} {b}>

Terminology

- Itemset
  - Non-empty set of items
  - Each itemset is mapped to an integer
- Sequence
  - Ordered list of itemsets
- Support for a Sequence
  - Fraction of sequence database supporting a sequence
- Maximal Sequence
  - A sequence not contained in any other sequence
- Large Sequence
  - Sequence that meets minisup

Generalized Sequential Pattern (GSP)

- Step 1:
  - Make the first pass over the sequence database D to yield all the 1-element frequent sequences
- Step 2: Repeat until no new frequent sequences are found
  - Candidate Generation:
    - Merge pairs of frequent subsequences found in the (k-1)th pass to generate candidates with k items
  - Candidate Pruning:
    - Prune candidate k-sequences that contain infrequent (k-1)-subsequences
  - Support Counting:
    - Make a new pass over the sequence database D to find the support for these candidate sequences
  - Candidate Elimination:
    - Eliminate candidate k-sequences whose actual support is less than minsup

Example

Table 1. A set of transactions sorted by customer ID and transaction time

Customer ID	Transaction Time	Transaction (items bought)
1	July 20, 2005	30
1	July 25, 2005	90
2	July 9, 2005	10, 20
2	July 14, 2005	30
2	July 20, 2005	40, 60, 70
3	July 25, 2005	30, 50, 70
4	July 25, 2005	30
4	July 29, 2005	40, 70
4	August 2, 2005	90
5	July 12, 2005	90

Example (cond)

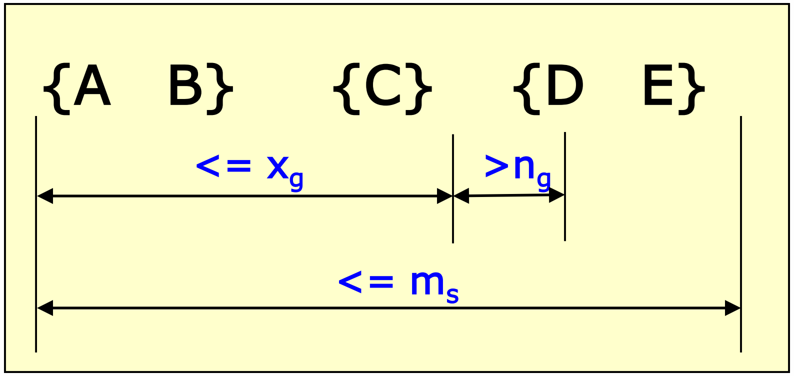
Table 2. Data sequences produced from the transaction database in Table 1.

Customer ID	Data Sequence
1	⟨{30} {90}⟩
2	⟨{10, 20} {30} {40, 60, 70}⟩
3	⟨{30, 50, 70}⟩
4	⟨{30} {40, 70} {90}⟩
5	⟨{90}⟩

Table 3. The final output sequential patterns

	Sequential Patterns with Support ≥ 25%
1-sequences	⟨{30}⟩, ⟨{40}⟩, ⟨{70}⟩, ⟨{90}⟩
2-sequences	⟨{30} {40}⟩, ⟨{30} {70}⟩, ⟨{30} {90}⟩, ⟨{40, 70}⟩
3-sequences	⟨{30} {40, 70}⟩

Timing Constraints



x_g: **max-gap**

n_g: **min-gap**

m_s: **maximum span**

$x_g = 2$, $n_g = 0$, $m_s = 4$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {4,7} {4,5} {8} >	< {6} {5} >	Yes
< {1} {2} {3} {4} {5}>	< {1} {4} >	No
< {1} {2,3} {3,4} {4,5}>	< {2} {3} {5} >	Yes
< {1,2} {3} {2,3} {3,4} {2,4} {4,5}>	< {1,2} {5} >	No