CSE 5243 INTRO. TO DATA MINING

Mining Frequent Patterns and Associations: Basic Concepts

Yu Su, CSE@The Ohio State University

Slides adapted from UIUC CS412 by Prof. Jiawei Han and OSU CSE5243 by Prof. Huan Sun

Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Efficient Pattern Mining Methods
- Pattern Evaluation
- Summary

Pattern Discovery: Basic Concepts

- What Is Pattern Discovery?   Why Is It Important?
- Basic Concepts: Frequent Patterns and Association Rules
- Compressed Representation: Closed Patterns and Max-Patterns

What Is Pattern Discovery?

- Motivating examples:
  - What products were often purchased together?
  - What are the subsequent purchases after buying an iPad?
  - What code segments likely contain copy-and-paste bugs?
  - What word sequences likely form phrases in this corpus?

What Is Pattern Discovery?

- Motivation examples:
  - What products were often purchased together?
  - What are the subsequent purchases after buying an iPad?
  - What code segments likely contain copy-and-paste bugs?
  - What word sequences likely form phrases in this corpus?
- What are patterns?
  - Patterns: A set of items, subsequences, or substructures that occur frequently together (or strongly correlated) in a data set
  - Patterns represent intrinsic and important properties of datasets

What Is Pattern Discovery?

- Motivation examples:
  - What products were often purchased together?
  - What are the subsequent purchases after buying an iPad?
  - What code segments likely contain copy-and-paste bugs?
  - What word sequences likely form phrases in this corpus?

- What are patterns?
  - Patterns: A set of items, subsequences, or substructures that occur frequently together (or strongly correlated) in a data set
  - Patterns represent intrinsic and important properties of datasets
- Pattern discovery: Uncovering patterns from massive data sets

Pattern Discovery: Why Is It Important?

- Finding inherent regularities in a data set
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Mining sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: Discriminative pattern-based analysis
  - Cluster analysis: Pattern-based subspace clustering

Pattern Discovery: Why Is It Important?

- Finding inherent regularities in a data set
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Mining sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: Discriminative pattern-based analysis
  - Cluster analysis: Pattern-based subspace clustering
- Broad applications
  - Market basket analysis, cross-marketing, catalog design, sale campaign analysis, Web log analysis, biological sequence analysis

Basic Concepts: k-Itemsets and Their Supports

- Itemset: A set of one or more items

Basic Concepts: k-Itemsets and Their Supports

- Itemset: A set of one or more items
- k-itemset: $X = \{x_1, \ldots, x_k\}$
  - Ex. {Beer, Nuts, Diaper} is a 3-itemset

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

Basic Concepts: k-Itemsets and Their Supports

- Itemset: A set of one or more items
- k-itemset: $X = \{x_1, \ldots, x_k\}$
  - Ex. {Beer, Nuts, Diaper} is a 3-itemset
- *(absolute) support (count)* of X, sup{X}: Frequency or the number of occurrences of an itemset X
  - Ex. sup{Beer} = 3
  - Ex. sup{Diaper} = 4
  - Ex. sup{Beer, Diaper} = 3
  - Ex. sup{Beer, Eggs} = 1

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

Basic Concepts: k-Itemsets and Their Supports

- Itemset: A set of one or more items
- k-itemset: $X = \{x_1, \ldots, x_k\}$
  - Ex. {Beer, Nuts, Diaper} is a 3-itemset
- *(absolute) support (count)* of X, sup{X}: Frequency or the number of occurrences of an itemset X
  - Ex. sup{Beer} = 3
  - Ex. sup{Diaper} = 4
  - Ex. sup{Beer, Diaper} = 3
  - Ex. sup{Beer, Eggs} = 1

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- *(relative) support*, *s{X}*: The fraction of transactions that contains X (i.e., the probability that a transaction contains X)
  - Ex. s{Beer} = 3/5 = 60%
  - Ex. s{Diaper} = 4/5 = 80%
  - Ex. s{Beer, Eggs} = 1/5 = 20%

Basic Concepts: Frequent Itemsets (Patterns)

- An itemset (or a pattern) X is *frequent* if the support of X is no less than a *minsup* threshold σ

Basic Concepts: Frequent Itemsets (Patterns)

- An itemset (or a pattern) X is *frequent* if the support of X is no less than a *minsup* threshold σ

- Let σ = *50%* (σ: *minsup* threshold)
  For the given 5-transaction dataset
  - All the frequent 1-itemsets:
    - Beer: 3/5 (60%); Nuts: 3/5 (60%)
    - Diaper: 4/5 (80%); Eggs: 3/5 (60%)
  - All the frequent 2-itemsets:
    - {Beer, Diaper}: 3/5 (60%)
  - All the frequent 3-itemsets?
    - None

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

Basic Concepts: Frequent Itemsets (Patterns)

- An itemset (or a pattern) X is *frequent* if the support of X is no less than a *minsup* threshold σ

- Let σ = *50%* (σ: *minsup* threshold)
  For the given 5-transaction dataset
  - All the frequent 1-itemsets:
    - Beer: 3/5 (60%); Nuts: 3/5 (60%)
    - Diaper: 4/5 (80%); Eggs: 3/5 (60%)
  - All the frequent 2-itemsets:
    - {Beer, Diaper}: 3/5 (60%)
  - All the frequent 3-itemsets?
    - None

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- Do these itemsets (shown on the left) form the complete set of frequent *k*-itemsets (patterns) for any *k*?
- **Observation:** We may need an efficient method to mine a complete set of frequent patterns

From Frequent Itemsets to Association Rules

- Comparing with itemsets, rules can be more telling
  - Ex. *Diaper* → *Beer*
    - *Buying diapers may likely lead to buying beers*

From Frequent Itemsets to Association Rules

  - Ex. *Diaper → Beer: Buying diapers may likely lead to buying beers*
- How strong is this rule? (support, confidence)
  - Measuring association rules: $X \rightarrow Y$ (s, c)
    - Both *X* and *Y* are itemsets

From Frequent Itemsets to Association Rules

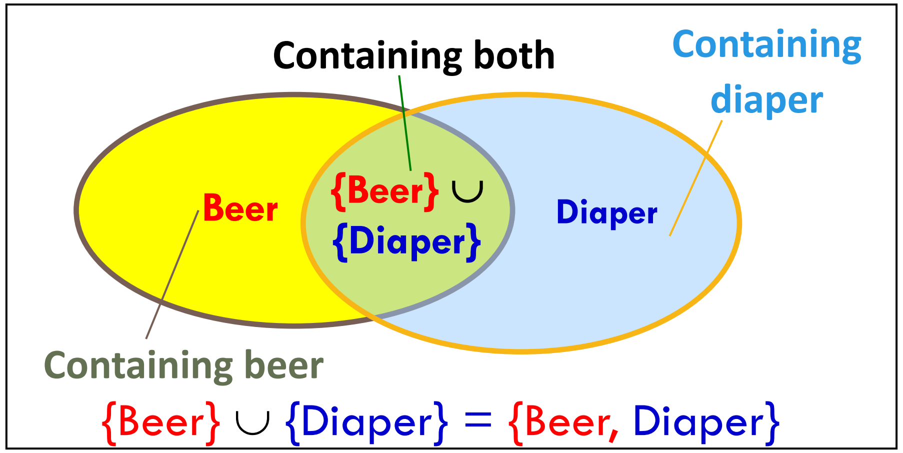
- Ex. *Diaper → Beer: Buying diapers may likely lead to buying beers*
- How strong is this rule? (support, confidence)
  - Measuring association rules: *X* → *Y* (s, c)
    - Both *X* and *Y* are itemsets
  - Support, *s*: The probability that a transaction contains X ∪ Y
    - Ex. s{Diaper, Beer} = 3/5 = 0.6 (i.e., 60%)

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

From Frequent Itemsets to Association Rules

- Ex. *Diaper → Beer : Buying diapers may likely lead to buying beers*
- How strong is this rule? (support, confidence)
  - Measuring association rules: *X* → *Y* (s, c)
    - Both *X* and *Y* are itemsets
  - Support, *s*: The probability that a transaction contains X ∪ Y
    - Ex. s{Diaper, Beer} = 3/5 = 0.6 (i.e., 60%)
  - Confidence, *c: The conditional probability* that a transaction containing X also contains *Y*
    - Calculation: *c* = sup(X ∪ Y) / sup(X)
    - Ex. *c* = sup{Diaper, Beer}/sup{Diaper} = ¾ = 0.75

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



Mining Frequent Itemsets and Association Rules

- **Association rule mining**
  - Given two thresholds: *minsup, minconf*
  - Find all of the rules, $X \rightarrow Y$ (s, c)
    - such that, s ≥ *minsup* and *c* ≥ *minconf*

Mining Frequent Itemsets and Association Rules

- **Association rule mining**
  - Given two thresholds: *minsup, minconf*
  - Find all of the rules, $X \rightarrow Y$ (s, c)
    - such that, s ≥ *minsup* and *c* ≥ *minconf*

- Let *minsup = 50%*
  - Freq. 1-itemsets: Beer: 3, Nuts: 3, Diaper: 4, Eggs: 3
  - Freq. 2-itemsets: {Beer, Diaper}: 3

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

Mining Frequent Itemsets and Association Rules

- **Association rule mining**
  - Given two thresholds: *minsup, minconf*
  - Find all of the rules, $X \rightarrow Y$ (s, c)
    - such that, $s \geq minsup$ and $c \geq minconf$

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- Let *minsup = 50%*
  - Freq. 1-itemsets: Beer: 3, Nuts: 3, Diaper: 4, Eggs: 3
  - Freq. 2-itemsets: {Beer, Diaper}: 3

- Let *minconf = 50%*
  - *Beer* → *Diaper* (60%, 100%)
  - *Diaper* → *Beer* (60%, 75%)

Mining Frequent Itemsets and Association Rules

- **Association rule mining**
  - Given two thresholds: *minsup, minconf*
  - Find all of the rules, $X \rightarrow Y$ (s, c)
    - such that, $s \geq$ *minsup* and $c \geq$ *minconf*

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- Let *minsup = 50%*
  - Freq. 1-itemsets: Beer: 3, Nuts: 3, Diaper: 4, Eggs: 3
  - Freq. 2-itemsets: {Beer, Diaper}: 3

- Let *minconf = 50%*
  - *Beer* $\rightarrow$ *Diaper* (60%, 100%)
  - *Diaper* $\rightarrow$ *Beer* (60%, 75%)

(Q: Are these all rules?)

Mining Frequent Itemsets and Association Rules

- **Association rule mining**
  - Given two thresholds: *minsup, minconf*
  - Find all of the rules, $X \rightarrow Y$ (s, c)
    - such that, s ≥ *minsup* and *c* ≥ *minconf*

- Let *minsup* = *50%*
  - Freq. 1-itemsets: Beer: 3, Nuts: 3, Diaper: 4, Eggs: 3
  - Freq. 2-itemsets: {Beer, Diaper}: 3

- Let *minconf* = *50%*
  - *Beer* → *Diaper* (60%, 100%)
  - *Diaper* → *Beer* (60%, 75%)

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- **Observations:**
  - Mining association rules and mining frequent patterns are very close problems
  - Scalable methods are needed for mining large datasets

Association Rule Mining: two-step process

In general, association rule mining can be viewed as a two-step process:

1. **Find all frequent itemsets:** By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, *min_sup*.

2. **Generate strong association rules from the frequent itemsets:** By definition, these rules must satisfy minimum support and minimum confidence.

Because the second step is much less costly than the first, the overall performance of mining association rules is determined by the first step.

Generating Association Rules from Frequent Patterns

- Recall that:

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

- Once we mined frequent patterns, association rules can be generated as follows:

  - For each frequent itemset l, generate all nonempty subsets of l.
  - For every nonempty subset s of l, output the rule "$s \Rightarrow (l - s)$" if $\frac{support\_count(l)}{support\_count(s)} \geq min\_conf$, where min_conf is the minimum confidence threshold.

Generating Association Rules from Frequent Patterns

- Recall that:

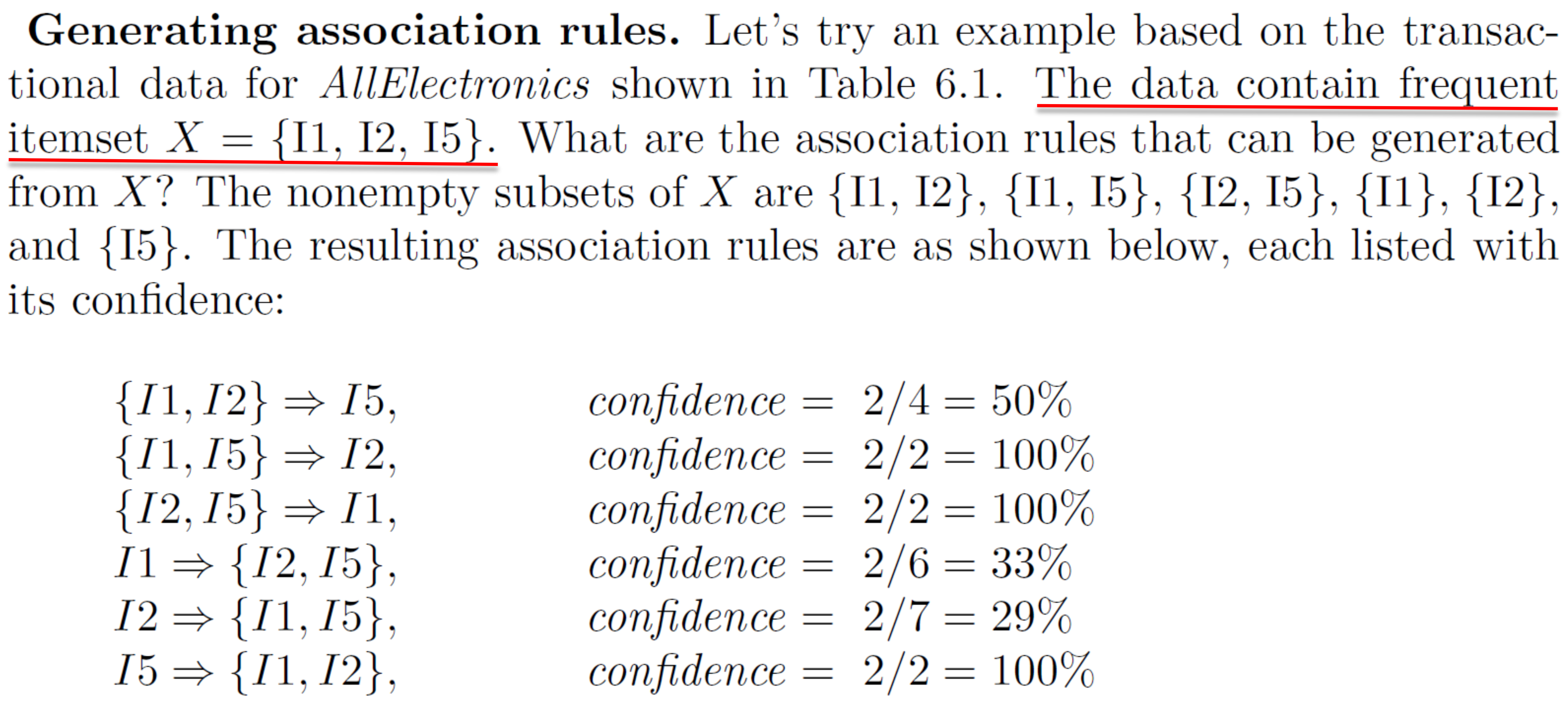
$$confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}$$

- Once we mined frequent patterns, association rules can be generated as follows:

  - For each frequent itemset l, generate all nonempty subsets of l.
  - For every nonempty subset s of l, output the rule "$s \Rightarrow (l - s)$" if $\frac{support\_count(l)}{support\_count(s)} \geq min\_conf$, where min_conf is the minimum confidence threshold.

Because l is a frequent itemset, each rule automatically satisfies the minimum support requirement.

Example: Generating Association Rules



Generating association rules. Let's try an example based on the transactional data for *AllElectronics* shown in Table 6.1. The data contain frequent itemset $X = \{\text{I1, I2, I5}\}$. What are the association rules that can be generated from X? The nonempty subsets of X are {I1, I2}, {I1, I5}, {I2, I5}, {I1}, {I2}, and {I5}. The resulting association rules are as shown below, each listed with its confidence:

$$\begin{array}{ll}
\{I1, I2\} \Rightarrow I5, & \mathit{confidence} = \; 2/4 = 50\% \\
\{I1, I5\} \Rightarrow I2, & \mathit{confidence} = \; 2/2 = 100\% \\
\{I2, I5\} \Rightarrow I1, & \mathit{confidence} = \; 2/2 = 100\% \\
I1 \Rightarrow \{I2, I5\}, & \mathit{confidence} = \; 2/6 = 33\% \\
I2 \Rightarrow \{I1, I5\}, & \mathit{confidence} = \; 2/7 = 29\% \\
I5 \Rightarrow \{I1, I2\}, & \mathit{confidence} = \; 2/2 = 100\%
\end{array}$$

Example from Chapter 6

If minimum confidence threshold: 70%, what will be output?

Challenge: There Are Too Many Frequent Patterns!

- A long pattern contains a combinatorial number of sub-patterns
- How many frequent itemsets does the following TDB_1 contain?
  - $TDB_{1:}$ T_1: $\{a_1, …, a_{50}\}$; T_2: $\{a_1, …, a_{100}\}$
  - Assuming (absolute) *minsup* = 1
  - Let's give it a try…

1-itemsets: $\{a_1\}$: 2, $\{a_2\}$: 2, …, $\{a_{50}\}$: 2, $\{a_{51}\}$: 1, …, $\{a_{100}\}$: 1,

2-itemsets: $\{a_1, a_2\}$: 2, …, $\{a_1, a_{50}\}$: 2, $\{a_1, a_{51}\}$: 1 …, …, $\{a_{99}, a_{100}\}$: 1,

…, …, …, …

99-itemsets: $\{a_1, a_2, …, a_{99}\}$: 1, …, $\{a_2, a_3, …, a_{100}\}$: 1

100-itemset: $\{a_1, a_2, …, a_{100}\}$: 1

Challenge: There Are Too Many Frequent Patterns!

- A long pattern contains a combinatorial number of sub-patterns
- How many frequent itemsets does the following TDB_1 contain?
  - $TDB_{1:}$ T_1: {a_1, …, a_{50}}; T_2: {a_1, …, a_{100}}
  - Assuming (absolute) *minsup* = 1
  - Let's give it a try…

1-itemsets: {a_1}: 2, {a_2}: 2, …, {a_{50}}: 2, {a_{51}}: 1, …, {a_{100}}: 1,

2-itemsets: {a_1, a_2}: 2, …, {a_1, a_{50}}: 2, {a_1, a_{51}}: 1 …, …, {a_{99}, a_{100}}: 1,

…, …, …, …

99-itemsets: {a_1, a_2, …, a_{99}}: 1, …, {a_2, a_3, …, a_{100}}: 1

100-itemset: {a_1, a_2, …, a_{100}}: 1

- The total number of frequent itemsets:

$$\binom{100}{1} + \binom{100}{2} + \binom{100}{3} + \cdots + \binom{100}{100} = 2^{100} - 1$$

Too huge a set for any one to compute or store!

Expressing Patterns in Compressed Form: Closed Patterns

- How to handle such a challenge?
- Solution 1: **Closed patterns:** A pattern (itemset) X is closed if X is *frequent*, and there exists *no super-pattern* Y ⊃ X, *with the same support* as X

Expressing Patterns in Compressed Form: Closed Patterns

- How to handle such a challenge?
- Solution 1: **Closed patterns:** A pattern (itemset) X is closed if X is *frequent*, and there exists *no super-pattern* Y ⊃ X, *with the same support* as X
  - Let Transaction DB TDB_1: T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - Suppose *minsup* = 1. How many closed patterns does TDB_1 contain?
    - Two: P_1: "$\{a_1, \ldots, a_{50}\}$: 2"; P_2: "$\{a_1, \ldots, a_{100}\}$: 1"

Why?

Expressing Patterns in Compressed Form: Closed Patterns

- How to handle such a challenge?
- Solution 1: **Closed patterns:** A pattern (itemset) X is closed if X is *frequent*, and there exists *no super-pattern* Y ⊃ X, *with the same support* as X
  - Let Transaction DB TDB_1: T_1: $\{a_1, …, a_{50}\}$; T_2: $\{a_1, …, a_{100}\}$
  - Suppose *minsup* = 1. How many closed patterns does TDB_1 contain?
    - Two: P_1: "$\{a_1, …, a_{50}\}$: 2"; P_2: "$\{a_1, …, a_{100}\}$: 1"
- Closed pattern is a lossless compression of frequent patterns
  - Reduces the # of patterns but does not lose the support information!
  - You will still be able to say: "$\{a_2, …, a_{40}\}$: 2", "$\{a_5, a_{51}\}$: 1"

Expressing Patterns in Compressed Form: Max-Patterns

- Solution 2: **Max-patterns:** A pattern X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ⊃ X

Expressing Patterns in Compressed Form: Max-Patterns

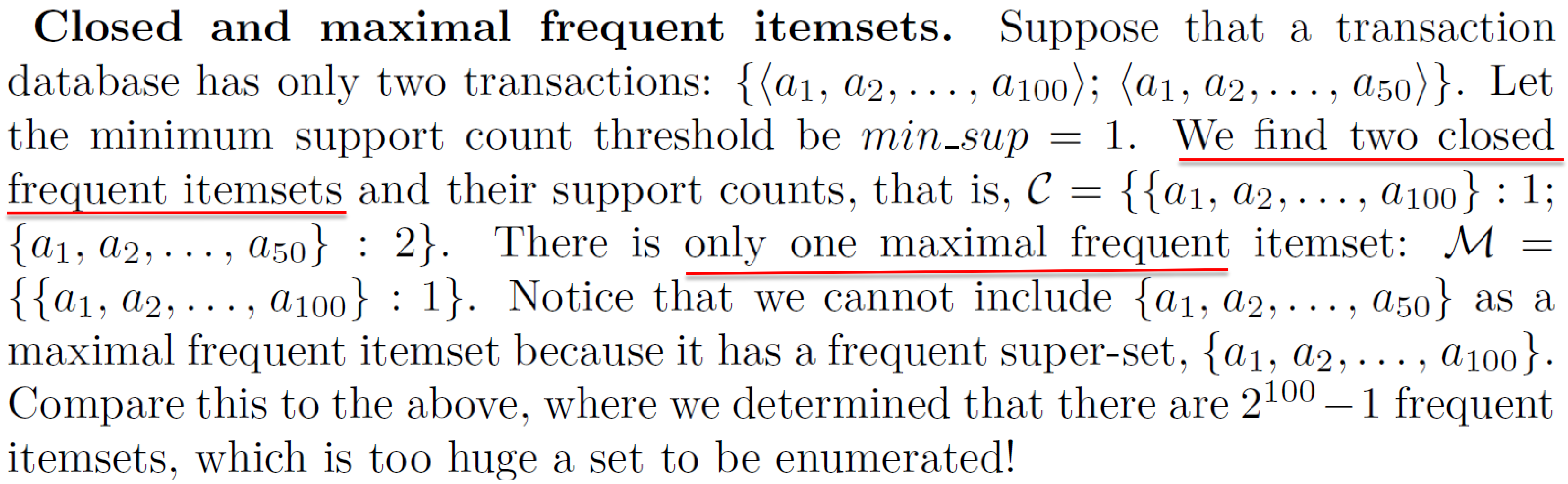
- Solution 2: **Max-patterns:** A pattern X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ⊃ X
- Difference with closed-patterns?
  - Do not care about the real support of the sub-patterns of a max-pattern
  - Let Transaction DB TDB$_1$: T$_1$: $\{a_1, \ldots, a_{50}\}$; T$_2$: $\{a_1, \ldots, a_{100}\}$
  - Suppose *minsup* = 1. How many max-patterns does TDB$_1$ contain?
    - One: P: "$\{a_1, \ldots, a_{100}\}$: 1"

Why?

Expressing Patterns in Compressed Form: Max-Patterns

- Solution 2: **Max-patterns:** A pattern X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ⊃ X
- Difference with closed-patterns?
  - Do not care about the real support of the sub-patterns of a max-pattern
  - Let Transaction DB TDB$_1$: T$_1$: {a_1, …, a_{50}}; T$_2$: {a_1, …, a_{100}}
  - Suppose *minsup* = 1. How many max-patterns does TDB$_1$ contain?
    - One: P: "{a_1, ..., a_{100}}: 1"
- Max-pattern is a lossy compression!
  - We only know {a_1, …, a_{40}} is frequent
  - But we do not know the real support of {a_1, …, a_{40}}, …, any more!
  - Thus in many applications, closed-patterns are more desirable than max-patterns

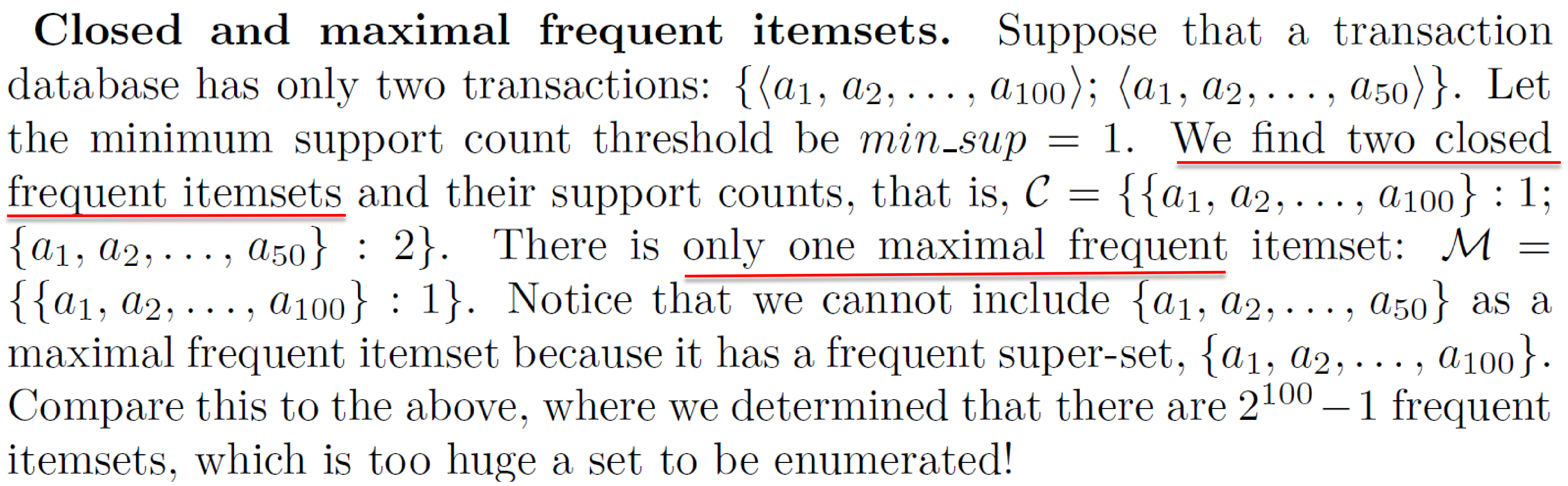
Example



Closed and maximal frequent itemsets. Suppose that a transaction database has only two transactions: $\{\langle a_1, a_2, \ldots, a_{100}\rangle; \langle a_1, a_2, \ldots, a_{50}\rangle\}$. Let the minimum support count threshold be $min\_sup = 1$. We find two closed frequent itemsets and their support counts, that is, $\mathcal{C} = \{\{a_1, a_2, \ldots, a_{100}\} : 1; \{a_1, a_2, \ldots, a_{50}\} : 2\}$. There is only one maximal frequent itemset: $\mathcal{M} = \{\{a_1, a_2, \ldots, a_{100}\} : 1\}$. Notice that we cannot include $\{a_1, a_2, \ldots, a_{50}\}$ as a maximal frequent itemset because it has a frequent super-set, $\{a_1, a_2, \ldots, a_{100}\}$. Compare this to the above, where we determined that there are $2^{100} - 1$ frequent itemsets, which is too huge a set to be enumerated!

{all frequent patterns} ⊇ {closed frequent patterns} ⊇ {max frequent patterns}

Example



Closed and maximal frequent itemsets. Suppose that a transaction database has only two transactions: $\{\langle a_1, a_2, \ldots, a_{100}\rangle; \langle a_1, a_2, \ldots, a_{50}\rangle\}$. Let the minimum support count threshold be $min\_sup = 1$. We find two closed frequent itemsets and their support counts, that is, $\mathcal{C} = \{\{a_1, a_2, \ldots, a_{100}\} : 1; \{a_1, a_2, \ldots, a_{50}\} : 2\}$. There is only one maximal frequent itemset: $\mathcal{M} = \{\{a_1, a_2, \ldots, a_{100}\} : 1\}$. Notice that we cannot include $\{a_1, a_2, \ldots, a_{50}\}$ as a maximal frequent itemset because it has a frequent super-set, $\{a_1, a_2, \ldots, a_{100}\}$. Compare this to the above, where we determined that there are $2^{100} - 1$ frequent itemsets, which is too huge a set to be enumerated!

The set of closed-patterns contains complete information regarding the frequent itemsets.

Quiz

- Given closed frequent itemsets:

$$C = \{\ \{a1, a2, \ldots, a100\}: 1; \quad \{a1, a2, \ldots, a50\}: 2\ \}$$

Is {a2, a45} frequent? Can we know its support?

Quiz (Cont'd)

- Given maximal frequent itemset:

  $M = \left\{ \{a1, a2, \ldots, a100\}: 1 \right\}$

What is the support of {a8, a55}?

Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Efficient Pattern Mining Methods
  - The Apriori Algorithm
  - Application in Classification
- Pattern Evaluation
- Summary

Efficient Pattern Mining Methods

- The Downward Closure Property of Frequent Patterns
- The Apriori Algorithm
- Extensions or Improvements of Apriori
- Mining Frequent Patterns by Exploring Vertical Data Format
- FPGrowth: A Frequent Pattern-Growth Approach
- Mining Closed Patterns

The Downward Closure Property of Frequent Patterns

- Observation: From TDB$_{1:}$ T$_1$: {a_1, …, a_{50}}; T$_2$: {a_1, …, a_{100}}
  - We get a frequent itemset: {a_1, …, a_{50}}
  - Also, its subsets are all frequent: {a_1}, {a_2}, …, {a_{50}}, {a_1, a_2}, …, {a_1, …, a_{49}}, …
  - There must be some hidden relationships among frequent patterns!

The Downward Closure Property of Frequent Patterns

- Observation: From TDB$_{1:}$ T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - We get a frequent itemset: $\{a_1, \ldots, a_{50}\}$
  - Also, its subsets are all frequent: $\{a_1\}, \{a_2\}, \ldots, \{a_{50}\}, \{a_1, a_2\}, \ldots, \{a_1, \ldots, a_{49}\}, \ldots$
  - There must be some hidden relationships among frequent patterns!
- The downward closure (also called "Apriori") property of frequent patterns
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - Every transaction containing {beer, diaper, nuts} also contains {beer, diaper}
  - Apriori: Any subset of a frequent itemset must be frequent

A sharp knife for pruning!

The Downward Closure Property of Frequent Patterns

- Observation: From TDB$_1$: T$_1$: $\{a_1, \ldots, a_{50}\}$; T$_2$: $\{a_1, \ldots, a_{100}\}$
  - We get a frequent itemset: $\{a_1, \ldots, a_{50}\}$
  - Also, its subsets are all frequent: $\{a_1\}, \{a_2\}, \ldots, \{a_{50}\}, \{a_1, a_2\}, \ldots, \{a_1, \ldots, a_{49}\}, \ldots$
  - There must be some hidden relationships among frequent patterns!
- The downward closure (also called "Apriori") property of frequent patterns
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - Every transaction containing {beer, diaper, nuts} also contains {beer, diaper}
  - Apriori: Any subset of a frequent itemset must be frequent

A sharp knife for pruning!

- Efficient mining methodology
  - If any subset of an itemset S is infrequent, then there is no chance for S to be frequent—why do we even have to consider S ?!

Apriori Pruning and Scalable Mining Methods

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not even be generated!
  - (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Scalable mining Methods: Three major approaches
  - Level-wise, join-based approach:
    - Apriori (Agrawal & Srikant@VLDB'94)
  - Vertical data format approach:
    - Eclat (Zaki, Parthasarathy, Ogihara, Li @KDD'97)
  - Frequent pattern projection and growth:
    - FPgrowth (Han, Pei, Yin @SIGMOD'00)

Apriori: A Candidate Generation & Test Approach

- Outline of Apriori (level-wise, candidate generation and test)
  - Initially, scan DB once to get frequent 1-itemset
  - Repeat
    - Generate length-(k+1) candidate itemsets from length-k frequent itemsets
    - Test the candidates against DB to find frequent (k+1)-itemsets
    - Set k := k +1
  - Until no frequent or candidate set can be generated
  - Return all the frequent itemsets derived

The Apriori Algorithm (Pseudo-Code)

```
$C_k$: Candidate itemset of size k
$F_k$ : Frequent itemset of size k

k := 1;
$F_k$ := {frequent items};   // frequent 1-itemset
While ($F_k$ != $\emptyset$) do {   // when $F_k$ is non-empty
    $C_{k+1}$ := candidates generated from $F_k$;  // candidate generation
    Derive $F_{k+1}$ by counting candidates in $C_{k+1}$ with respect to TDB at minsup;
    k := k + 1
    }
return $\cup_k F_k$                // return $F_k$ generated at each level
```

The Apriori Algorithm—An Example

Database TDB

minsup = 2

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

F_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

F_2

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

C_3

Itemset
{B, C, E}

3rd scan

F_3

Itemset	sup
{B, C, E}	2

The Apriori Algorithm—An Example

Database TDB

minsup = 2

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

F_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

F_2

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

C_3

Itemset
{B, C, E}

3rd scan

F_3

Itemset	sup
{B, C, E}	2

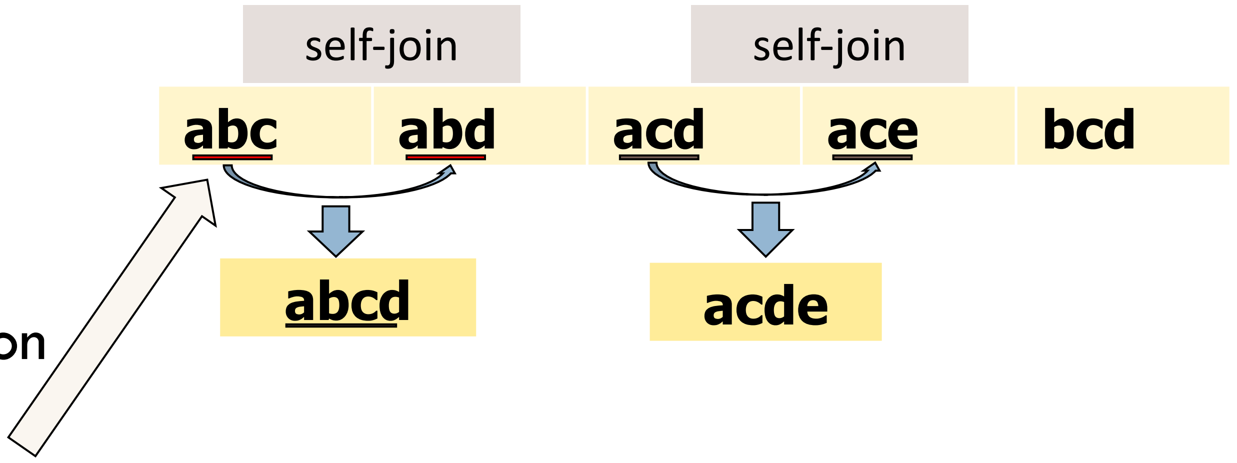
Why?

Apriori: Implementation Tricks

- How to generate candidates?
  - Step 1: self-joining F_k
  - Step 2: pruning

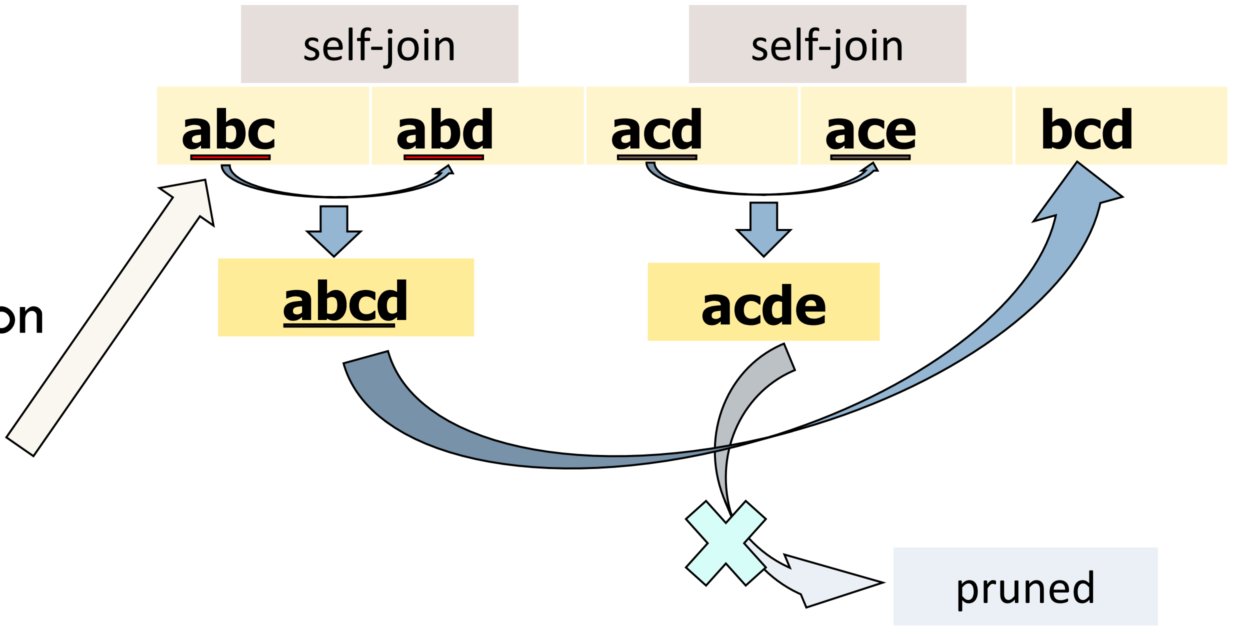
Apriori: Implementation Tricks

- How to generate candidates?
  - Step 1: self-joining F_k
  - Step 2: pruning
- Example of candidate-generation
  - $F_3 = \{abc, abd, acd, ace, bcd\}$
  - Self-joining: $F_3 * F_3$
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*



Apriori: Implementation Tricks

- How to generate candidates?
  - Step 1: self-joining F_k
  - Step 2: pruning
- Example of candidate-generation
  - $F_3 = \{abc, abd, acd, ace, bcd\}$
  - Self-joining: $F_3 * F_3$
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in F_3
  - $C_4 = \{abcd\}$



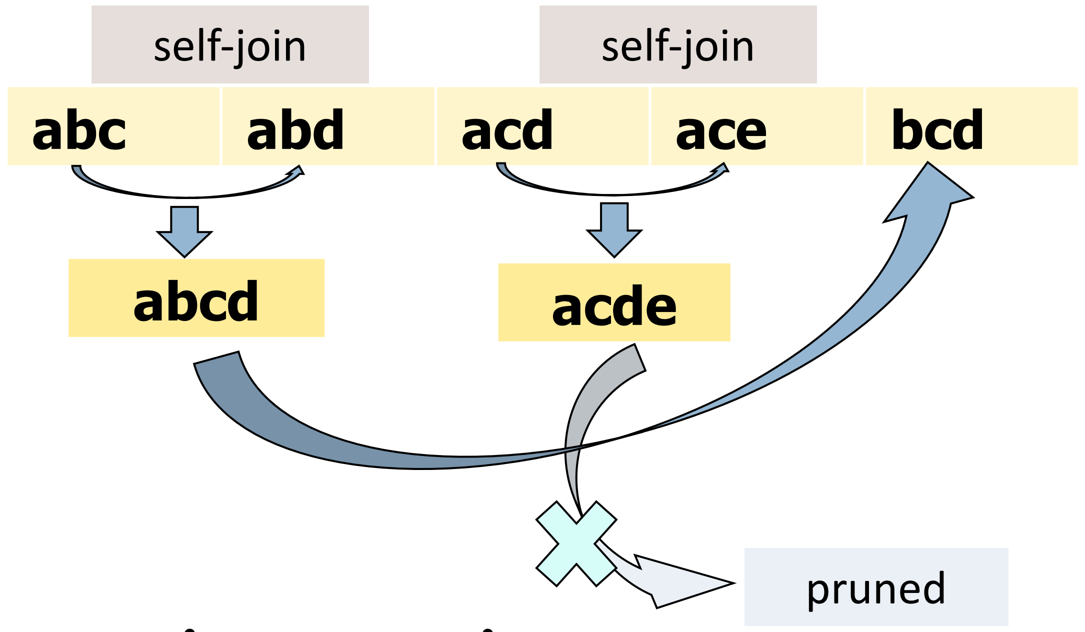
Candidate Generation: An SQL Implementation

- Suppose the items in F_{k-1} are listed in an order
- Step 1: self-joining F_{k-1}

```
insert into C_k
select p.item_1, p.item_2, …, p.item_{k-1}, q.item_{k-1}
from F_{k-1} as p, F_{k-1} as q
where p.item_1 = q.item_1, …, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}
```

- Step 2: pruning

```
for all itemsets c in C_k do
  for all (k-1)-subsets s of c do
    if (s is not in F_{k-1}) then delete c from C_k
```



Apriori Adv/Disadv

- ***Advantages:***
  - Uses large itemset property
  - Easily parallelized
  - Easy to implement

- ***Disadvantages:***
  - Assumes transaction database is memory resident
  - Requires up to m database scans

Classification based on Association Rules (CBA)

- Why?
  - Can effectively uncover the correlation structure in data
  - AR are typically quite scalable in practice
  - Rules are often very intuitive
    - Hence classifier built on intuitive rules is easier to interpret

- When to use?
  - On large dynamic datasets where class labels are available and the correlation structure is unknown.
  - Multi-class categorization problems
  - E.g. Web/Text Categorization, Network Intrusion Detection

Classification based on Association Rules (CBA)

- Input
  - <feature vector> <class label(s)>
  - <feature vector> = w1,…,wN
  - <class label(s)> = c1,…,cM

e.g., text categorization

- Run AR with minsup and minconf
  - Prune rules of form
    - w1 → w2, [w1,c2] → c3 etc.
  - Keep only rules satisfying the constraints:
    - W → C (Left: only composed of w1,…wN and Right: only composed of c1,…cM)

CBA: Text Categorization (cont.)

- Order remaining rules
  - By confidence
    - 100%
      - R1: W1 $\rightarrow$ C1 (support 40%)
      - R2: W4 $\rightarrow$ C2 (support 60%)
    - 95%
      - R3: W3 $\rightarrow$ C2 (support 30%)
      - R4: W5 $\rightarrow$ C4 (support 70%)
  - And within each confidence level by support
    - Ordering R2, R1, R4, R3

Classification based Association

CBA: Text Categorization (cont.)

- Take training data and evaluate the predictive ability of each rule, <u>prune rules that are subsumed by superior rules</u>
  - T1: W1 W5 C1,C4
  - T2: W2 W4 C2
  - T3: W3 W4 C2
  - T4: W5 W8 C4

Note: only a subset of transactions in training data

- Rule R3 would be pruned in this example if it is always subsumed by Rule R2

R3: W3 → C2

R2: W4 → C2

Why?

CBA: Text Categorization (cont.)

- Take training data and evaluate the predictive ability of each rule, <u>prune rules that are subsumed by superior rules</u>
  - T1: W1 W5 C1,C4
  - T2: W2 W4 C2
  - T3: W3 W4 C2
  - T4: W5 W8 C4

Note: only a subset of transactions in training data

- Rule R3 would be pruned in this example if it is always subsumed by Rule R2

{T3} is predictable by R3: W3 → C2

{T2, T3} is predictable by R2: W4 → C2

R3 is subsumed by R2, and will therefore be pruned.

Formal Concepts of Model

- Given two rules r_i and r_j, define: $r_i \succ r_j$ if
  - The confidence of r_i is greater than that of r_j, or
  - Their confidences are the same, but the support of r_i is greater than that of r_j, or
  - Both the confidences and supports are the same, but r_i is generated earlier than r_j.

- Our classifier model is of the following format:

  $<r_1, r_2, \ldots, r_n, \text{default\_class}>$,

  where $r_i \in R$, $r_a \succ r_b$ if $b > a$

- Other models possible
  - Sort by length of antecedent

Using the CBA model to classify

- For a new transaction
  - W1, W3, W5
  - Pick the k-most confident rules that apply (using the precedence ordering established in the baseline model)
  - The resulting classes are the predictions for this transaction
    - If k = 1 you would pick ?
    - If k = 2 you would pick ?

- Conf: 100%
  - R1: W1 $\rightarrow$ C1 (support 40%)
  - R2: W4 $\rightarrow$ C2 (support 60%)
- Conf: 95%
  - R3: W3 $\rightarrow$ C2 (support 30%)
  - R4: W5 $\rightarrow$ C4 (support 70%)

Using the CBA model to classify

- For a new transaction
  - W1, W3, W5
  - Pick the k-most confident rules that apply (using the precedence ordering established in the baseline model)
  - The resulting classes are the predictions for this transaction
    - If k = 1 you would pick C1
    - If k = 2 you would pick C1, C4 (multi-class)
  - If W9, W10 (not covered by any rule), you would pick C2 (assuming it's the default, most dominant class)
  - Accuracy measurements as before (Classification Error)

CBA: Procedural Steps

- Preprocessing, Training and Testing data split

- Compute AR on Training data
  - Keep only rules of form X→ C
    - C is class label itemset and X is feature itemset

- Order AR
  - According to confidence
  - According to support (at each confidence level)

- Prune away rules that lack sufficient predictive ability on training data (starting top-down)
  - Rule subsumption

- For data that is not predictable, pick most dominant class as default class

- Test on testing data and report accuracy

Apriori: Improvements and Alternatives

- Reduce passes of transaction database scans
  - Partitioning (e.g., Savasere, et al., 1995)
  - Dynamic itemset counting (Brin, et al., 1997)

To be discussed in subsequent slides

- Shrink the number of candidates
  - Hashing (e.g., DHP: Park, et al., 1995)
  - Pruning by support lower bounding (e.g., Bayardo 1998)
  - Sampling (e.g., Toivonen, 1996)

To be discussed in subsequent slides

- Exploring special data structures
  - Tree projection (Agarwal, et al., 2001)
  - H-miner (Pei, et al., 2001)
  - Hypecube decomposition (e.g., LCM: Uno, et al., 2004)

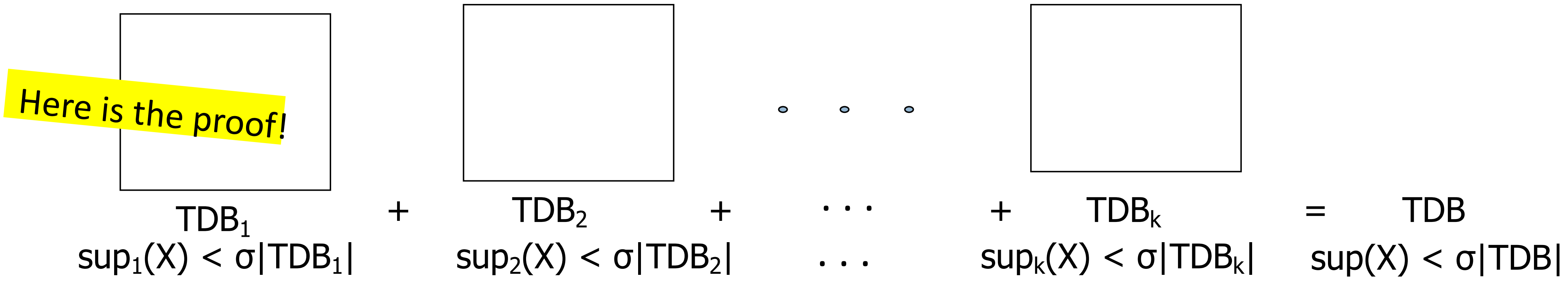
<1> Partitioning: Scan Database Only Twice

- Theorem: *Any itemset that is potentially frequent in TDB must be frequent in at least one of the partitions of TDB*

Why?

<1> Partitioning: Scan Database Only Twice

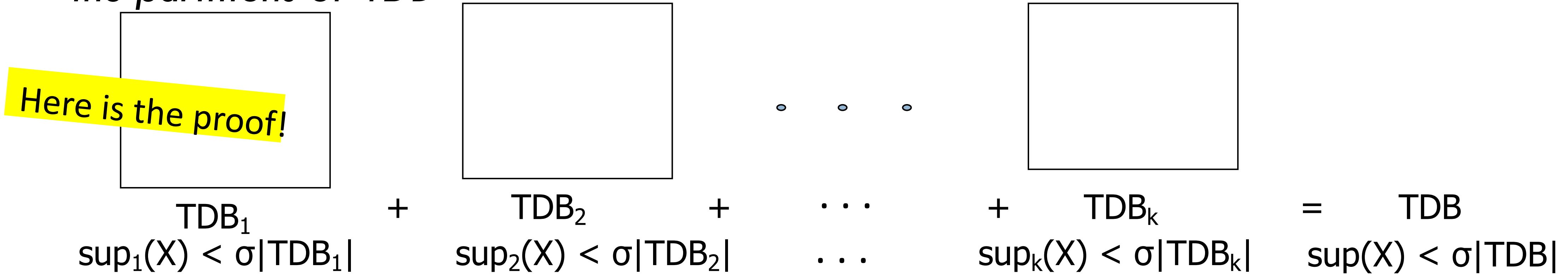
- Theorem: *Any itemset that is potentially frequent in TDB must be frequent in at least one of the partitions of TDB*


Here is the proof!
TDB_1
$sup_1(X) < \sigma|TDB_1|$
+
TDB_2
$sup_2(X) < \sigma|TDB_2|$
+
. . .
. . .
+
TDB_k
$sup_k(X) < \sigma|TDB_k|$
=
TDB
$sup(X) < \sigma|TDB|$

Proof by contradiction

<1> Partitioning: Scan Database Only Twice

- Theorem: *Any itemset that is potentially frequent in TDB must be frequent in at least one of the partitions of TDB*



$$TDB_1 + TDB_2 + \cdots + TDB_k = TDB$$

$$sup_1(X) < \sigma|TDB_1| \quad sup_2(X) < \sigma|TDB_2| \quad \ldots \quad sup_k(X) < \sigma|TDB_k| \quad sup(X) < \sigma|TDB|$$

- Method: Scan DB twice (A. Savasere, E. Omiecinski and S. Navathe, *VLDB'95*)
  - Scan 1: Partition database so that each partition can fit in main memory
    - Mine local frequent patterns in this partition
  - Scan 2: Consolidate global frequent patterns
    - Find global frequent itemset candidates (those frequent in at least one partition)
    - Find the true frequency of those candidates, by scanning TDB_i one more time

<2> Direct Hashing and Pruning (DHP):

- Reduce candidate number: (J. Park, M. Chen, and P. Yu, SIGMOD'95)
- Hashing: Different itemsets may have the same hash value: v = *hash*(itemset)
- 1st scan: When counting the 1-itemset, hash 2-itemset to calculate the bucket count
- Observation: A *k*-itemset cannot be frequent if its corresponding hashing bucket count is below the *minsup* threshold
- Example: At the 1st scan of TDB, count 1-itemset, and
  - Hash 2-itemsets in each transaction to its bucket
    - {ab, ad, ce}
    - {bd, be, de}
    - …
  - At the end of the first scan,
    - *if minsup = 80, remove ab, ad, ce, since count*{*ab, ad, ce*} < 80

Itemsets	Count
{*ab, ad, ce*}	35
{*bd, be, de*}	298
……	…
{*yz, qs, wt*}	58

Hash Table

<2> Direct Hashing and Pruning (DHP)

- DHP (Direct Hashing and Pruning): (J. Park, M. Chen, and P. Yu, SIGMOD'95)
- Hashing: Different itemsets may have the same hash value: v = *hash*(itemset)
- 1st scan: When counting the 1-itemset, hash 2-itemset to calculate the bucket count
- Observation: A *k*-itemset cannot be frequent if its corresponding hashing bucket count is below the *minsup* threshold
- Example:

Create hash table H_2 using hash function $h(x, y) = ((\mathit{order\ of\ } x) \times 10 + (\mathit{order\ of\ } y)) \bmod 7$ $\longrightarrow$

H_2

bucket address	0	1	2	3	4	5	6
bucket count	2	2	4	2	2	4	4
bucket contents	{I1, I4} {I3, I5}	{I1, I5} {I1, I5}	{I2, I3} {I2, I3} {I2, I3} {I2, I3}	{I2, I4} {I2, I4}	{I2, I5} {I2, I5}	{I1, I2} {I1, I2} {I1, I2} {I1, I2}	{I1, I3} {I1, I3} {I1, I3} {I1, I3}

Figure 6.5: Hash table, H_2, for candidate 2-itemsets: This hash table was generated by scanning the transactions of Table 6.1 while determining L_1. If the minimum support count is, say, 3, then the itemsets in buckets 0, 1, 3, and 4 cannot be frequent and so they should not be included in C_2.

<3> Exploring Vertical Data Format: ECLAT

- ECLAT (Equivalence Class Transformation): A depth-first search algorithm using set intersection [Zaki et al. @KDD'97]
- Tid-List: List of transaction-ids containing an itemset
- Vertical format: $t(e) = \{T_{10}, T_{20}, T_{30}\}$; $t(a) = \{T_{10}, T_{20}\}$; $t(ae) = \{T_{10}, T_{20}\}$
- Properties of Tid-Lists
  - $t(X) = t(Y)$: X and Y always happen together (e.g., $t(ac\} = t(d\})$)
  - $t(X) \subset t(Y)$: transaction having X always has Y (e.g., $t(ac) \subset t(ce)$)
- Deriving frequent patterns based on vertical intersections
- Using diffset to accelerate mining
  - Only keep track of differences of tids
  - $t(e) = \{T_{10}, T_{20}, T_{30}\}$, $t(ce) = \{T_{10}, T_{30}\} \rightarrow$ Diffset $(ce, e) = \{T_{20}\}$

A transaction DB in Horizontal Data Format

Tid	Itemset
10	a, c, d, e
20	a, b, e
30	b, c, e

The transaction DB in Vertical Data Format

Item	TidList
a	10, 20
b	20, 30
c	10, 30
d	10
e	10, 20, 30

<4> Mining Frequent Patterns by Pattern Growth

- Apriori: A *breadth-first search* mining algorithm
  - First find the complete set of frequent k-itemsets
  - Then derive frequent (k+1)-itemset candidates
  - Scan DB again to find true frequent (k+1)-itemsets

Two nontrivial costs:

- *It may still need to generate a huge number of candidate sets.* For example, if there are 10^4 frequent 1-itemsets, the Apriori algorithm will need to generate more than 10^7 candidate 2-itemsets.

- *It may need to repeatedly scan the whole database and check a large set of candidates by pattern matching.* It is costly to go over each transaction in the database to determine the support of the candidate itemsets.

<4> Mining Frequent Patterns by Pattern Growth

- Apriori: A *breadth-first search* mining algorithm
    - First find the complete set of frequent k-itemsets
    - Then derive frequent (k+1)-itemset candidates
    - Scan DB again to find true frequent (k+1)-itemsets
- Motivation for a different mining methodology
    - Can we mine the complete set of frequent patterns without such a costly generation process?
    - For a frequent itemset ρ, can subsequent search be confined to only those transactions that contain ρ?
        - A *depth-first search* mining algorithm?
- Such thinking leads to a frequent pattern (FP) growth approach:
    - FPGrowth (J. Han, J. Pei, Y. Yin, "Mining Frequent Patterns without Candidate Generation," SIGMOD 2000)

<4> High-level Idea of FP-growth Method

- Essence of frequent pattern growth (FPGrowth) methodology
  - Find frequent single items and partition the database based on each such single item pattern
  - Recursively grow frequent patterns by doing the above for each *partitioned database* (also called the pattern's *conditional database*)
  - To facilitate efficient processing, an efficient data structure, FP-tree, can be constructed
- Mining becomes
  - Recursively construct and mine (conditional) FP-trees
  - Until the resulting FP-tree is empty, or until it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

Example: Construct FP-tree from a Transaction DB

TID	Items in the Transaction	Ordered, frequent itemlist
100	*{f, a, c, d, g, i, m, p}*	
200	*{a, b, c, f, l, m, o}*	
300	*{b, f, h, j, o, w}*	
400	*{b, c, k, s, p}*	
500	*{a, f, c, e, l, p, m, n}*	

1. Scan DB once, find single item frequent pattern:

   Let min_support = 3

   f:4, a:3, c:4, b:3, m:3, p:3

2. Sort frequent items in frequency descending order, f-list

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

Example: Construct FP-tree from a Transaction DB

TID	Items in the Transaction	Ordered, frequent itemlist
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*

1. Scan DB once, find single item frequent pattern:

   Let min_support = 3

   f:4, a:3, c:4, b:3, m:3, p:3

2. Sort frequent items in frequency descending order, f-list

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

Example: Construct FP-tree from a Transaction DB

TID	Items in the Transaction	Ordered, frequent itemlist
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*

After inserting the 1st frequent Itemlist: "*f, c, a, m, p*"

1. Scan DB once, find single item frequent pattern:

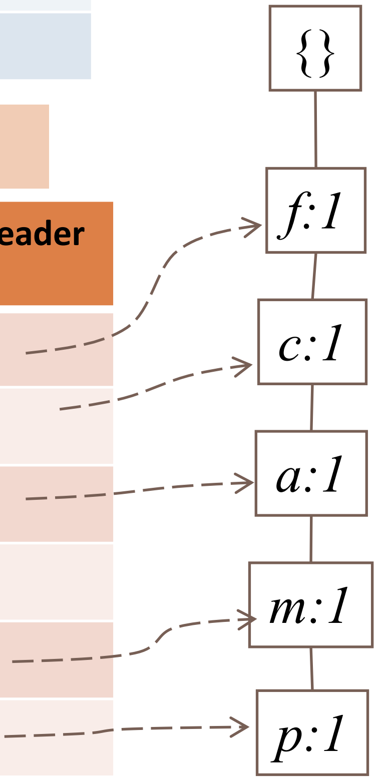
   Let min_support = 3

   f:4, a:3, c:4, b:3, m:3, p:3

2. Sort frequent items in frequency descending order, f-list F-list = f-c-a-b-m-p
3. Scan DB again, construct FP-tree
   - The frequent itemlist of each transaction is inserted as a branch, with shared sub-branches merged, counts accumulated

Header Table

Item	Frequency	header
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



Example: Construct FP-tree from a Transaction DB

TID	Items in the Transaction	Ordered, frequent itemlist
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*

After inserting the 2nd frequent itemlist "*f, c, a, b, m*"

1. Scan DB once, find single item frequent pattern:

   Let min_support = 3

   f:4, a:3, c:4, b:3, m:3, p:3

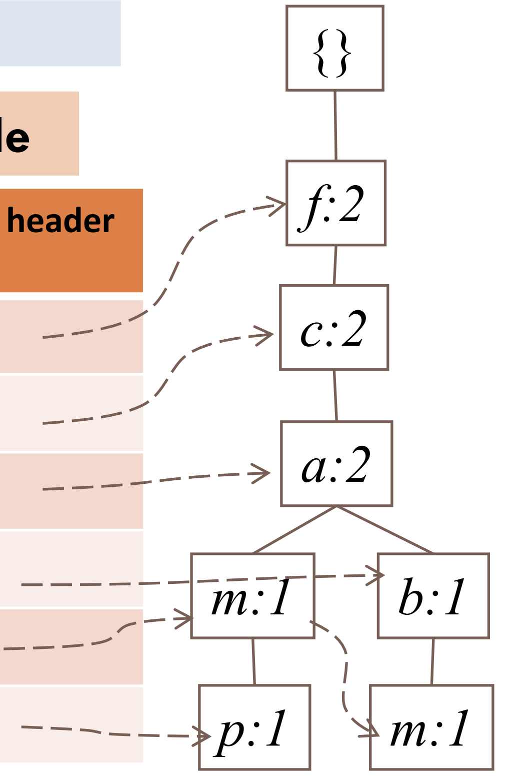
2. Sort frequent items in frequency descending order, f-list

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

3. Scan DB again, construct FP-tree
   - The frequent itemlist of each transaction is inserted as a branch, with shared sub-branches merged, counts accumulated

Header Table

Item	Frequency	header
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



Example: Construct FP-tree from a Transaction DB

TID	Items in the Transaction	Ordered, frequent itemlist
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*

1. Scan DB once, find single item frequent pattern:

   Let min_support = 3

   f:4, a:3, c:4, b:3, m:3, p:3

2. Sort frequent items in frequency descending order, f-list

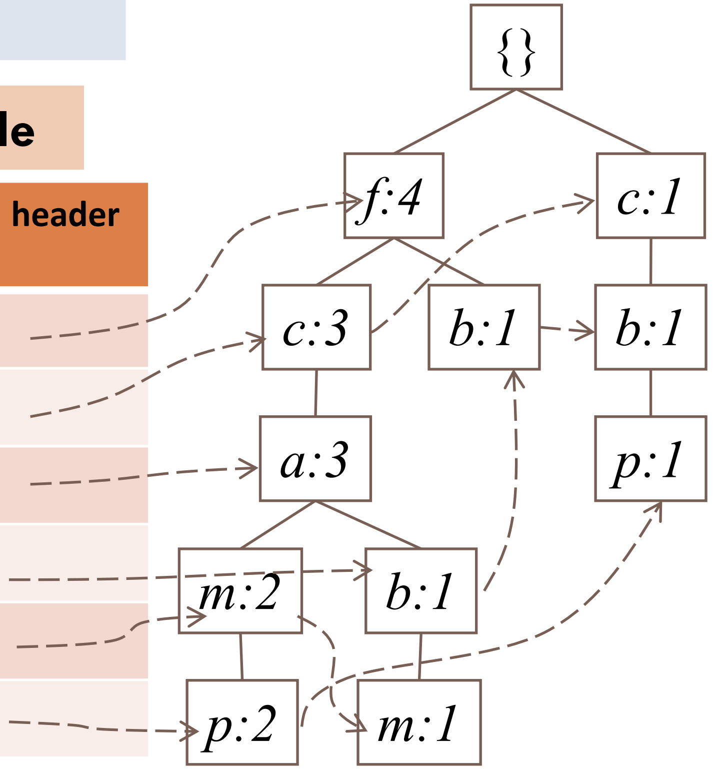
   F-list = f-c-a-b-m-p

3. Scan DB again, construct FP-tree
   - The frequent itemlist of each transaction is inserted as a branch, with shared sub-branches merged, counts accumulated

Header Table

Item	Frequency	header
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

After inserting all the frequent itemlists

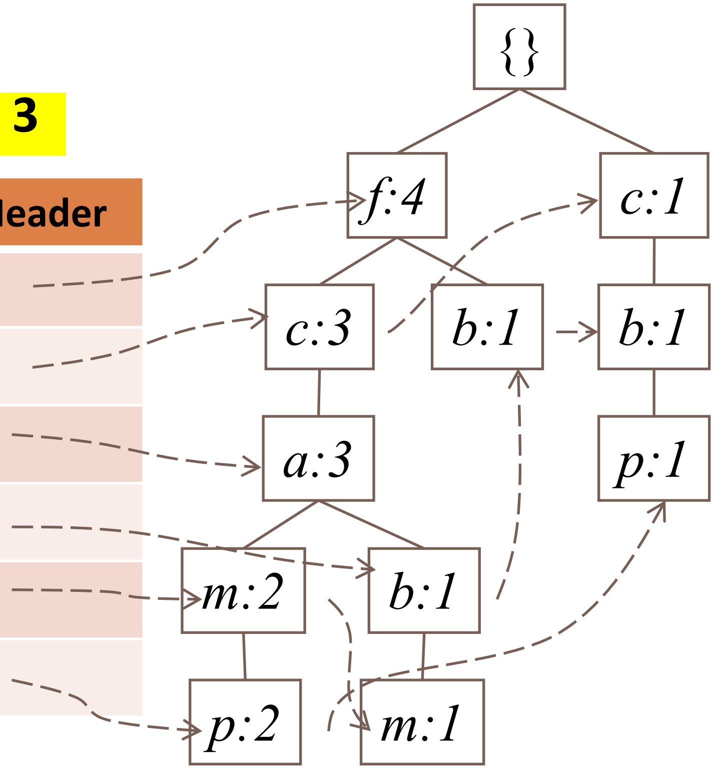


Mining FP-Tree: Divide and Conquer Based on Patterns and Data

- Pattern mining can be partitioned according to current patterns
  - Patterns containing *p*: *p*'s conditional database: *fcam:2, cb:1*
    - *p*'s conditional database (i.e., the database under the condition that *p* exists):
      - *transformed prefix paths* of item *p*
  - Patterns having m but no p: m's conditional database: *fca:2, fcab:1*
  - …… ……

min_support = 3

Item	Frequency	Header
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



Conditional database of each pattern

Item	*Conditional database*
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

Mine Each Conditional Database Recursively

min_support = 3

Conditional Data Bases

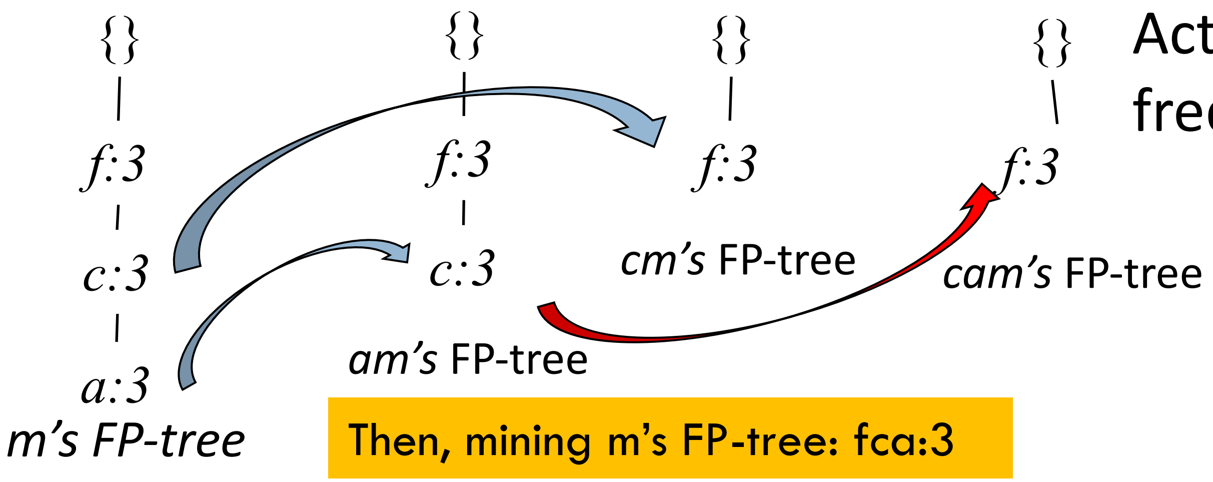
item	***cond. data base***
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

- For each conditional database
  - Mine single-item patterns
  - Construct its FP-tree & mine it

p's conditional DB: ***fcam:2, cb:1 → c: 3***

m's conditional DB: ***fca:2, fcab:1 → fca: 3***

b's conditional DB: ***fca:1, f:1, c:1 → ϕ***



Then, mining m's FP-tree: fca:3

Actually, for single branch FP-tree, all the frequent patterns can be generated in one shot

m: 3

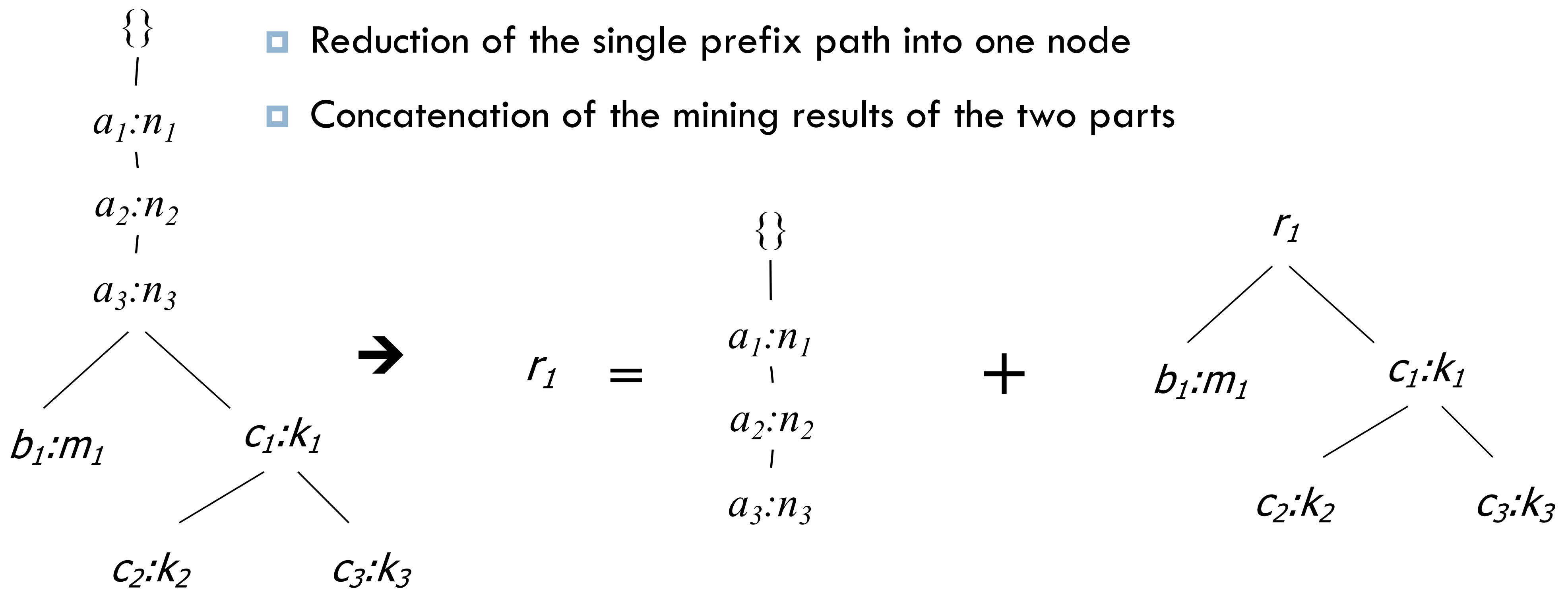
fm: 3, cm: 3, am: 3

fcm: 3, fam:3, cam: 3

fcam: 3

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

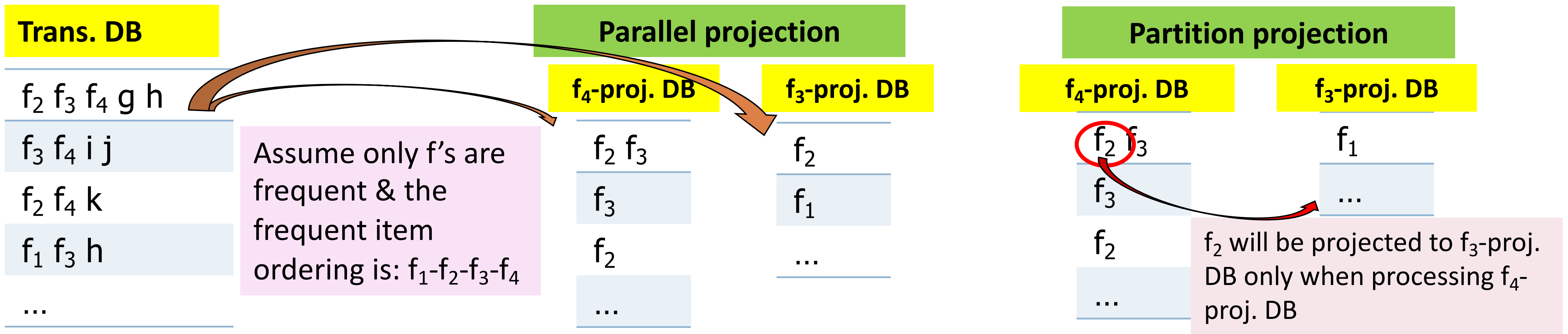


FPGrowth: Mining Frequent Patterns by Pattern Growth

- Essence of frequent pattern growth (FPGrowth) methodology
  - Find frequent single items and partition the database based on each such single item pattern
  - Recursively grow frequent patterns by doing the above for each *partitioned database* (also called the pattern's *conditional database*)
  - To facilitate efficient processing, an efficient data structure, FP-tree, can be constructed
- Mining becomes
  - Recursively construct and mine (conditional) FP-trees
  - Until the resulting FP-tree is empty, or until it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

Scaling FP-growth by Item-Based Data Projection

- What if FP-tree cannot fit in memory?—Do not construct FP-tree
  - “Project” the database based on frequent single items
  - Construct & mine FP-tree for each projected DB
- Parallel projection vs. partition projection
  - Parallel projection: Project the DB on each frequent item
    - Space costly, all partitions can be processed in parallel
  - Partition projection: Partition the DB in order
    - Passing the unprocessed parts to subsequent partitions



Assume only f’s are frequent & the frequent item ordering is: f_1-f_2-f_3-f_4

f_2 will be projected to f_3-proj. DB only when processing f_4-proj. DB

Chapter 6: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Efficient Pattern Mining Methods
- Pattern Evaluation
- Summary

Pattern Evaluation

- Limitation of the Support-Confidence Framework
- Interestingness Measures: Lift and χ^2
- Null-Invariant Measures
- Comparison of Interestingness Measures

- Pattern mining will generate a large set of patterns/rules
  - Not all the generated patterns/rules are interesting

How to Judge if a Rule/Pattern Is Interesting?

- Pattern-mining will generate a large set of patterns/rules
  - Not all the generated patterns/rules are interesting
- Interestingness measures: Objective vs. subjective

How to Judge if a Rule/Pattern Is Interesting?

- Pattern-mining will generate a large set of patterns/rules
  - Not all the generated patterns/rules are interesting
- Interestingness measures: Objective vs. subjective
  - Objective interestingness measures
    - Support, confidence, correlation, …
  - Subjective interestingness measures:
    - Different users may judge interestingness differently
    - Let a user specify
      - Query-based: Relevant to a user's particular request
    - Judge against one's knowledge base
      - unexpected, freshness, timeliness

Limitation of the Support-Confidence Framework

- Are *s* and *c* interesting in association rules: “A $\Rightarrow$ B” [*s*, *c*]?

Limitation of the Support-Confidence Framework

- Are *s* and *c* interesting in association rules: "A $\Rightarrow$ B" [*s*, *c*]?
- Example: Suppose one school may have the following statistics on # of students who may play basketball and/or eat cereal:

	play-basketball	not play-basketball	sum (row)
eat-cereal	400	350	750
not eat-cereal	200	50	250
sum(col.)	600	400	1000

2-way contingency table

Limitation of the Support-Confidence Framework

- Are *s* and *c* interesting in association rules: "A ⇒ B" [*s*, *c*]?
- Example: Suppose one school may have the following statistics on # of students who may play basketball and/or eat cereal:

	play-basketball	not play-basketball	sum (row)
eat-cereal	400	350	750
not eat-cereal	200	50	250
sum(col.)	600	400	1000

2-way contingency table

- Association rule mining may generate the following:
  - *play-basketball* ⇒ *eat-cereal* [40%, 66.7%] (higher s & c)
- But this strong association rule is misleading: The overall % of students eating cereal is 75% > 66.7%, a more telling rule:
  - ¬ *play-basketball* ⇒ *eat-cereal* [35%, 87.5%] (high s & c)

Interestingness Measure: Lift

- Measure of dependent/correlated events: **lift**

$$lift(B,C) = \frac{c(B \rightarrow C)}{s(C)} = \frac{s(B \cup C)}{s(B) \times s(C)}$$

Lift is more telling than s & c

	B	¬B	Σ_{row}
C	400	350	750
¬C	200	50	250
$\Sigma_{col.}$	600	400	1000

Interestingness Measure: Lift

- Measure of dependent/correlated events: **lift**

$$lift(B,C) = \frac{c(B \rightarrow C)}{s(C)} = \frac{P(B \cup C)}{P(B) \times P(C)}$$

- Lift(B, C) may tell how B and C are correlated
  - Lift(B, C) = 1: B and C are independent
  - > 1: positively correlated
  - < 1: negatively correlated

Lift is more telling than s & c

	B	¬B	Σ_{row}
C	400	350	750
¬C	200	50	250
$\Sigma_{col.}$	600	400	1000

Interestingness Measure: Lift

- Measure of dependent/correlated events: **lift**

$$lift(B,C) = \frac{c(B \rightarrow C)}{s(C)} = \frac{s(B \cup C)}{s(B) \times s(C)}$$

Lift is more telling than s & c

	B	¬B	Σ_{row}
C	400	350	750
¬C	200	50	250
$\Sigma_{col.}$	600	400	1000

- Lift(B, C) may tell how B and C are correlated
  - Lift(B, C) = 1: B and C are independent
  - > 1: positively correlated
  - < 1: negatively correlated
- In our example,

$$lift(B,C) = \frac{400/1000}{600/1000 \times 750/1000} = 0.89$$

$$lift(B,\neg C) = \frac{200/1000}{600/1000 \times 250/1000} = 1.33$$

- Thus, B and C are negatively correlated since lift(B, C) < 1;
  - B and ¬C are positively correlated since lift(B, ¬C) > 1

Interestingness Measure: χ^2

- Another measure to test correlated events: χ^2

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

	B	¬B	Σ_{row}
C	400 (450)	350 (300)	750
¬C	200 (150)	50 (100)	250
Σ_{col}	600	400	1000

Expected value

Observed value

Interestingness Measure: χ^2

- Another measure to test correlated events: **χ²**

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

	B	¬B	Σ_{row}
C	400 (450)	350 (300)	750
¬C	200 (150)	50 (100)	250
Σ_{col}	600	400	1000

Expected value

Observed value

- For the table on the right,

$$\chi^2 = \frac{(400-450)^2}{450} + \frac{(350-300)^2}{300} + \frac{(200-150)^2}{150} + \frac{(50-100)^2}{100} = 55.56$$

- By consulting a table of critical values of the **χ²** distribution, one can conclude that the chance for B and C to be independent is very low (< 0.01)
- **χ²**-test shows B and C are negatively correlated since the expected value is 450 but the observed is only 400
- Thus, **χ²** is also more telling than the support-confidence framework

Lift and χ^2 : Are They Always Good Measures?

- Null transactions: Transactions that contain neither B nor C
- Let's examine the new dataset D
  - BC (100) is much rarer than B¬C (1000) and ¬BC (1000), but there are many ¬B¬C (100000)
  - Unlikely B & C will happen together!

	B	¬B	Σ_{row}
C	100	1000	1100
¬C	1000	100000	101000
$\Sigma_{col.}$	1100	101000	102100

null transactions

Contingency table with expected values added

	B	¬B	Σ_{row}
C	100 (11.85)	1000	1100
¬C	1000 (988.15)	100000	101000
$\Sigma_{col.}$	1100	101000	102100

- But, Lift(B, C) = 8.44 >> 1 (Lift shows B and C are strongly positively correlated!)
- χ^2 = 670: Observed(BC) >> expected value (11.85)
- *Too many null transactions may "spoil the soup"!*

Interestingness Measures & Null-Invariance

- *Null invariance:* Value does not change with the # of null-transactions
- A few interestingness measures: Some are null invariant

Measure	Definition	Range	Null-Invariant?
$\chi^2(A,B)$	$\sum_{i,j} \frac{(e(a_i,b_j)-o(a_i,b_j))^2}{e(a_i,b_j)}$	$[0, \infty]$	No
$Lift(A,B)$	$\frac{s(A\cup B)}{s(A)\times s(B)}$	$[0, \infty]$	No
$Allconf(A,B)$	$\frac{s(A\cup B)}{max\{s(A),s(B)\}}$	$[0, 1]$	Yes
$Jaccard(A,B)$	$\frac{s(A\cup B)}{s(A)+s(B)-s(A\cup B)}$	$[0, 1]$	Yes
$Cosine(A,B)$	$\frac{s(A\cup B)}{\sqrt{s(A)\times s(B)}}$	$[0, 1]$	Yes
$Kulczynski(A,B)$	$\frac{1}{2}(\frac{s(A\cup B)}{s(A)}+\frac{s(A\cup B)}{s(B)})$	$[0, 1]$	Yes
$MaxConf(A,B)$	$max\{\frac{s(A\cup B)}{s(A)},\frac{s(A\cup B)}{s(B)}\}$	$[0, 1]$	Yes

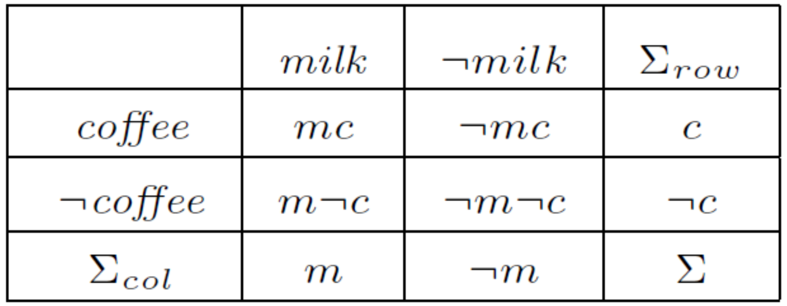
***X^2** and lift are not null-invariant*

Jaccard, consine, AllConf, MaxConf, and *Kulczynski* are *null-invariant measures*

Null Invariance: An Important Property

- Why is null invariance crucial for the analysis of massive transaction data?
  - Many transactions may contain neither milk nor coffee!

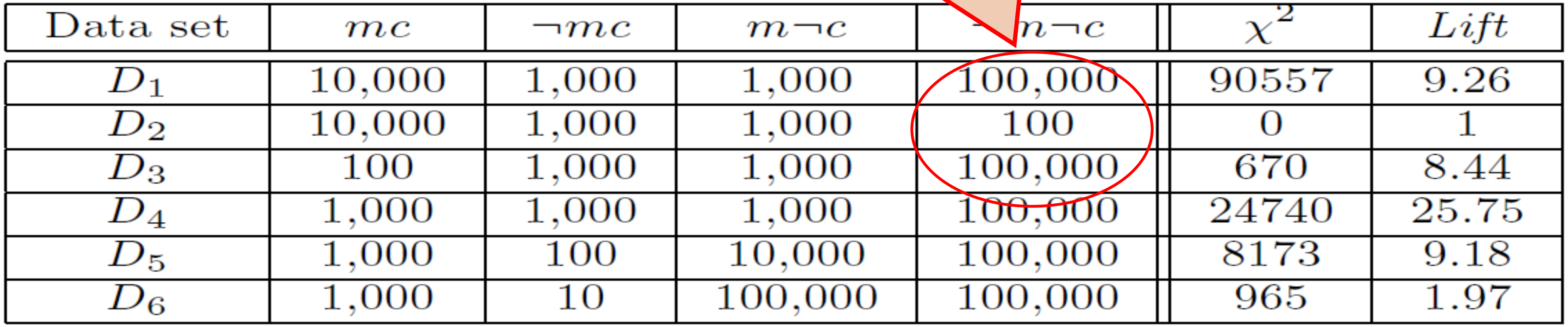
milk vs. coffee contingency table



	$milk$	$\neg milk$	Σ_{row}
$coffee$	mc	$\neg mc$	c
$\neg coffee$	$m\neg c$	$\neg m\neg c$	$\neg c$
Σ_{col}	m	$\neg m$	Σ

- Lift and χ^2 are not null-invariant: not good to evaluate data that contain too many or too few null transactions!
- Many measures are not null-invariant!

Null-transactions w.r.t. m and c



Data set	mc	$\neg mc$	$m\neg c$	$\neg m\neg c$	χ^2	$Lift$
D_1	10,000	1,000	1,000	100,000	90557	9.26
D_2	10,000	1,000	1,000	100	0	1
D_3	100	1,000	1,000	100,000	670	8.44
D_4	1,000	1,000	1,000	100,000	24740	25.75
D_5	1,000	100	10,000	100,000	8173	9.18
D_6	1,000	10	100,000	100,000	965	1.97

Comparison of Null-Invariant Measures

- Not all null-invariant measures are created equal
- Which one is better?
  - D_4—D_6 differentiate the null-invariant measures
  - Kulc (Kulczynski 1927) holds firm and is in balance of both directional implications

2-variable contingency table

	$milk$	$\neg milk$	Σ_{row}
$coffee$	mc	$\neg mc$	c
$\neg coffee$	$m\neg c$	$\neg m\neg c$	$\neg c$
Σ_{col}	m	$\neg m$	Σ

All 5 are null-invariant

Data set	mc	$\neg mc$	$m\neg c$	$\neg m\neg c$	$AllConf$	Jaccard	$Cosine$	$Kulc$	$MaxConf$
D_1	10,000	1,000	1,000	100,000	0.91	0.83	0.91	0.91	0.91
D_2	10,000	1,000	1,000	100	0.91	0.83	0.91	0.91	0.91
D_3	100	1,000	1,000	100,000	0.09	0.05	0.09	0.09	0.09
D_4	1,000	1,000	1,000	100,000	0.5	0.33	0.5	0.5	0.5
D_5	1,000	100	10,000	100,000	0.09	0.09	0.29	0.5	0.91
D_6	1,000	10	100,000	100,000	0.01	0.01	0.10	0.5	0.99

Subtle: They disagree on those cases

Imbalance Ratio with Kulczynski Measure

- IR (Imbalance Ratio): measure the imbalance of two itemsets A and B in rule implications:

$$IR(A, B) = \frac{|s(A) - s(B)|}{s(A) + s(B) - s(A \cup B)}$$

- Kulczynski and Imbalance Ratio (IR) together present a clear picture for all the three datasets D_4 through D_6
  - D_4 is neutral & balanced; D_5 is neutral but imbalanced
  - D_6 is neutral but very imbalanced

Data set	mc	$\neg mc$	$m\neg c$	$\neg m\neg c$	Jaccard	*Cosine*	*Kulc*	IR
D_1	10,000	1,000	1,000	100,000	0.83	0.91	0.91	0
D_2	10,000	1,000	1,000	100	0.83	0.91	0.91	0
D_3	100	1,000	1,000	100,000	0.05	0.09	0.09	0
D_4	1,000	1,000	1,000	100,000	0.33	0.5	0.5	0
D_5	1,000	100	10,000	100,000	0.09	0.29	0.5	0.89
D_6	1,000	10	100,000	100,000	0.01	0.10	0.5	0.99

What Measures to Choose for Effective Pattern Evaluation?

- Null value cases are predominant in many large datasets
  - Neither milk nor coffee is in most of the baskets; neither Mike nor Jim is an author in most of the papers; ……
- *Null-invariance* is an important property
- Lift, χ^2 and cosine are good measures if null transactions are not predominant
  - Otherwise, *Kulczynski* + *Imbalance Ratio* should be used to judge the interestingness of a pattern

Chapter 6: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Efficient Pattern Mining Methods
- Pattern Evaluation
- Summary