Association Rules Mining

Lecturer: Dr. Nguyen Ngoc Thao
Department of Computer Science, FIT, HCMUS

Many slides adapted from Bing Liu

Outline

- What is Association Rule Mining?
- Basic Concepts of Association Rules
- The Apriori Algorithm
- The FP-Growth Algorithm
- Which Patterns Are Interesting?
- Maximal Frequent Patterns and Closed Frequent Patterns
- Summary

Outline

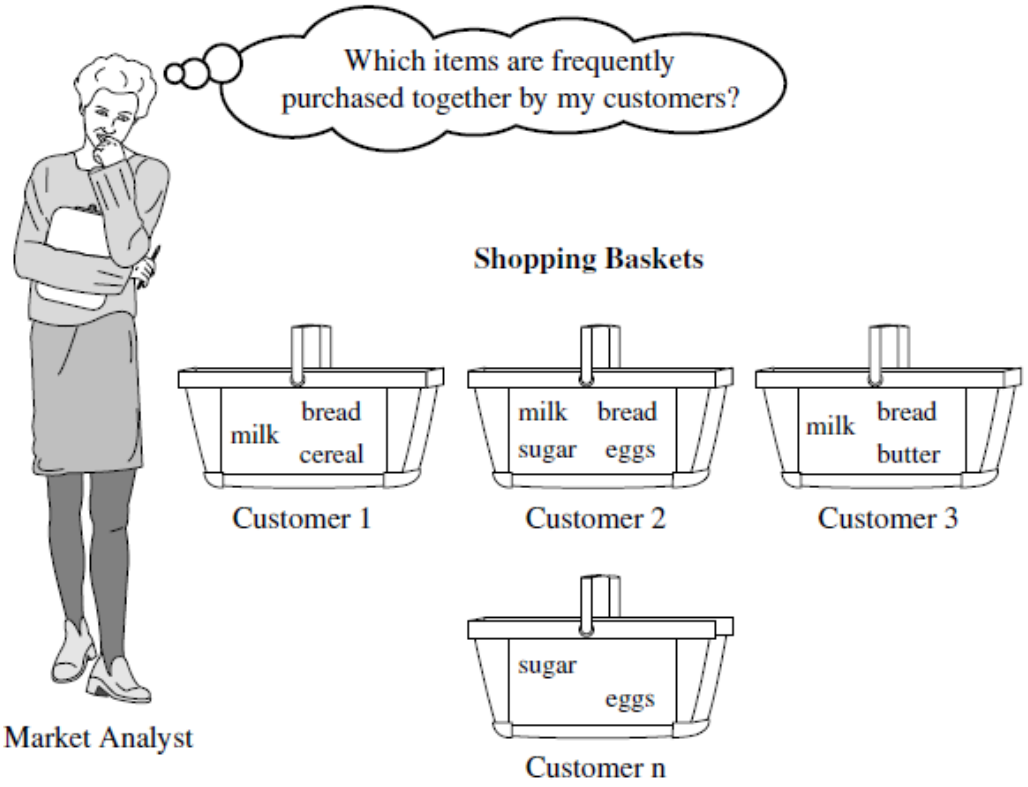
- **What is Association Rule Mining?**
- Basic Concepts of Association Rules
- The Apriori Algorithm
- The FP-Growth Algorithm
- Which Patterns Are Interesting?
- Maximal Frequent Patterns and Closed Frequent Patterns
- Summary

Association Rule Mining

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- Association rule mining generate (strong) association rules from frequent patterns that represent inherent regularities in data
  - What products were often purchased together?
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications:
  - Market basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (clickstream) analysis, and DNA sequence analysis, etc.

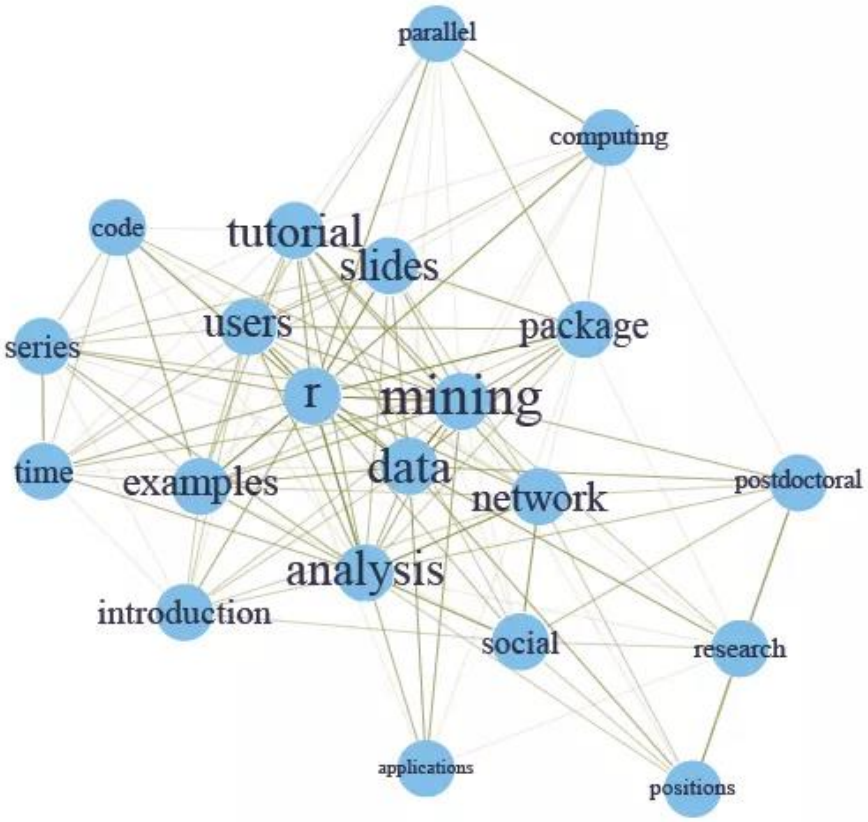
Applications: Market Basket Data Analysis

- Discover how items purchased by customers in a supermarket (or a store) are associated
  - Plan marketing or advertising strategies, design new catalogs, and design different store layouts



Applications: Finding Linguistic Patterns

- Discover the co-occurrence relationships of words (or linguistic patterns) in text documents
  - Group documents according to topics, and infer the writing style



Applications: Web Usage Mining

- Discover patterns in clickstreams, user transactions and other user interactions with Web resources
  - Provide personalization to visitors, effectively arrange the site content and optimize the space for storing

Association Rule Mining

- First introduced by Agrawal et al in 1993
- Initially used for Market Basket Analysis to find how items purchased by customers are related.

  Bread $\rightarrow$ Milk [sup = 5%, conf = 100%]

- It is an important data mining model studied extensively by the database and data mining community
- Assume all data are categorical
- No good algorithm for numeric data

Outline

- What is Association Rule Mining?
- **Basic Concepts of Association Rules**
- The Apriori Algorithm
- The FP-Growth Algorithm
- Which Patterns Are Interesting?
- Maximal Frequent Patterns and Closed Frequent Patterns
- Summary

The Model: Data

- $I = \{i_1, i_2, \ldots, i_m\}$: a set of items
- Transaction t: a set of items and $t \subset I$
- Transaction database T: a set of transactions, $T = \{t_1, t_2, \ldots, t_n\}$

Transaction Data: Market Basket Data

- Market basket transactions:

TID	Transaction
t1	{Bread, Cheese, Milk}
t2	{Milk, Bread, Yogurt}
…	…
tn	{Biscuit, Eggs, Milk}

- Concepts:
  - An item: an item/article in a basket
  - I: the set of all items sold in the store
  - A transaction: a set of items purchased in a basket by a customer; it may have TID (transaction ID)
  - A transactional dataset: a set of transactions
- This is a simplistic view of shopping baskets
  - The quantity and price of each item, for example, are not considered in the model

Transaction Data: A Set of Documents

- A text document dataset

  doc1: Student, Teach, School

  doc2: Student, School

  doc3: Teach, School, City, Game

  doc4: Baseball, Basketball

  doc5: Basketball, Player, Spectator

  doc6: Baseball, Coach, Game, Team

  doc7: Basketball, Team, City, Game

- Each document is treated as a “bag” of keywords (transaction), without considering word sequence and the number of occurrences of each word

The Model: Rules

- A transaction t contains X, a set of items (itemset) in I, if $X \subset t$
- An association rule is an implication of the form:

  $X \rightarrow Y$, where $X, Y \subset I$, *and* $X \cap Y = \varnothing$

  - E.g., given a transaction {Milk, Bread, Cereal}, an association rule may be Milk, Bread → Cereal

    X ***Y***

- An itemset is a set of items
  - E.g., X = {Milk, Bread, Cereal} is an itemset
- A *k*-itemset is an itemset with *k* items.
  - E.g., {Milk, Bread, Cereal} is a 3-itemset

Rule Strength Measures

- The support count of X in T (denoted by $X.count$) is the number of transactions in T that contain X
  - E.g., in the transaction database T, {Bread}.count = 4, {Milk, Eggs}.count = 0

Transaction database *T*

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}

Rule Strength Measures

- The strength of a rule is measured by its support and confidence
- Assume the transaction database T has n transactions
- Support: The rule holds with support sup in T if sup% of transactions contain $X \cup Y$

$$support = \Pr(X \cup Y) = \frac{(X \cup Y).count}{n}$$

  - Support determines how frequent the rule is applicable in T
- Confidence: The rule holds in T with confidence conf if conf% of transactions that contain X also contain Y.

$$confidence = \Pr(Y \mid X) = \frac{(X \cup Y).count}{X.count}$$

  - Confidence thus determines the predictability of the rule

Goal and Key Features

- Goal: Find all rules that satisfy the user-specified minimum support (minsup) and minimum confidence (minconf)
- Key Features
  - Completeness: find all rules.
  - No target item(s) on the right-hand-side
  - Mining with data on hard disk (not in memory)

Association Rules Mining Algorithms

- In general, association rule mining can be viewed as a two-step process:
  1. Find all frequent itemsets: a frequent itemset is an itemset that has transaction support above minsup
  2. Generate strong association rules from the frequent itemsets: a confident association rule is a rule with confidence above minconf

An Example

- The transaction database T has seven transactions. Let minsup = 30% and minconf = 80%.
- {Chicken, Clothes, Milk} is a frequent 3-itemset

TID	Transaction
t1	Beef, Chicken, Milk
t2	Beef, Cheese
t3	Cheese, Boots
t4	Beef, Chicken, Cheese
t5	Beef, Chicken, Clothes, Cheese, Milk
t6	Chicken, Clothes, Milk
t7	Chicken, Milk, Clothes

- Chicken, Clothes $\rightarrow$ Milk [sup = 3/7, conf = 3/3] is valid
  - sup > 30% and conf > 80%
  - As well as Clothes, Milk $\rightarrow$ Chicken, Clothes $\rightarrow$ Milk, Chicken

An Example

- The transaction database T has seven transactions. Let minsup = 30% and minconf = 80%.
- {Chicken, Clothes, Milk} is a frequent 3-itemset

TID	Transaction
t1	Beef, Chicken, Milk
t2	Beef, Cheese
t3	Cheese, Boots
t4	Beef, Chicken, Cheese
t5	Beef, Chicken, Clothes, Cheese, Milk
t6	Chicken, Clothes, Milk
t7	Chicken, Milk, Clothes

- Chicken $\rightarrow$ Clothes, Milk [sup = 3/7, conf = 3/5] is invalid
  - sup > 30% but conf < 80%
  - As well as Chicken, Milk $\rightarrow$ Clothes, Milk $\rightarrow$ Chicken, Clothes

Association Rules Mining Algorithms

- There are a large number of them!!
  - Apriori, FP-Growth, ECLAT,…
- They use different strategies and data structures
- Their resulting sets of rules are all the same
  - Given a transaction data set T, and a minimum support and a minimum confident, the set of association rules existing in T is uniquely determined
- Any algorithm should find the same set of rules although their computational efficiencies and memory requirements may be different

Outline

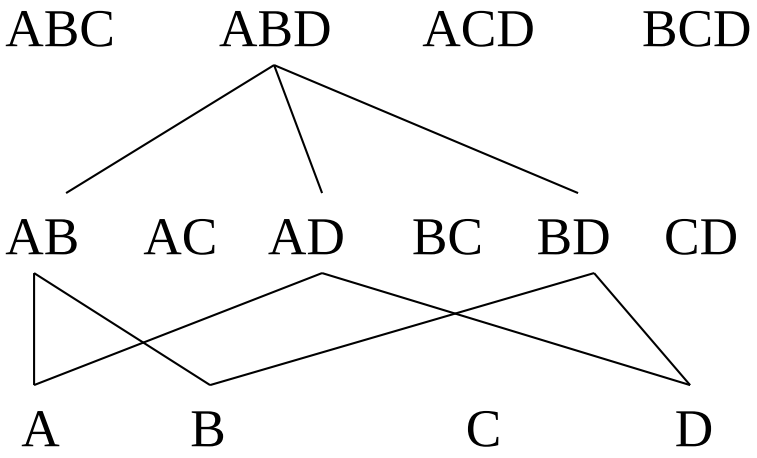
- What is Association Rule Mining?
- Basic Concetps of Association Rules
- **The Apriori Algorithm**
- The FP-Growth Algorithm
- Which Patterns Are Interesting?
- Maximal Frequent Patterns and Closed Frequent Patterns
- Summary

The Apriori Algorithm

- Proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules
  - Agrawal, R. and R. Srikant. Fast algorithms for mining association rules. In Proceedings of International Conference on Very Large Data Bases (VLDB-1994), 1994
- The best known algorithm
- Designed to operate on transaction databases
- It uses prior knowledge of frequent itemset properties

Step 1: Mining all frequent itemsets

- A frequent itemset is an itemset whose support is ≥ minsup
- The apriori property (downward closure property): any subsets of a frequent itemset are also frequent itemsets
  - If an itemset has at least one infrequent subset, can this itemset be frequent?



Illustrating Apriori Principle

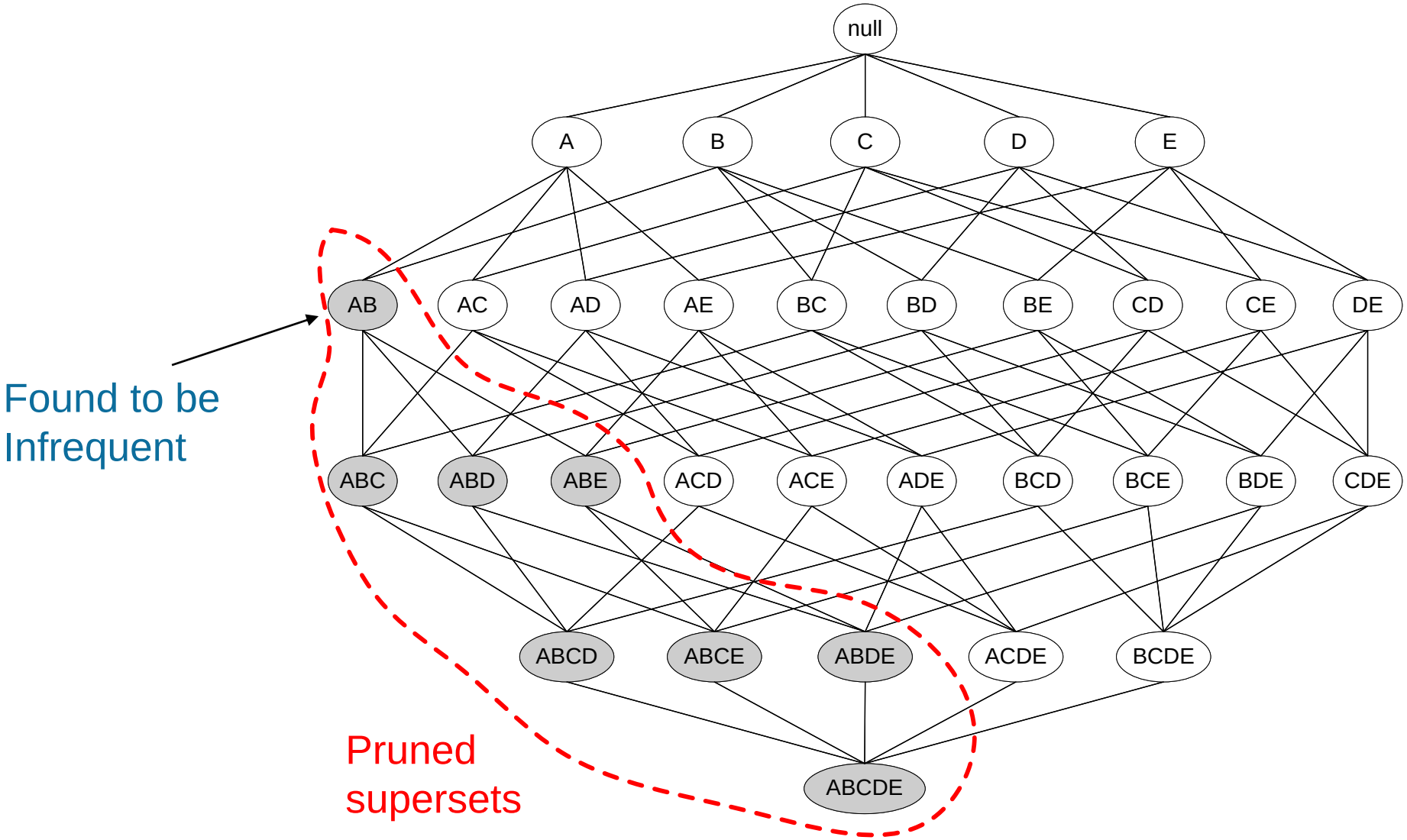
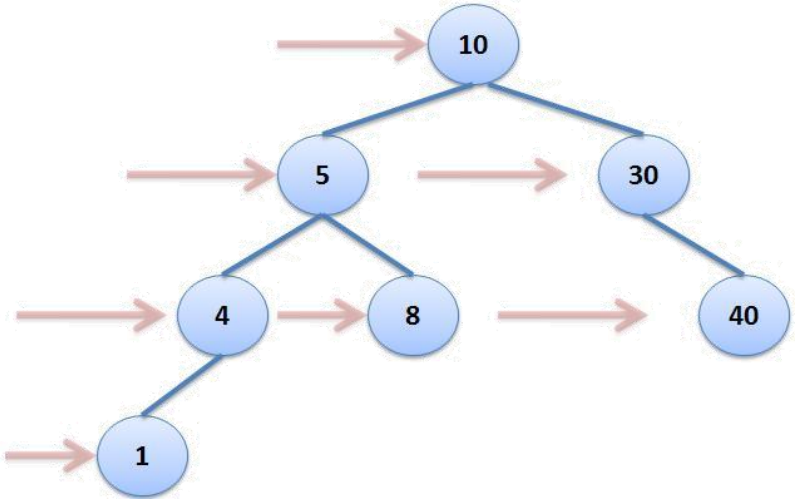


Figure from the book Introduction to Data Mining, Tan, Steinbach, Kumar

Step 1: Mining all frequent itemsets

- Assume that the items in I are sorted in lexicographic order (a total order)
- Let $\{w[1], w[2], \ldots, w[k]\}$ be a k-itemset w consisting of items $w[1]$, $w[2]$, …, $w[k]$, where $w[1] < w[2] < \cdots < w[k]$
- The Apriori is based on level-wise search, making multiple passes over the data.



Apriori algorithm for Step 1

- Input: Transaction database T
- Output: The set F of all frequent itemsets (line 13)

Algorithm Apriori(T)

```
1    C_1 ← init-pass(T);                          // the first pass over T
2    F_1 ← {f | f ∈ C_1, f.count/n ≥ minsup};     // n is the no. of transactions in T
3    for (k = 2; F_{k-1} ≠ ∅; k++) do             // subsequent passes over T
4       C_k ← candidate-gen(F_{k-1});
5       for each transaction t ∈ T do             // scan the data once
6           for each candidate c ∈ C_k do
7               if c is contained in t then
8                   c.count++;
9           endfor
10      endfor
11      F_k ← {c ∈ C_k | c.count/n ≥ minsup}
12   endfor
13   return F ← ⋃_k F_k;
```

Step 1: Pseudo Code

Algorithm Apriori(T)

1 $C_1 \leftarrow$ init-pass(T); // the first pass over T
2 $F_1 \leftarrow \{f \mid f \in C_1, f.\text{count}/n \geq minsup\}$; // n is the no. of transactions in T
3 **for** ($k = 2$; F_k
4 $C_k \leftarrow$ candi
5 **for** each tra
6 **for** eacl
7 **if** c is contained in t **then**
8 $c.count$++;
9 **endfor**
10 **endfor**
11 $F_k \leftarrow \{c \in C_k \mid c.count/n \geq minsup\}$
12 **endfor**
13 **return** $F \leftarrow \bigcup_k F_k$;

The first pass

- Count the supports of individual items (line 1)
- Determine the set of frequent 1-itemsets F_1 (line 2)

Step 1: An Example

Transaction database T

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}

Minsup = 40%

1st scan

C_1

Item	Sup
Bread	4
Cheese	3
Eggs	1
Juice	4
Milk	3
Yogurt	1

F_1

Item	Sup
Bread	4
Cheese	3
Juice	4
Milk	3

Step 1: Pseudo Code (cont.)

```
Algorithm Apriori(T)
1    C1 ← init-pass(T);                          // the first pass over T
2    F1 ← {f | f ∈ C1, f.count/n ≥ minsup};      // n is the no. of transactions in T
3    for (k = 2; Fk−1 ≠ ∅; k++) do               // subsequent passes over T
4        Ck ← candidate-gen(Fk−1);
5        for each transaction t ∈ T do           // scan the data once
6            for each candidate c ∈ Ck do
7                if c is contained in t then
8                    c.count++;
9            endfor
10       endfor
```

In each subsequent pass k ($k > 2$)

1. Generate candidate itemsets C_k from the frequent itemsets F_{k-1} using the function *candidate-gen* (line 4)
2. Scan the database again to count the actual support of each candidate itemset c in C_k (line 5-10)

Step 1: An Example (cont.)

Transaction database T

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}

Minsup = 40%

1st scan

C_1

Item	Sup
Bread	4
Cheese	3
Eggs	1
Juice	4
Milk	3
Yogurt	1

F_1

Item	Sup
Bread	4
Cheese	3
Juice	4
Milk	3

C_2

Item
Bread, Cheese
Bread, Juice
Bread, Milk
Cheese, Juice
Cheese, Milk
Juice, Milk

2nd scan

C_2

Item	Sup
Bread, Cheese	2
Bread, Juice	3
Bread, Milk	2
Cheese, Juice	3
Cheese, Milk	1
Juice, Milk	2

Step 1: Pseudo Code (cont.)

Algorithm Apriori(T)

```
1    C_1 ← init-pass(T);                          // the first pass over T
2    F_1 ← {f | f ∈ C_1, f.count/n ≥ minsup};     // n is the no. of transactions in T
3    for (k = 2; F_{k-1} ≠ ∅; k++) do             // subsequent passes over T
4       C_k ← candidate-gen(F_{k-1});
5       for each transaction t ∈ T do             // scan the data once
6          for each candidate c ∈ C_k do
7             if c is contained in t then
8                c.count++;
9          endfor
10      endfor
11      F_k ← {c ∈ C_k | c.count/n ≥ minsup}
```

In each subsequent pass k ($k > 2$)

3. Build the frequent itemset F_k by determines which of the candidate itemsets c are actually frequent (line 11)

Step 1: An Example (cont.)

Transaction database T

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}

Minsup = 40%

1st scan

C_1

Item	Sup
Bread	4
Cheese	3
Eggs	1
Juice	4
Milk	3
Yogurt	1

F_1

Item	Sup
Bread	4
Cheese	3
Juice	4
Milk	3

C_2

Item
Bread, Cheese
Bread, Juice
Bread, Milk
Cheese, Juice
Cheese, Milk
Juice, Milk

2nd scan

C_2

Item	Sup
Bread, Cheese	2
Bread, Juice	3
Bread, Milk	2
Cheese, Juice	3
Cheese, Milk	1
Juice, Milk	2

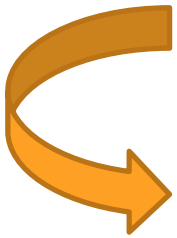
F_2

Item	Sup
Bread, Cheese	2
Bread, Juice	3
Bread, Milk	2
Cheese, Juice	3
Juice, Milk	2

Step 1: An Example (cont.)

F_2

Item	Sup
Bread, Cheese	2
Bread, Juice	3
Bread, Milk	2
Cheese, Juice	3
Juice, Milk	2



C_3

Item
Bread, Cheese, Juice
Bread, Juice, Milk

3rd scan

C_3

Item	Sup
Bread, Cheese, Juice	2
Bread, Juice, Milk	1

F_3

Item	Sup
Bread, Cheese, Juice	2

$$F = F_1 \cup F_2 \cup F_3$$

Candidate-gen Function

- The candidate generation function consists of two steps: the join step and the pruning step

Function candidate-gen(F_{k-1})

```
1   C_k ← ∅;                                  // initialize the set of candidates
2   forall f_1, f_2 ∈ F_{k-1}                 // find all pairs of frequent itemsets
3     with f_1 = {i_1, ..., i_{k-2}, i_{k-1}} // that differ only in the last item
4     and f_2 = {i_1, ..., i_{k-2}, i'_{k-1}}
5     and i_{k-1} < i'_{k-1} do               // according to the lexicographic order
6       c ← {i_1, ..., i_{k-1}, i'_{k-1}};    // join the two itemsets f_1 and f_2
7       C_k ← C_k ∪ {c};                      // add the new itemset c to the candidates
8       for each (k-1)-subset s of c do
9         if (s ∉ F_{k-1}) then
10          delete c from C_k;                // delete c from the candidates
11      endfor
12  endfor
13  return C_k;                               // return the generated candidates
```

Join step

Pruning step

Candidate-gen Function

- Join step (lines 2–6): Two frequent (k-1) itemsets are joined to produce a possible candidate c (line 6)
  - The two frequent itemsets f_1 and f_2 have exactly the same items except the last one (lines 3–5)
  - c is added to the set of candidates C_k (line 7)
- Pruning step (lines 8–11): A candidate c from the join step will be deleted from C_k if any of its $k-1$ subsets (there are k of them) is not in F_{k-1}
  - Explain why?

Candidate-gen Function: Example

- Let the set of frequent itemsets at level 3 be

$$F_3 = \{\{1,2,3\},\{1,2,4\},\{1,3,4\},\{1,3,5\},\{2,3,4\}\}$$

- The join step (which generates candidates for level 4) will produce two candidate itemsets

$$C_4 = \{\{1,2,3,4\},\{1,3,4,5\}\}$$

  - $\{1,2,3,4\}$ is generated by joining $\{1,2,3\}$ and $\{1,2,4\}$
  - $\{1,3,4,5\}$ is generated by joining $\{1,3,4\}$ and $\{1,3,5\}$

- After the pruning step: $C_4 = \{\{1,2,3,4\}\}$
  - $\{1,4,5\}$ is not in F_3 and thus $\{1,3,4,5\}$ cannot be frequent

Exercise 1

- A database has seven transactions. Let minsup = 30%. Find all frequent itemsets using Apriori algorithm.

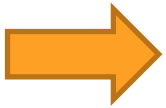
TID	Transaction
t1	Beef, Chicken, Milk
t2	Beef, Cheese
t3	Cheese, Boots
t4	Beef, Chicken, Cheese
t5	Beef, Chicken, Clothes, Cheese, Milk
t6	Chicken, Clothes, Milk
t7	Chicken, Milk, Clothes

Algorithm Efficiency

- Apriori is theoretically an exponential algorithm
  - Let the number of items in I be m. The space of all itemsets is $O(2^m)$ because each item may or may not be in an itemset
- The sparseness of the data and the high minimum support value make the mining possible and efficient
  - Sparseness: the store sells a lot of items, but each shopper only purchases a few of them

Transaction data

TID	Transaction
10	{Juice}
20	{Milk, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese}
50	{Cheese}



Binaries data

B	C	E	J	M	Y
0	0	0	1	0	0
0	0	0	0	1	1
1	0	0	1	1	0
1	1	1	0	0	0
0	1	0	0	0	0

Algorithm Efficiency

- The algorithm can scale up to large data sets as it does not load the entire data into the memory
  - It only scans the data K times, where K is the size of the largest itemset. In practice, K is often small (e.g., < 10)
- The algorithm is based on level-wise search. It has the flexibility to stop at any level
  - Long frequent itemsets or long association rules may not be meaningful in some applications
- Given a transaction database T and a minsup, any algorithm should find the same set of frequent itemsets
  - This property does not hold for many other data mining tasks, e.g., classification or clustering

Algorithm Efficiency

- The interestingness problem: a huge number of itemsets (and rules), tens of thousands, or more, are produced
  - It is hard for the user to analyze them to find those useful ones

Exercise 2

- A database has four transactions. Let minsup = 50%. Find all frequent itemsets using Apriori algorithm.

TID	Transaction
t1	Crab, Milk, Cheese, Bread
t2	Cheese, Milk, Apple, Pie, Bread
t3	Apple, Crab, Pie, Bread
t4	Bread, Milk, Cheese

Step 2: Generating association rules

- Frequent itemsets ≠ association rules
- One more step is needed to generate association rules
- For each frequent itemset X,

  For each proper nonempty subset B of X

  - Let $A = X - B$
  - $A \rightarrow B$ is an association rule if

    $$confidence\,(A \rightarrow B) \geq minconf$$

- $support(A \rightarrow B) = support(A \cup B) = support(X)$
- $confidence(A \rightarrow B) = \frac{support(A \cup B)}{support(A)}$

Step 2: An Example

- Suppose $\{2, 3, 4\}$ is frequent, with sup = 50%
- Proper nonempty subsets: $\{2,3\}$, $\{2,4\}$, $\{3,4\}$, $\{2\}$, $\{3\}$, $\{4\}$, with sup = 50%, 50%, 75%, 75%, 75%, 75% respectively
- These generate these (candidate) association rules:
  - 2,3 $\rightarrow$ 4, confidence=100%
  - 2,4 $\rightarrow$ 3, confidence=100%
  - 3,4 $\rightarrow$ 2, confidence=67%
  - 2 $\rightarrow$ 3,4, confidence=67%
  - 3 $\rightarrow$ 2,4, confidence=67%
  - 4 $\rightarrow$ 2,3, confidence=67%
  - All rules have support = 50%

Step 2: Generating association rules

- Such exhaustive rule generation strategy is inefficient
- Alternative strategy: If $X - B \rightarrow B$ is valid then all rules $X - B_{sub} \rightarrow B_{sub}$ must be valid
  - B_{sub} is a non-empty subset of B
- For example: Given a frequent itemset $\{A, B, C, D\}$. If $(A, B \rightarrow C, D)$ is valid then $(A, B, C \rightarrow D)$ and $(A, B, D \rightarrow C)$ are also valid

Rule Generation for Apriori Algorithm

Lattice of rules

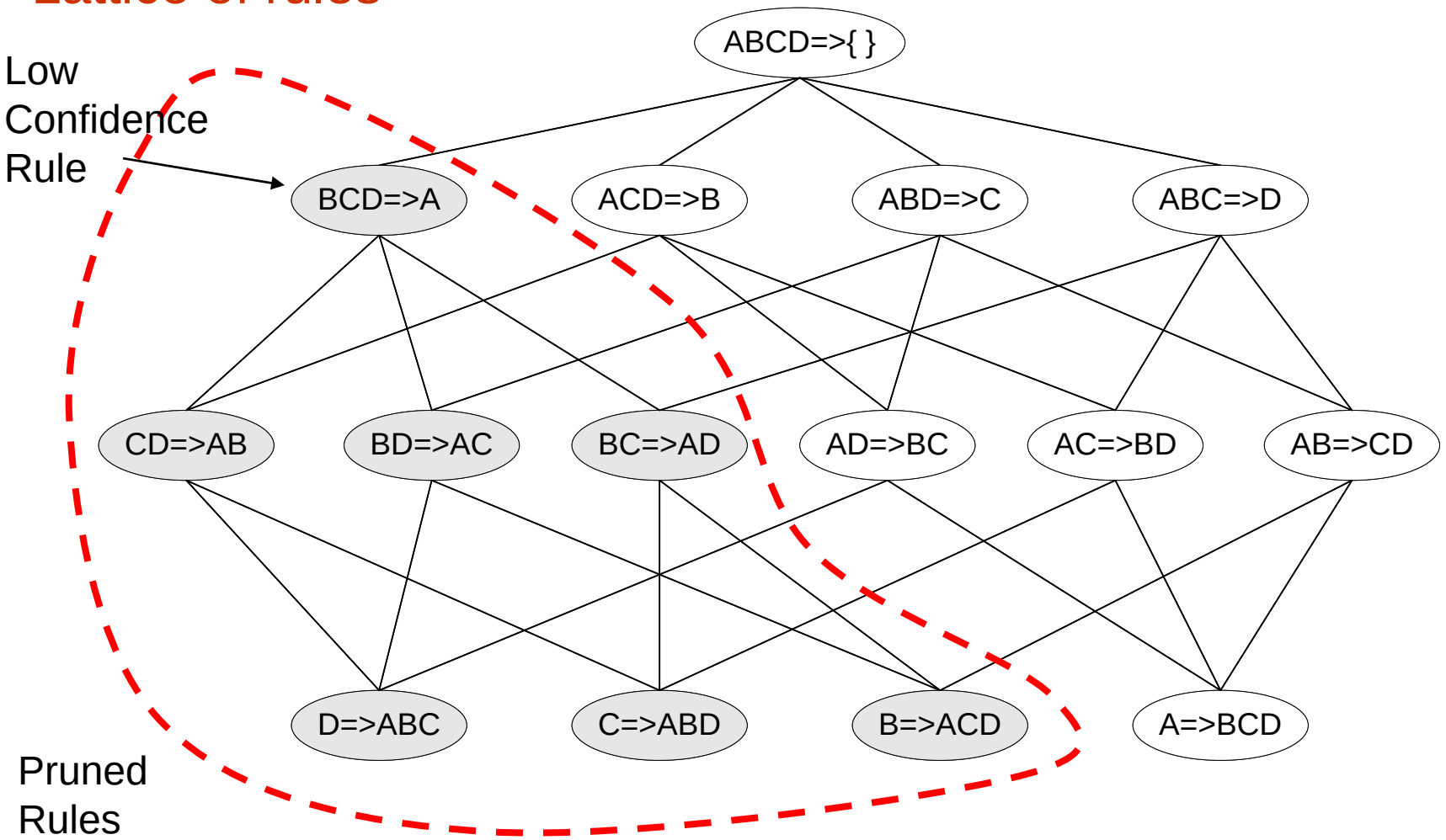


Figure from the book Introduction to Data Mining, Tan, Steinbach, Kumar

Step 2: Generating association rules

- To recap, in order to obtain $A \rightarrow B$, we need to have support ($A \rightarrow B$) and support (A)
- All the required information for confidence computation has already been recorded in itemset generation. No need to see the data T any more
- This step is not as time-consuming as frequent itemsets generation

Step 2: Algorithm for Generating Rules

Algorithm genRules(F) // F is the set of all frequent itemsets
1 **for** each frequent k-itemset f_k in F, $k \geq 2$ **do**
2 output every 1-item consequent rule of f_k with confidence $\geq$ *minconf* and
support $\leftarrow f_k.count / n$ // n is the total number of transactions in T
3 $H_1 \leftarrow$ {consequents of all 1-item consequent rules derived from f_k above};
4 ap-genRules(f_k, H_1);
5 **endfor**

Procedure ap-genRules(f_k, H_m) // H_m is the set of m-item consequents
1 **if** ($k > m + 1$) AND ($H_m \neq \emptyset$) **then**
2 $H_{m+1} \leftarrow$ candidate-gen(H_m);
3 **for** each h_{m+1} in H_{m+1} **do**
4 $conf \leftarrow f_k.count / (f_k - h_{m+1}).count$;
5 **if** ($conf \geq minconf$) **then**
6 output the rule $(f_k - h_{m+1}) \rightarrow h_{m+1}$ with confidence = *conf* and
support = $f_k.count / n$; // n is the total number of transactions in T
7 **else**
8 delete h_{m+1} from H_{m+1};
9 **endfor**
10 ap-genRules(f_k, H_{m+1});
11 **endif**

Step 2: An Example

Transaction database *T*

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}

Minsup = 40%
Minconf = 60%

- F_1 = {{Bread}:4, {Cheese}:3, {Juice}:4, {Milk}:3}
- F_2 = {{Bread, Cheese}:2, {Bread, Juice}:3, {Bread, Milk}:2, {Cheese, Juice}:3, {Juice, Milk}:2}
- F_3 = {{Bread, Cheese, Juice}:2}
- Let's generate rules for F_3
- While rules for F_2 can be generated in a similar way

Step 2: Algorithm for Generating Rules

Algorithm genRules(F) // F is the set of all frequent itemsets

1 **for** each frequent k-itemset f_k in F, $k \geq 2$ **do**

2 output every 1-item consequent rule of f_k with confidence $\geq$ *minconf* and support $\leftarrow f_k.count / n$ // n is the total number of transactions in T

First generate all rules with one item in the consequent and output those whose conf $\geq$ minconf

1. Bread, Cheese $\rightarrow$ Juice [sup = 2/5, conf = 1] ✓
2. Bread, Juice $\rightarrow$ Cheese [sup = 2/5, conf = 2/3] ✓
3. Cheese, Juice $\rightarrow$ Bread [sup = 2/5, conf = 2/3] ✓

Step 2: Algorithm for Generating Rules

Algorithm genRules(F) // F is the set of all frequent itemsets

1 **for** each frequent k-itemset f_k in F, $k \geq 2$ **do**

2 output every 1-item consequent rule of f_k with confidence $\geq$ *minconf* and support $\leftarrow f_k.count / n$ // n is the total number of transactions in T

3 $H_1 \leftarrow$ {consequents of all 1-item consequent rules derived from f_k above};

Add to H_1 the consequents of all **valid** 1-item consequent association rules

1. Bread, Cheese $\rightarrow$ Juice [sup = 2/5, conf = 1] ✓
2. Bread, Juice $\rightarrow$ Cheese [sup = 2/5, conf = 2/3] ✓
3. Cheese, Juice $\rightarrow$ Bread [sup = 2/5, conf = 2/3] ✓

H_1 = {{Bread}, {Cheese}, {Juice}}

Step 2: Algorithm for Generating Rules

Procedure ap-genRules(f_k, H_m) // H_m is the set of m-item consequents

1 **if** ($k > m + 1$) AND ($H_m \neq \emptyset$) **then**

2 $H_{m+1} \leftarrow$ candidate-gen(H_m);

Generate H_{m+1} from H_m using candidate-gen function

4 $conf \leftarrow f_k.count / (f_k - h_{m+1}).count$;

5 **if** ($conf \geq minconf$) **then**

6 output the rule $(f_k - h_{m+1}) \rightarrow h_{m+1}$ with confidence = $conf$ and support = $f_k.count / n$; // n is the total number of transactions in T

7 **else**

8 delete h_{m+1} from H_{m+1};

9 **endfor**

10 ap-genRules(f_k, H_{m+1});

11 **endif**

H_1 = {{Bread}, {Cheese}, {Juice}}

$\rightarrow H_2$ = {{Bread, Cheese}, {Bread, Juice}, {Cheese, Juice}}

Step 2: Algorithm for Generating Rules

Generate a candidate rule having its consequent $h_{m+1} \in H_{m+1}$

- If $conf(f_k - h_{m+1} \rightarrow h_{m+1}) \geq minconf$ then output the rule
- Otherwise, delete h_{m+1} from H_{m+1}

```
4       conf ← f_k.count / (f_k − h_{m+1}).count;
5       if (conf ≥ minconf) then
6           output the rule (f_k − h_{m+1}) → h_{m+1} with confidence = conf and
            support = f_k.count / n;     // n is the total number of transactions in T
7       else
8         delete h_{m+1} from H_{m+1};
```

4. Juice → Bread, Cheese [sup = 2/5, conf = 2/4] ✘
   Cheese → Bread, Juice [sup = 2/5, conf = 2/3] ✓
5. Bread → Cheese, Juice [sup = 2/5, conf = 2/4] ✘

H_2 = {{~~Bread, Cheese~~}, {Bread, Juice}, {~~Cheese, Juice~~}}

Step 2: Algorithm for Generating Rules

Procedure ap-genRules(f_k, H_m) // H_m is the set of m-item consequents

1 **if** ($k > m + 1$) AND ($H_m \neq \emptyset$) **then**

2 $H_{m+1} \leftarrow$ candidate-gen(H_m);

3 **for** each h_{m+1} in H_{m+1} **do**

4 $conf \leftarrow f_k.count / (f_k - h_{m+1}).count$;

5 **if** ($conf \geq minconf$) **then**

6 output the rule $(f_k - h_{m+1}) \rightarrow h_{m+1}$ with confidence = $conf$ and support = $f_k.count / n$; // n is the total number of transactions in T

7 **else**

8 delete h_{m+1} from H_{m+1};

9 **endfor**

10 ap-genRules(f_k, H_{m+1});

Recursively call the function with H_{m+1}

Exercise 1 (cont.)

- A database has seven transactions. Let minsup = 30% and minconf = 80%. Find all association rules from the set of frequent itemsets found in the first step of Apriori.

TID	Transaction
t1	Beef, Chicken, Milk
t2	Beef, Cheese
t3	Cheese, Boots
t4	Beef, Chicken, Cheese
t5	Beef, Chicken, Clothes, Cheese, Milk
t6	Chicken, Clothes, Milk
t7	Chicken, Milk, Clothes

Exercise 1 (cont.) - Solution

- ~~Beef → Cheese [sup = 3/7, conf = 3/4]~~
- ~~Cheese → Beef [sup = 3/7, conf = 3/4]~~
- ~~Beef → Chicken [sup = 3/7, conf = 3/4]~~
- ~~Chicken → Beef [sup = 3/7, conf = 3/5]~~
- ~~Chicken → Clothes [sup = 3/7, conf = 3/5]~~

1. Clothes → Chicken [sup = 3/7, conf = 3/3]
2. Chicken → Milk [sup = 4/7, conf = 4/5]
3. Milk → Chicken [sup = 4/7, conf = 4/4]
4. Clothes → Milk [sup = 3/7, conf = 3/3]

- ~~Milk → Clothes [sup = 3/7, conf = 3/4]~~

5. Chicken, Clothes → Milk [sup = 3/7, conf = 3/3]

- ~~Chicken, Milk → Clothes [sup = 3/7, conf = 3/4]~~

6. Clothes, Milk → Chicken [sup = 3/7, conf = 3/3]
7. Clothes → Milk, Chicken [sup = 3/7, conf = 3/3]

Exercise 2 (cont.)

- A database has four transactions. Let minsup = 50% and minconf = 80%. Find all association rules from the set of frequent itemsets found in the first step of Apriori.

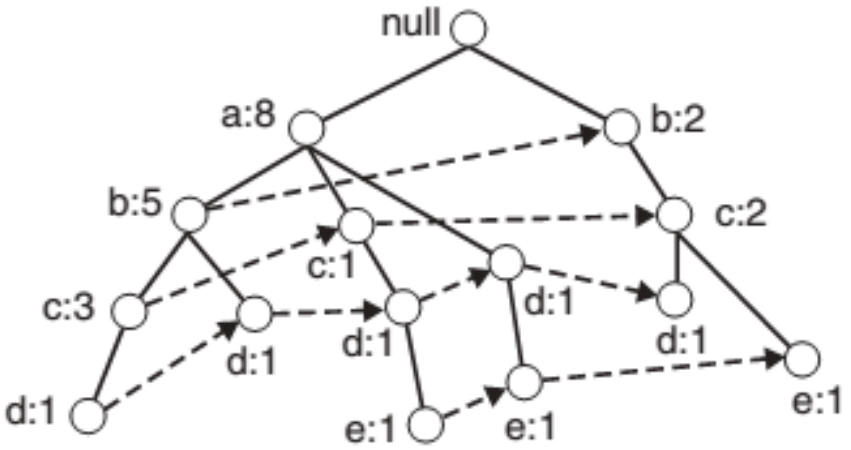
TID	Transaction
t1	Crab, Milk, Cheese, Bread
t2	Cheese, Milk, Apple, Pie, Bread
t3	Apple, Crab, Pie, Bread
t4	Bread, Milk, Cheese

Outline

- What is Association Rule Mining?
- Basic Concetps of Association Rules
- The Apriori Algorithm
- **The FP-Growth Algorithm**
- Which Patterns Are Interesting?
- Maximal Frequent Patterns and Closed Frequent Patterns
- Summary

The FP-Growth Algorithm

- Encodes the data set using a compact data structure called an FP-tree and extracts frequent itemsets directly from this structure
- Idea: Recursively grow frequent patterns by pattern and database partition
- The FP-Growth Approach
  - Depth-first search (Apriori: Breadth-first search)
  - Avoid explicit candidate gen



The FP-Growth Algorithm

FP-tree construction:

- Scan DB once, find frequent 1-itemset (single item pattern)
- Sort frequent items in frequency descending order, f-list
- Scan DB again, construct FP-tree

FP-Growth approach:

- For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
- Repeat the process on newly created conditional FP-trees
- Until the resulting FP-tree is empty, or it contains only one path. A single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

Construct FP-tree from a Transaction Database

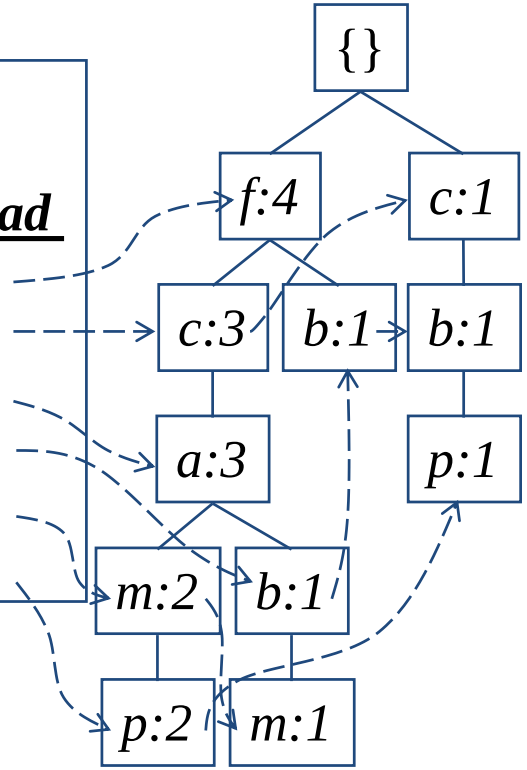
TID	Transaction	(ordered frequent items)
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

Minsup = 60%

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in **frequency descending order**, f-list
3. Scan DB again, construct FP-tree

Header Table

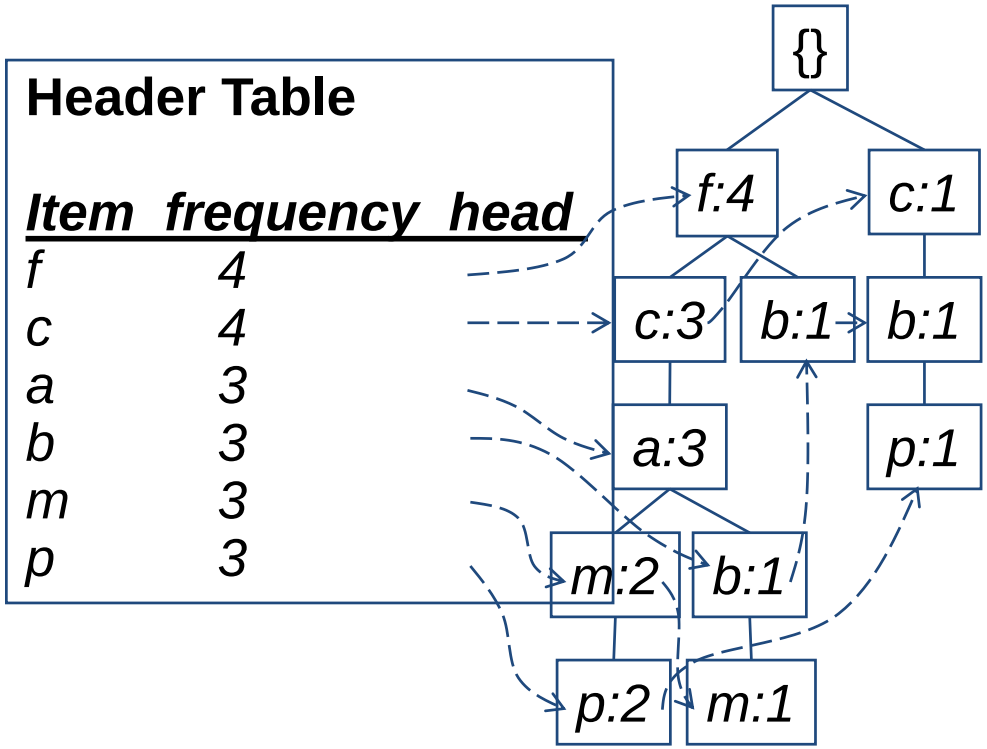
Item	*frequency*	*head*
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p	*3*	



F-list = f-c-a-b-m-p

Find Patterns Having P From P-conditional Database

- Starting at the frequent item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item p
- Accumulate all of transformed prefix paths of item p to form p's conditional pattern base

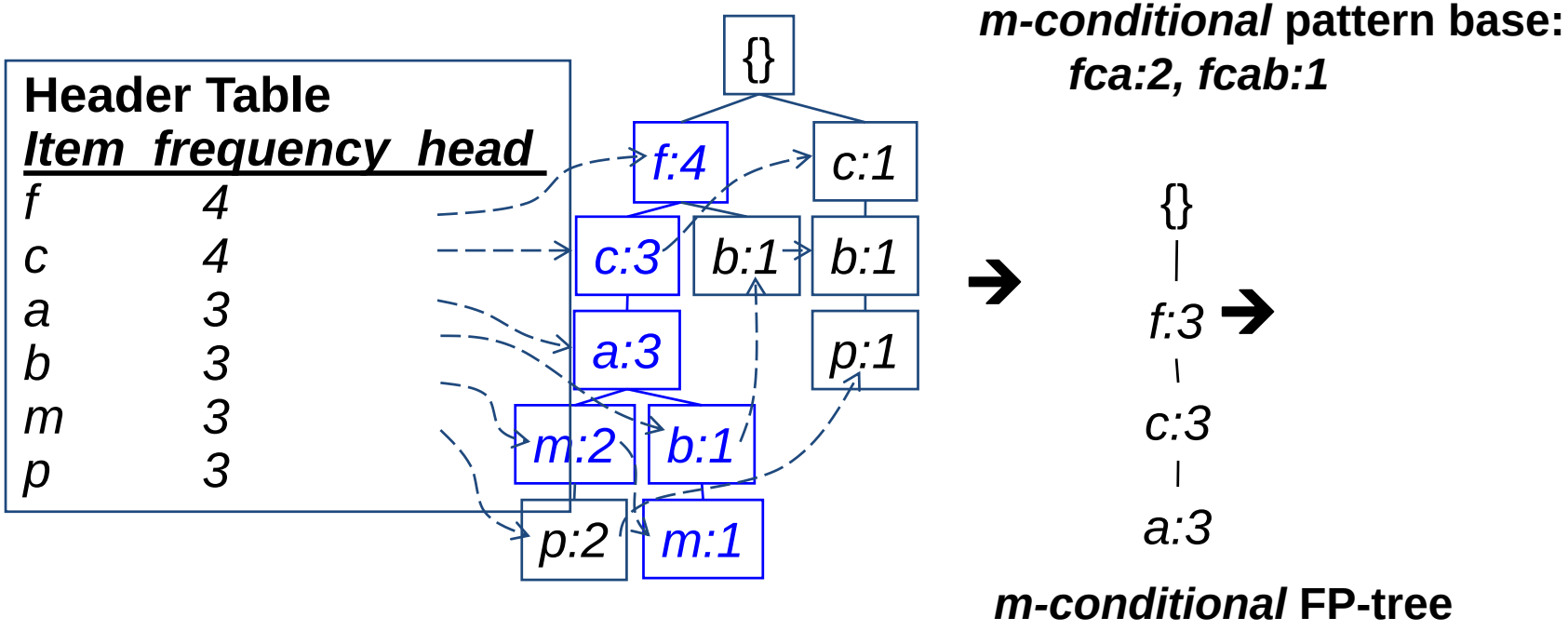


***Conditional* pattern bases**

item	***cond. pattern base***
f	
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

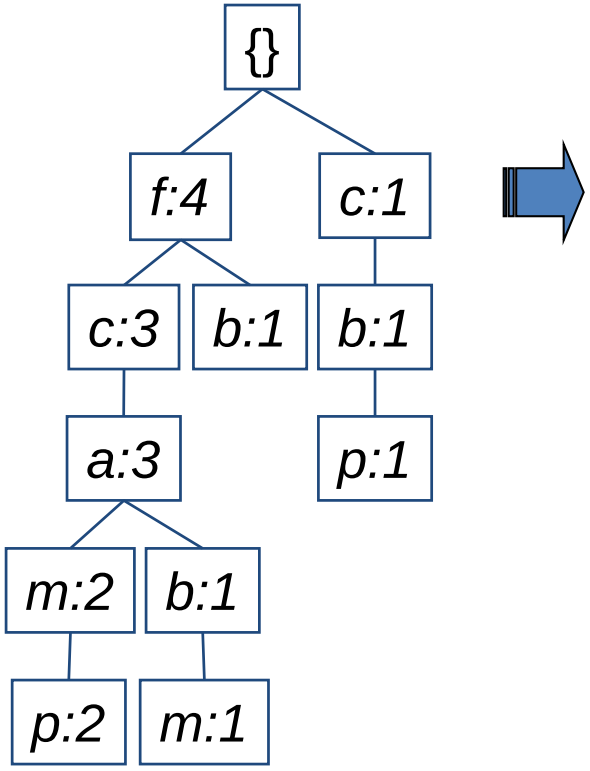
From Conditional Pattern-bases to Conditional FP-trees

- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the **frequent items** of the pattern base

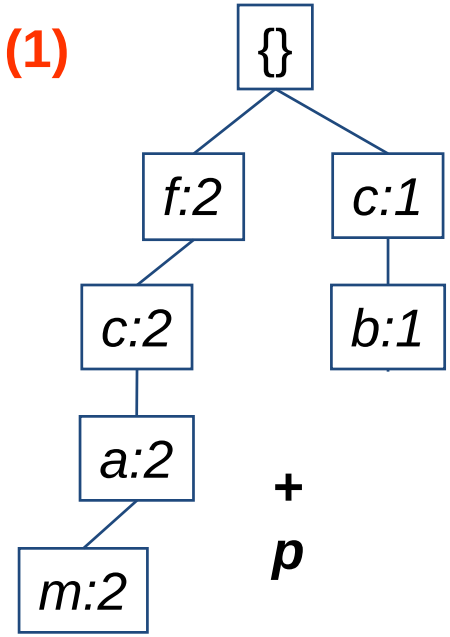


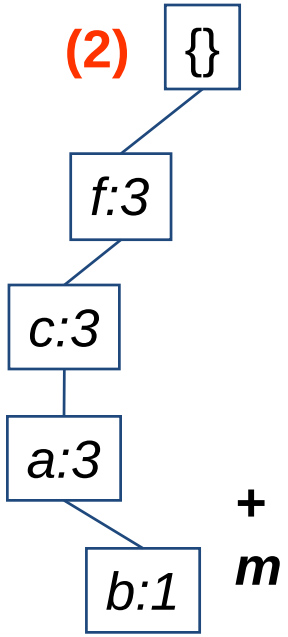
***Conditional* pattern bases**

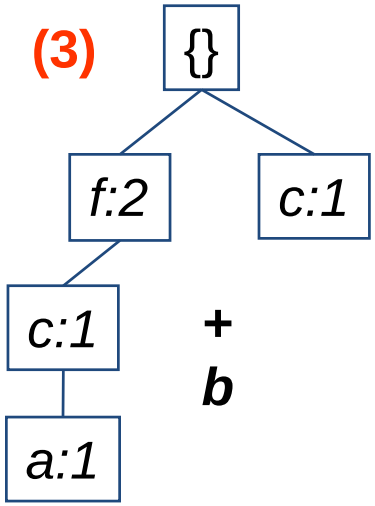
item	***cond. pattern base***
f	
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

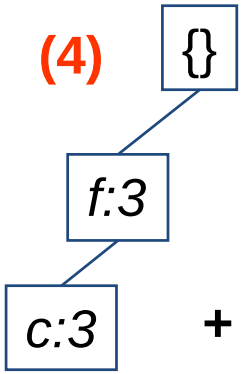


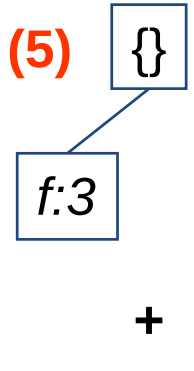
Subtrees





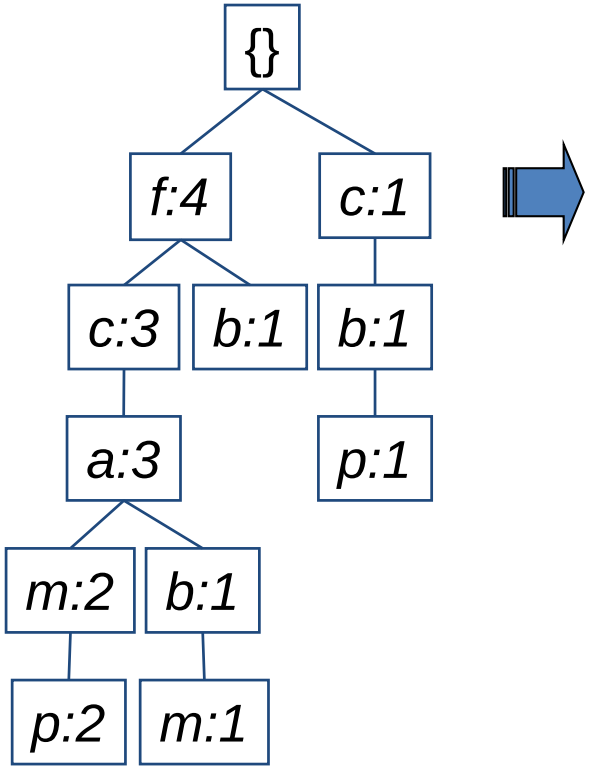






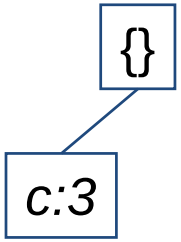
***Conditional* pattern bases**

item	***cond. pattern base***
f	
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***



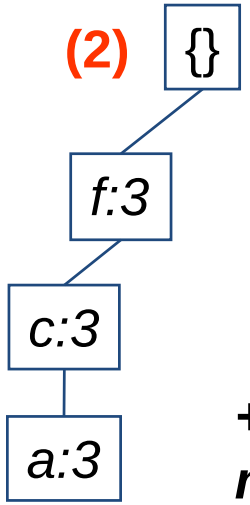
Conditional FP-trees

(1)



\+ ***p***

(2)

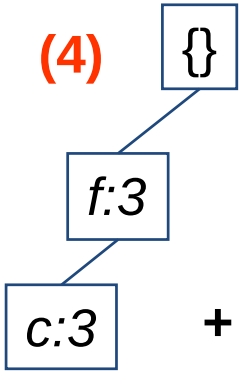


\+ ***m***

(3)

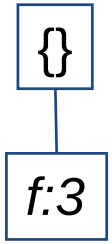
\+ ***b***

(4)



\+ ***a***

(5)



\+ ***c***

(6)

\+ ***f***

Mining Frequent Itemsets from Conditional FP-Trees

- If the conditional FP-tree T has one single prefix path P
  - Generate frequent itemsets that are subpaths of P

***m-conditional* FP-tree**

All frequent patterns relate to *m*

***m*,**

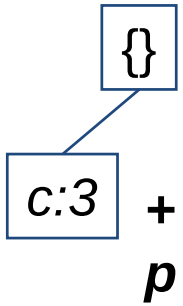
fm, cm, am,

fcm, fam, cam,

fcam

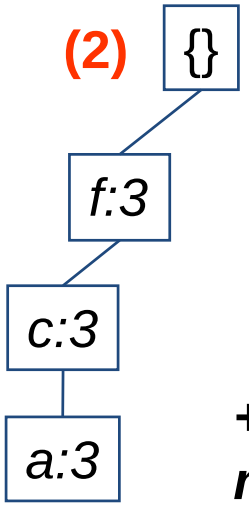
- If the conditional FP-tree has multiple prefix paths
  - Recursively apply the process (find the conditional pattern-bases and build conditional FP-trees) for T

(1)



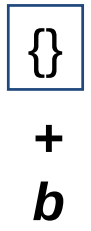
p:3
cp:3

(2)



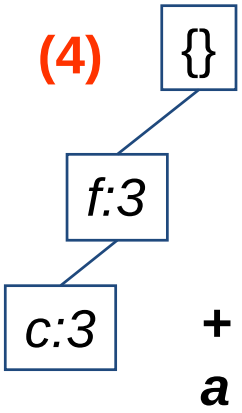
m:3
fm:3, cm:3, am:3
fcm:3, fam:3, cam:3
fcam:3

(3)



b:3

(4)



a:3
fa:3, ca:3
fca:3

(5)

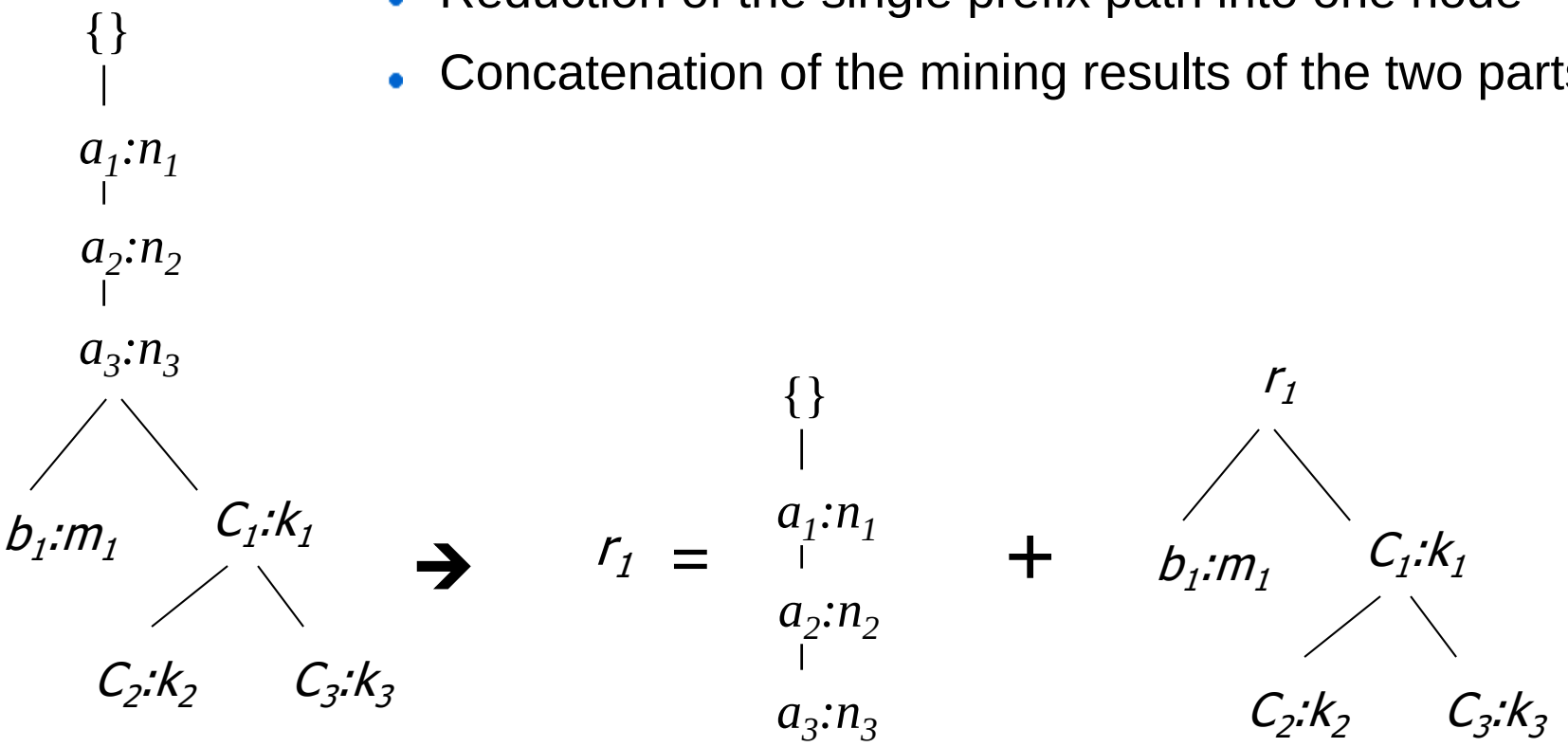
c:4
fc:3

(6)

f:4

A Special Case: Single Prefix Path in FP-tree

- Suppose a (conditional) FP-tree T has a shared single prefix-path P
- Mining can be decomposed into two parts
  - Reduction of the single prefix path into one node
  - Concatenation of the mining results of the two parts



Benefits of the FP-tree Structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the count field)

Outline

- What is Association Rule Mining?
- Basic Concetps of Association Rules
- The Apriori Algorithm
- The FP-Growth Algorithm
- **Which Patterns Are Interesting?**
- Maximal Frequent Patterns and Closed Frequent Patterns
- Summary

Pattern Evaluation

- Association rule algorithms tend to produce a large number of rules
  - Many of them are uninteresting or redundant
  - Redundant if A, B, C $\rightarrow$ D and A, B $\rightarrow$ D have same support and 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 Interestingness Measure

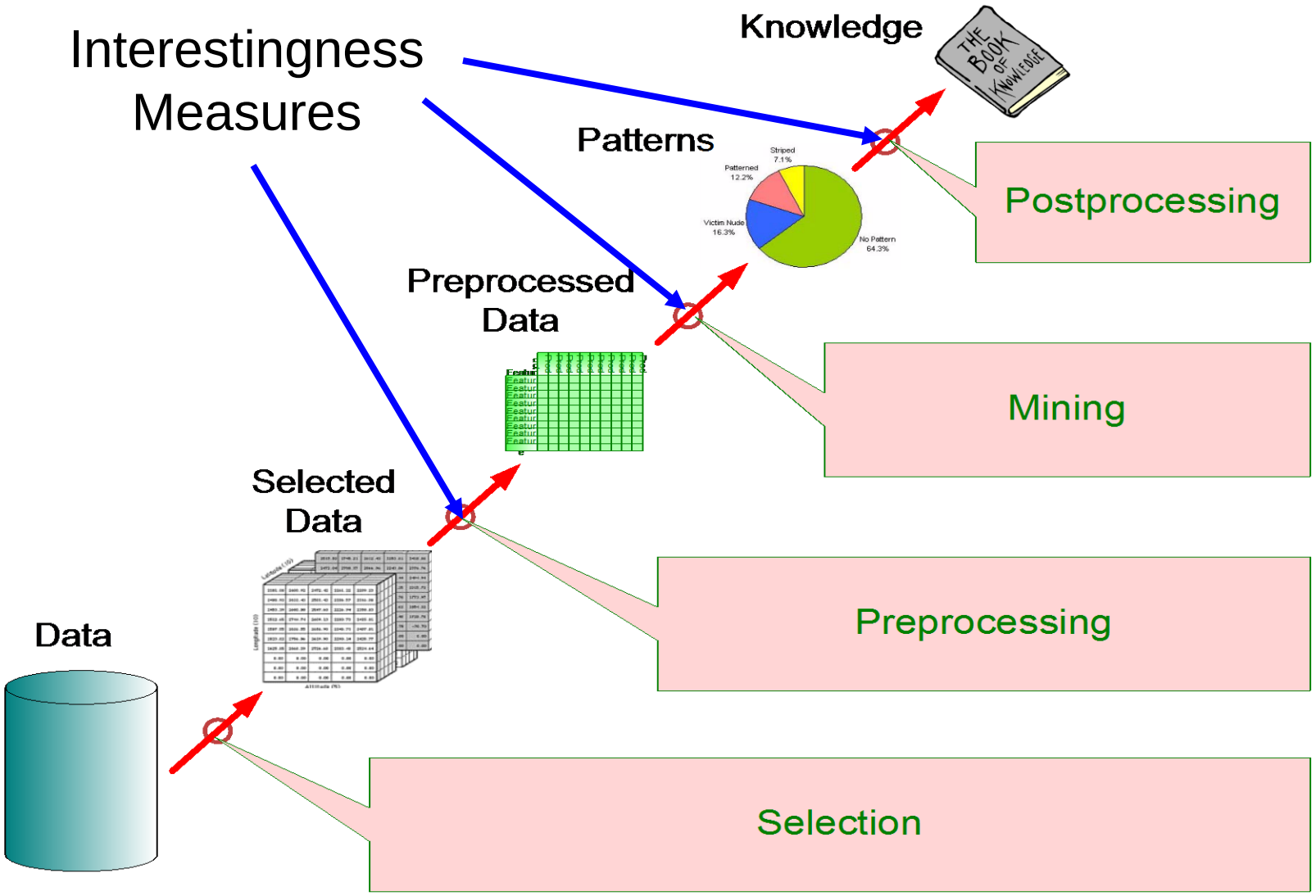


Figure from Lecture Notes for Chapter 6, Introduction to Data Mining, Tan, Steinbach, Kumar

Computing Interestingness Measure

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

Contingency table for $A \rightarrow B$

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

f_{11}: support of A and B
f_{10}: support of $\overline{A}$ and $\overline{B}$
f_{01}: support of $\overline{A}$ and B
f_{00}: support of A and $\overline{B}$

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 → Coffee

- Confidence = P(Coffee | Tea) = 0.75 but P(Coffee) = 0.9

  ⇒ Although confidence is high, rule is misleading

  ⇒ 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)

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

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

Statistical-based Measures

- Measures that take into account statistical dependence
- $Lift = \frac{P(B|A)}{P(B)} = \frac{sup(A \cup B)}{sup(A)sup(B)}$ (also called $Interest$)
- $Leverage = P(A,B) - P(A)P(B)$
  $= sup(A \cup B) - sup(A)sup(B)$
- $Conviction = \frac{P(A)P(\neg B)}{P(A \neg B)} = \frac{1 - sup(B)}{1 - conf(A \rightarrow B)}$

Statistical-based Measures: An Example

TID	Transaction
t1	Milk, Bread
t2	Butter
t3	Beer, Diapers
t4	Milk, Bread, Butter
t5	Bread

Association Rule:

Bread, Milk → Butter

- Sup(Bread, Butter, Milk) = 0.2, sup(Bread, Milk) = 0.4, sup(Butter) = 0.4
- Support = 0.2, confidence = 0.2 / 0.4 = 0.5
- Lift = 0.2 / (0.4 * 0.4) = 1.25
- Leverage = 0.2 – 0.4*0.4 = 0.04
- Conviction = (1 – 0.4) / (1 – 0.5) = 1.2

Drawback of Lift & Interest

- Contingency tables for the word pairs $\{p, q\}$ and $\{r, s\}$

	p	$\overline{p}$	
q	880	50	930
$\overline{q}$	50	20	70
	930	70	1000

	r	$\overline{r}$	
s	20	50	70
$\overline{s}$	50	880	930
	70	930	1000

- The lift factor for $\{p, q\}$ is 1.02 and for $\{r, s\}$ is 4.08.
- p and q appear together in 88% of the documents, their interest factor is close to 1 (independence)
- The lift for $\{p, q\}$ is higher than for $\{r, s\}$ even though r and s seldom appear together in the same document.
- Confidence is perhaps the better choice in this situation

There are lots of measures proposed in the literature

Some measures are good for certain applications, but not for others

What criteria should we use to determine whether a measure is good or bad?

Read Chapter 6, Introduction to Data Mining, Tan, Steinbach, Kumar

#	Measure	Formula
1	ϕ-coefficient	$\frac{P(A,B)-P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
2	Goodman-Kruskal's (λ)	$\frac{\sum_j \max_k P(A_j,B_k)+\sum_k \max_j P(A_j,B_k)-\max_j P(A_j)-\max_k P(B_k)}{2-\max_j P(A_j)-\max_k P(B_k)}$
3	Odds ratio (α)	$\frac{P(A,B)P(\overline{A},\overline{B})}{P(A,\overline{B})P(\overline{A},B)}$
4	Yule's Q	$\frac{P(A,B)P(\overline{AB})-P(A,\overline{B})P(\overline{A},B)}{P(A,B)P(\overline{AB})+P(A,\overline{B})P(\overline{A},B)} = \frac{\alpha-1}{\alpha+1}$
5	Yule's Y	$\frac{\sqrt{P(A,B)P(\overline{AB})}-\sqrt{P(A,\overline{B})P(\overline{A},B)}}{\sqrt{P(A,B)P(\overline{AB})}+\sqrt{P(A,\overline{B})P(\overline{A},B)}} = \frac{\sqrt{\alpha}-1}{\sqrt{\alpha}+1}$
6	Kappa (κ)	$\frac{P(A,B)+P(\overline{A},\overline{B})-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A)P(B)-P(\overline{A})P(\overline{B})}$
7	Mutual Information (M)	$\frac{\sum_i \sum_j P(A_i,B_j)\log \frac{P(A_i,B_j)}{P(A_i)P(B_j)}}{\min(-\sum_i P(A_i)\log P(A_i), -\sum_j P(B_j)\log P(B_j))}$
8	J-Measure (J)	$\max\left(P(A,B)\log(\frac{P(B\|A)}{P(B)}) + P(A\overline{B})\log(\frac{P(\overline{B}\|A)}{P(\overline{B})}),\right.$ $\left. P(A,B)\log(\frac{P(A\|B)}{P(A)}) + P(\overline{A}B)\log(\frac{P(\overline{A}\|B)}{P(\overline{A})})\right)$
9	Gini index (G)	$\max\Big(P(A)[P(B\|A)^2+P(\overline{B}\|A)^2]+P(\overline{A})[P(B\|\overline{A})^2+P(\overline{B}\|\overline{A})^2]$ $-P(B)^2-P(\overline{B})^2,$ $P(B)[P(A\|B)^2+P(\overline{A}\|B)^2]+P(\overline{B})[P(A\|\overline{B})^2+P(\overline{A}\|\overline{B})^2]$ $-P(A)^2-P(\overline{A})^2\Big)$
10	Support (s)	$P(A,B)$
11	Confidence (c)	$\max(P(B\|A), P(A\|B))$
12	Laplace (L)	$\max\left(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2}\right)$
13	Conviction (V)	$\max\left(\frac{P(A)P(\overline{B})}{P(A\overline{B})}, \frac{P(B)P(\overline{A})}{P(B\overline{A})}\right)$
14	Interest (I)	$\frac{P(A,B)}{P(A)P(B)}$
15	cosine (IS)	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
16	Piatetsky-Shapiro's (PS)	$P(A,B)-P(A)P(B)$
17	Certainty factor (F)	$\max\left(\frac{P(B\|A)-P(B)}{1-P(B)}, \frac{P(A\|B)-P(A)}{1-P(A)}\right)$
18	Added Value (AV)	$\max(P(B\|A)-P(B), P(A\|B)-P(A))$
19	Collective strength (S)	$\frac{P(A,B)+P(\overline{AB})}{P(A)P(B)+P(\overline{A})P(\overline{B})} \times \frac{1-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A,B)-P(\overline{AB})}$
20	Jaccard (ζ)	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
21	Klosgen (K)	$\sqrt{P(A,B)}\max(P(B\|A)-P(B), P(A\|B)-P(A))$

Comparing Different Measures

10 examples of contingency tables:

Example	f_{11}	f_{10}	f_{01}	f_{00}
E1	8123	83	424	1370
E2	8330	2	622	1046
E3	9481	94	127	298
E4	3954	3080	5	2961
E5	2886	1363	1320	4431
E6	1500	2000	500	6000
E7	4000	2000	1000	3000
E8	4000	2000	2000	2000
E9	1720	7121	5	1154
E10	61	2483	4	7452

Rankings of contingency tables using various measures:

#	ϕ	λ	α	Q	Y	κ	M	J	G	s	c	L	V	I	IS	PS	F	AV	S	ζ	K
E1	1	1	3	3	3	1	2	2	1	3	5	5	4	6	2	2	4	6	1	2	5
E2	2	2	1	1	1	2	1	3	2	2	1	1	1	8	3	5	1	8	2	3	6
E3	3	3	4	4	4	3	3	8	7	1	4	4	6	10	1	8	6	10	3	1	10
E4	4	7	2	2	2	5	4	1	3	6	2	2	2	4	4	1	2	3	4	5	1
E5	5	4	8	8	8	4	7	5	4	7	9	9	9	3	6	3	9	4	5	6	3
E6	6	6	7	7	7	7	6	4	6	9	8	8	7	2	8	6	7	2	7	8	2
E7	7	5	9	9	9	6	8	6	5	4	7	7	8	5	5	4	8	5	6	4	4
E8	8	9	10	10	10	8	10	10	8	4	10	10	10	9	7	7	10	9	8	7	9
E9	9	9	5	5	5	9	9	7	9	8	3	3	3	7	9	9	3	7	9	9	8
E10	10	8	6	6	6	10	5	9	10	10	6	6	5	1	10	10	5	1	10	10	7

Subjective Interestingness Measure

- Objective measure: Based on statistics
  - Rank patterns based on statistics computed from data
  - e.g., 21 measures of association (support, confidence, Laplace, Gini, mutual information, Jaccard, etc.)

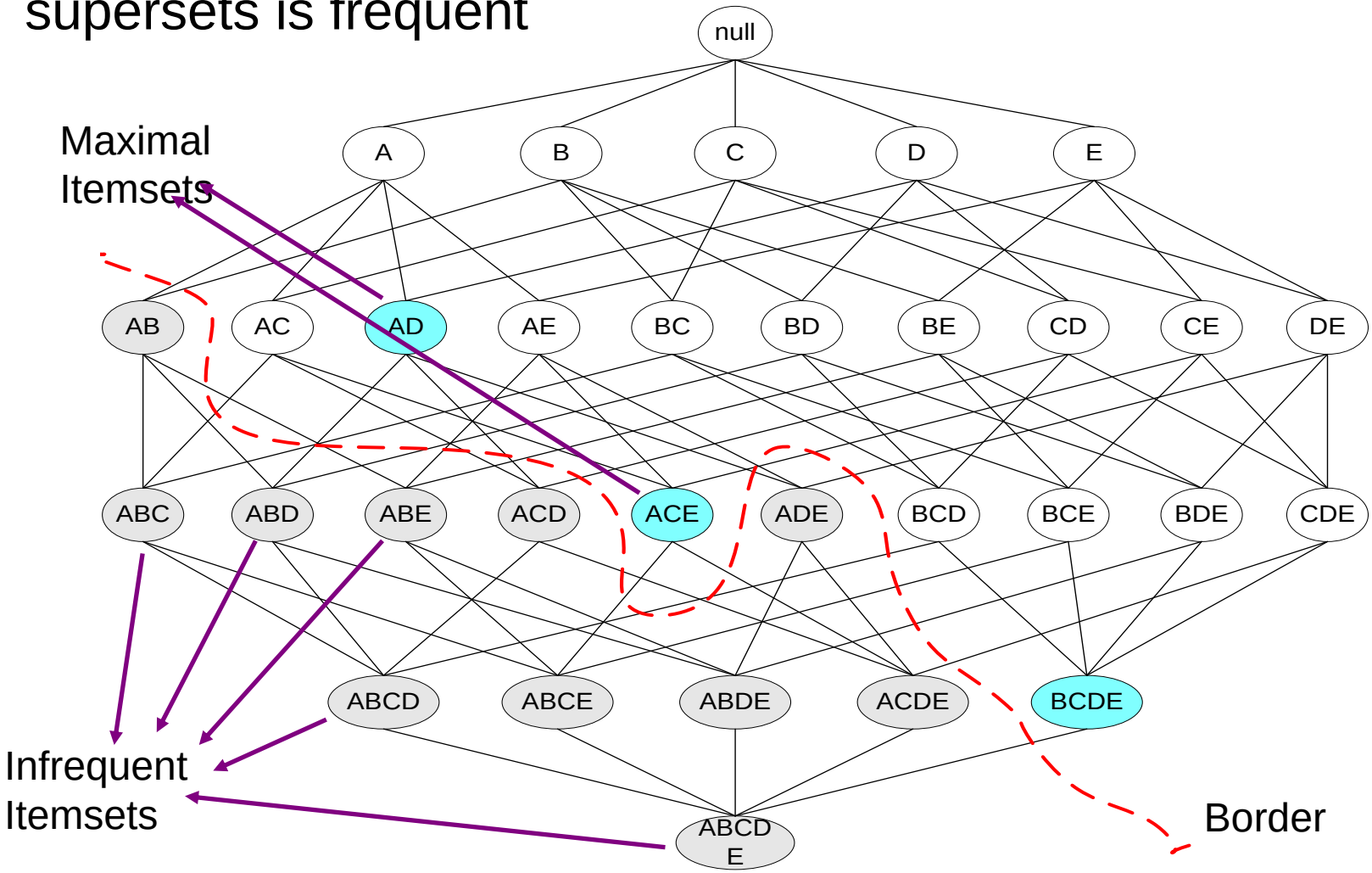
- Subjective measure: Rank patterns according to user's interpretation
  - A pattern is subjectively interesting if it contradicts the expectation of a user (Silberschatz & Tuzhilin)
  - A pattern is subjectively interesting if it is actionable (Silberschatz & Tuzhilin)

Outline

- What is Association Rule Mining?
- Basic Concetps of Association Rules
- The Apriori Algorithm
- The FP-Growth Algorithm
- Which Patterns Are Interesting?
- **Maximal Frequent Patterns and Closed Frequent Patterns**
- Summary

Maximal Frequent Itemset

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



Closed Frequent Itemset

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

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

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

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

Maximal vs Closed Frequent Itemsets

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

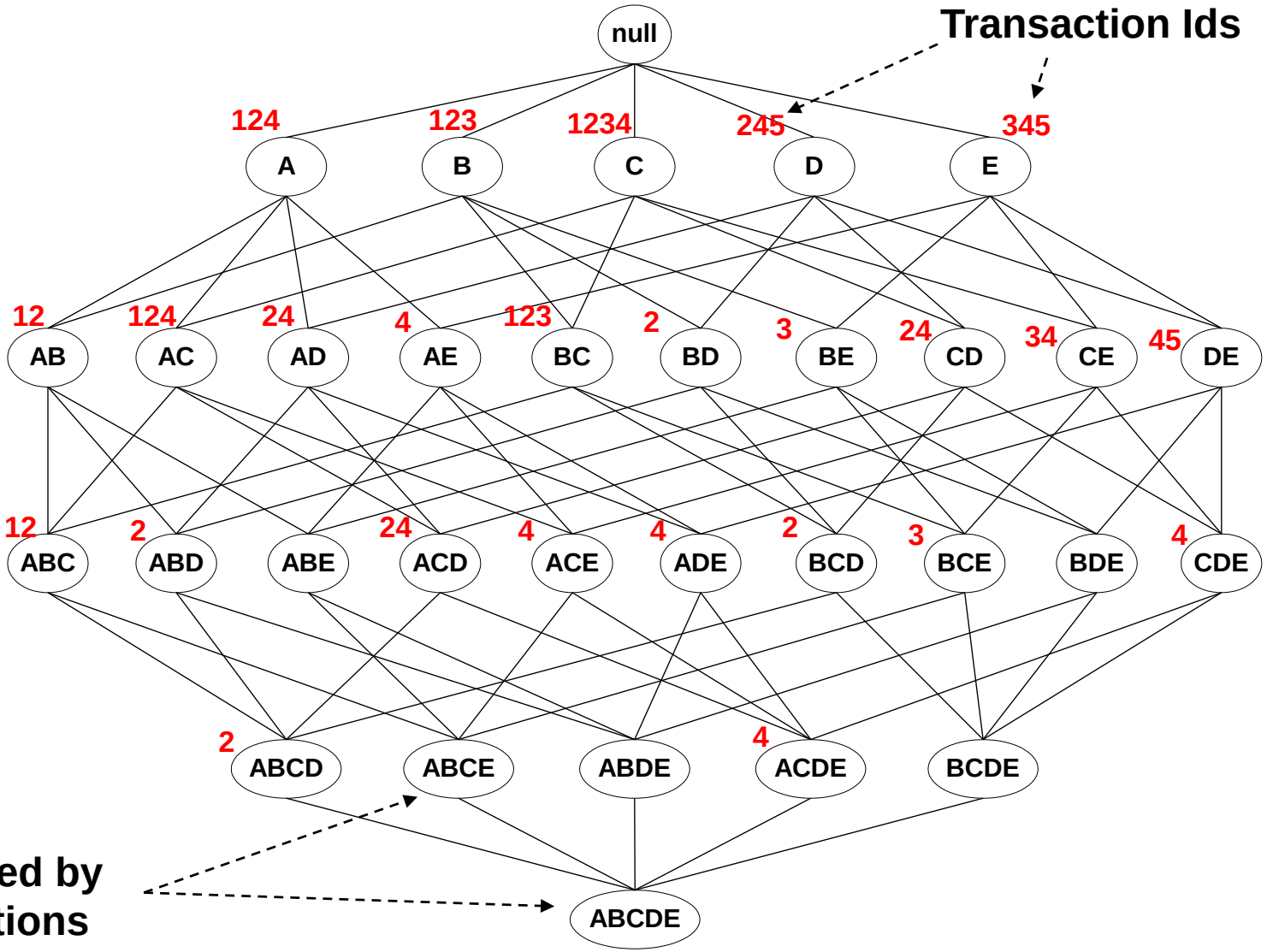


Figure from Lecture Notes for Chapter 6, Introduction to Data Mining, Tan, Steinbach, Kumar

Maximal vs Closed Frequent Itemsets

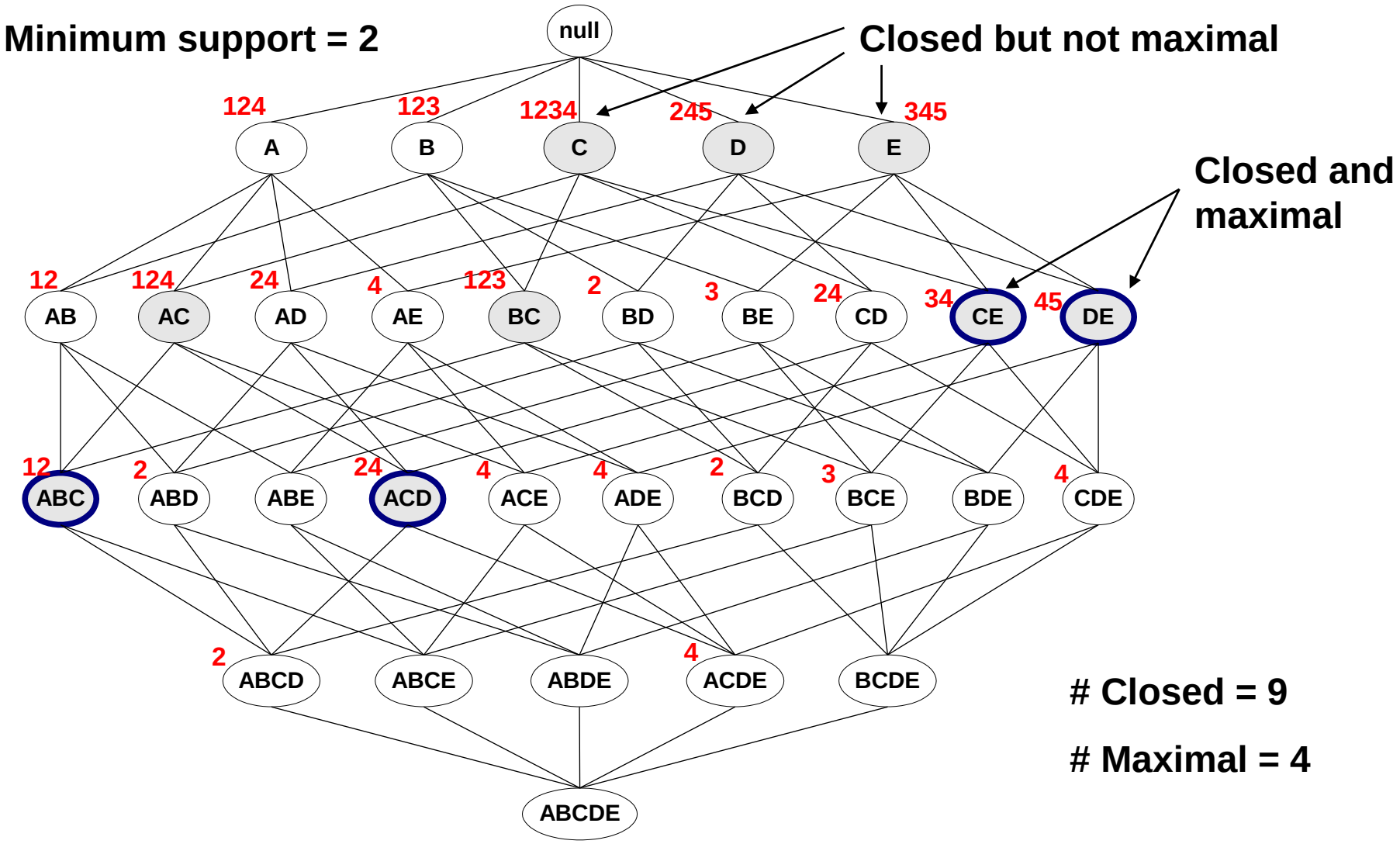


Figure from Lecture Notes for Chapter 6, Introduction to Data Mining, Tan, Steinbach, Kumar

Maximal vs Closed Frequent Itemsets

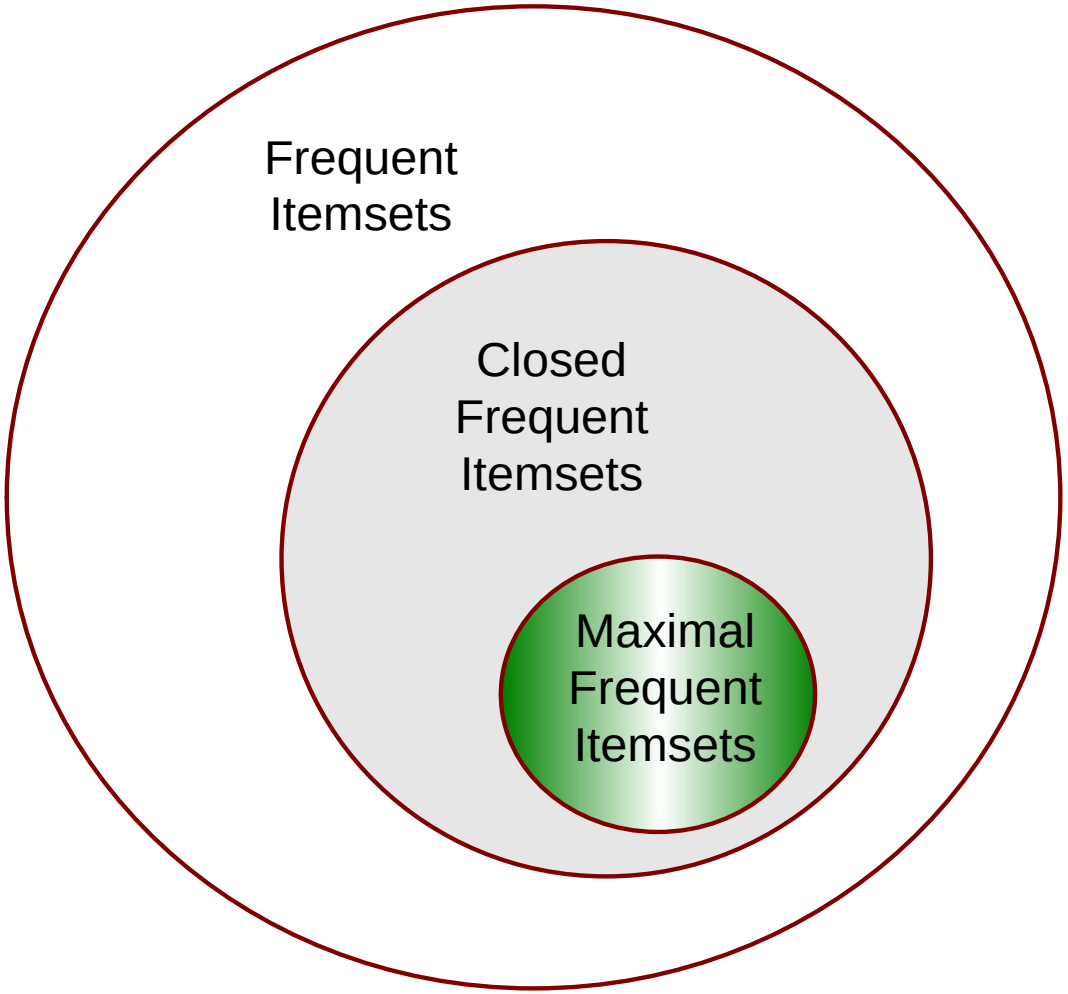


Figure from Lecture Notes for Chapter 6, Introduction to Data Mining, Tan, Steinbach, Kumar

Mining Algorithms for Maximal/Closed Itemsets

- Mining maximal frequent itemsets
  - MaxMiner (1998), Mafia (2001), GenMax (2005), etc.
- Mining closed frequent itemsets
  - Close (1999), Closet (2000), Mafia (2001), Pascal (2000), CHARM (2002), etc.

- Reference: Zaki, Mohammed J. "Mining Closed & Maximal Frequent Itemsets." *NSF CAREER Award IIS-0092978, DOE Early Career Award DE-FG02-02ER25538, NSF grant EIA-0103708* (1).

Outline

- What is Association Rule Mining?
- Basic Concetps of Association Rules
- The Apriori Algorithm
- The FP-Growth Algorithm
- Which Patterns Are Interesting?
- Maximal Frequent Patterns and Closed Frequent Patterns
- Summary

Summary

- Basic concepts: items, transaction, transaction database, frequent pattern, itemset, k-itemset
- Association rule mining problem: goal and key features, rule strength measures (support count, support, confidence)
- The Apriori algorithm: the apriori property, mining all frequent itemsets and generating association rules, algorithm efficiency
- The FP-growth algorithm: mining all frequent itemsets (espcially the FP-tree), algorithm efficiency
- Interestingness measures: Confidence, Lift, Leverage, Conviction
- Closed frequent pattern and maximal frequent pattern