



CS 412 Intro. to Data Mining

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

Jiawei Han, Computer Science, Univ. Illinois at Urbana-Champaign, 2017





Chapter 6: 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?

- 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
- 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?

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
 - ↳ pattern → unam.
 - 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

ຂະໜາດຕັ້ງທີ່

- **Itemset:** A set of one or more items
ເພື່ອກົດໃຫຍ່ໄລຍ່ → ປິບຕະນະທີ່ອັນດີ Ex. $k=2 \Rightarrow$ ໂກງໍາ ສົກກໍາ 2 ອຳນວຍຕົວມີຄວາມ
- **k-itemset:** $X = \{x_1, \dots, x_k\}$
- Ex. {Beer, Nuts, Diaper} is a 3-itemset
- **(absolute) support (count)** of X , $\text{sup}\{X\}$:
Frequency or the number of occurrences of an itemset X
| ຂົດປະກາດສູນ
- Ex. $\text{sup}\{\text{Beer}\} = 3$
- Ex. $\text{sup}\{\text{Diaper}\} = 4$
- Ex. $\text{sup}\{\text{Beer}, \text{Diaper}\} = 3$
- Ex. $\text{sup}\{\text{Beer}, \text{Eggs}\} = 1$



Absolute support

| Tid | Items bought |
|-----|----------------------------------|
| 10 | Beer, Nuts, Diaper 1 |
| 20 | Beer, Coffee, Diaper 2 |
| 30 | Beer, Diaper, Eggs 3 1 |
| 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. $s\{\text{Beer}\} = 3/5 = 60\%$
- Ex. $s\{\text{Diaper}\} = 4/5 = 80\%$
- Ex. $s\{\text{Beer}, \text{Eggs}\} = 1/5 = 20\%$

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
- Let $\sigma = 50\%$ (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 |

- Why 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*
 - How strong is this rule? (support, confidence)
 - Measuring association rules: $X \rightarrow Y$ (s, c)
 - Both X and Y are itemsets
 - **Support, s:** The probability that a transaction contains $X \cup Y$
 - Ex. $s\{\text{Diaper, Beer}\} = 3/5 = 0.6$ (i.e., 60%)
 - **Confidence, c:** The *conditional probability* that a transaction containing X also contains Y
 - Calculation: $c = \text{sup}(X \cup Y) / \text{sup}(X)$
 - Ex. $c = \text{sup}\{\text{Diaper, Beer}\}/\text{sup}\{\text{Diaper}\} = \frac{3}{4} = 0.75$

| 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 |

A Venn diagram illustrating the union of two itemsets, $\{Beer\}$ and $\{Diaper\}$. It consists of two overlapping circles: a yellow circle labeled "Containing beer" on its left and a blue circle labeled "Containing diaper" on its right. The intersection of the two circles is shaded green and labeled $\{Beer\} \cup \{Diaper\}$. A green line connects the text "Containing both" above the intersection to the green-shaded area. An orange arrow points from the text "Both X and Y are itemsets" in the list above to the intersection area of the Venn diagram.

Containing both

Containing beer

Containing diaper

$\{Beer\} \cup \{Diaper\} = \{Beer, Diaper\}$

Note: $X \cup Y$: the union of two itemsets
■ The set contains both X and Y

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 $c \geq minconf$
- 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?)

| 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

Challenge: There Are Too Many Frequent Patterns!

- A long pattern contains a combinatorial number of sub-patterns
- How many frequent itemsets does the following TDB₁ contain?

- TDB₁: T₁: {a₁, ..., a₅₀}; T₂: {a₁, ..., a₁₀₀}

- Assuming (absolute) *minsup* = 1

- Let's have a try

1-itemsets: {a₁} : 2, {a₂} : 2, ..., {a₅₀} : 2, {a₅₁} : 1, ..., {a₁₀₀} : 1,

2-itemsets: {a₁, a₂} : 2, ..., {a₁, a₅₀} : 2, {a₁, a₅₁} : 1, ..., ..., {a₉₉, a₁₀₀} : 1,

..., ..., ..., ...

99-itemsets: {a₁, a₂, ..., a₉₉} : 1, ..., {a₂, a₃, ..., a₁₀₀} : 1

100-itemset: {a₁, a₂, ..., a₁₀₀} : 1

- The total number of frequent itemsets:

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

A too huge 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* $Y \supset X$, *with the same support* as X
 - Let Transaction DB TDB_1 : $T_1: \{a_1, \dots, a_{50}\}$; $T_2: \{a_1, \dots, a_{100}\}$
 - Suppose $minsup = 1$. How many closed patterns does TDB_1 contain?
 - Two: $P_1: \{\{a_1, \dots, a_{50}\}: 2\}$; $P_2: \{\{a_1, \dots, a_{100}\}: 1\}$
 - **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: $\{\{a_2, \dots, a_{40}\}: 2\}$, $\{\{a_5, a_{51}\}: 1\}$

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 $Y \supset X$
- Difference from close-patterns?
 - Do not care the real support of the sub-patterns of a max-pattern
 - Let Transaction DB TDB_1 : $T_1: \{a_1, \dots, a_{50}\}$; $T_2: \{a_1, \dots, a_{100}\}$
 - Suppose $minsup = 1$. How many max-patterns does TDB_1 contain?
 - One: $P: \{a_1, \dots, a_{100}\}: 1$
 - **Max-pattern is a lossy compression!**
 - We only know $\{a_1, \dots, a_{40}\}$ is frequent
 - But we do not know the real support of $\{a_1, \dots, a_{40}\}, \dots$, any more!
 - Thus in many applications, mining close-patterns is more desirable than mining max-patterns

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

- Basic Concepts
- Efficient Pattern Mining Methods
- 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
- FP-Growth: A Frequent Pattern-Growth Approach
- Mining Closed Patterns

The Downward Closure Property of Frequent Patterns

- Observation: From TDB₁: T₁: {a₁, ..., a₅₀}; T₂: {a₁, ..., a₁₀₀}
 - We get a frequent itemset: {a₁, ..., a₅₀}
 - Also, its subsets are all frequent: {a₁}, {a₂}, ..., {a₅₀}, {a₁, a₂}, ..., {a₁, ..., a₄₉}, ...
 - 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 \Rightarrow Item set $\{ \text{beer, diaper, nuts} \}$ \Rightarrow What's the item set {beer, diaper}
- 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!?  A sharp knife for pruning!

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, Ogihsara, 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 ($F_k \neq \emptyset$) **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

gen'g 1st Item set
of the min sup

The Apriori Algorithm—An Example

Database TDB

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

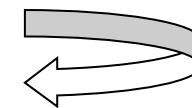
minsup = 2

C_1
1st scan

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

F_1

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



F_2

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

C_2

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

C_2

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

2nd scan

C_3

| Itemset |
|-----------|
| {B, C, E} |

3rd scan

| Itemset | sup |
|-----------|-----|
| {B, C, E} | 2 |

| Transaction ID | Items Purchased |
|----------------|--------------------------|
| 1 | Milk, Bread, Butter |
| 2 | Milk, Eggs |
| 3 | Bread, Butter, Jam |
| 4 | Milk, Eggs, Butter |
| 5 | Bread, Butter, Jam, Milk |

3rd - Scan

C3

| Item set | Sup |
|---------------------|-----|
| milk, Bread, Butter | 2 |
| milk, Butter, Eggs | 1 |
| Bread, Butter, Jam | 2 |

1st - Scan ; min sup = 2

C1

| Item set | Sup |
|----------|-----|
| milk | 4 |
| Bread | 3 |
| Butter | 4 |
| Eggs | 2 |
| Jam | 2 |

1st - Scan ; min sup = 2

F1

| Item set | Sup |
|----------|-----|
| milk | 4 |
| Bread | 3 |
| Butter | 4 |
| Eggs | 2 |
| Jam | 2 |

2nd - Scan

C2

| Item set | Sup |
|---------------|-----|
| milk, bread | 2 |
| milk, Butter | 2 |
| milk, Eggs | 2 |
| milk, Jam | 1 |
| Bread, Butter | 3 |
| Bread, Eggs | 0 |
| Bread, Jam | 2 |
| Butter, Eggs | 1 |
| Butter, Jam | 2 |
| Eggs, Jam | 0 |

F2

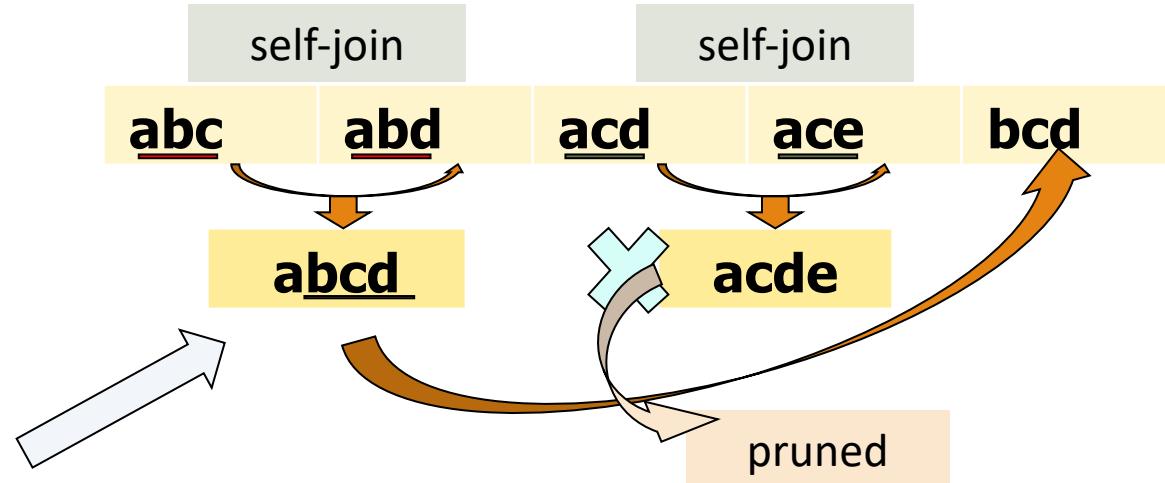
| Item set | Sup |
|---------------|-----|
| milk, Bread | 2 |
| milk, Butter | 2 |
| milk, Eggs | 2 |
| Bread, Butter | 3 |
| Bread, Jam | 2 |
| Butter, Jam | 2 |

F3

| Item set | Sup |
|---------------------|-----|
| milk, Bread, Butter | 2 |
| Bread, Butter, Jam | 2 |

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 $p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}$

from F_{k-1} as p, F_{k-1} as q

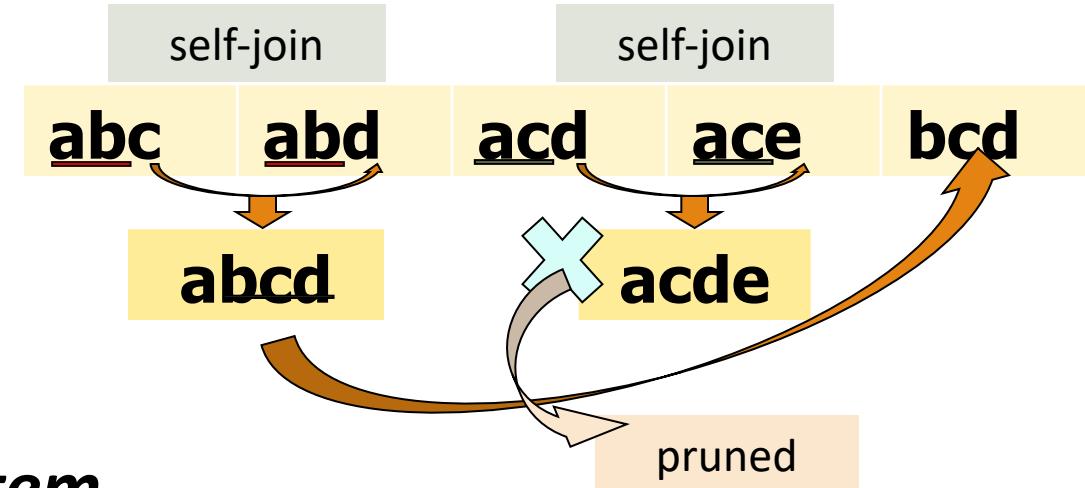
where $p.item_1 = q.item_1, \dots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$

- ❑ 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: Improvements and Alternatives

- ❑ Reduce passes of transaction database scans
 - ❑ Partitioning (e.g., Savasere, et al., 1995)
 - ❑ Dynamic itemset counting (Brin, et al., 1997)
- ❑ 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)
- ❑ Exploring special data structures
 - ❑ Tree projection (Agarwal, et al., 2001)
 - ❑ H-miner (Pei, et al., 2001)
 - ❑ Hypcube decomposition (e.g., LCM: Uno, et al., 2004)

To be discussed in
subsequent slides

To be discussed in
subsequent slides

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*

The diagram illustrates the decomposition of a database TDB into k partitions. It shows a large rectangle labeled TDB at the top, which is partitioned into k smaller rectangles below it. The first partition is labeled TDB_1 and has the condition $\text{sup}_1(X) < \sigma|TDB_1|$ underneath. The second partition is labeled TDB_2 and has the condition $\text{sup}_2(X) < \sigma|TDB_2|$ underneath. Between the partitions are plus signs ($+$). Ellipses (\dots) between the partitions indicate intermediate partitions. Ellipses (\dots) under the conditions indicate more partitions. The final partition is labeled TDB_k and has the condition $\text{sup}_k(X) < \sigma|TDB_k|$ underneath. The entire sum is equated to TDB , with the total support condition $\text{sup}(X) < \sigma|TDB|$ underneath.

$$TDB_1 + TDB_2 + \dots + TDB_k = TDB$$
$$\text{sup}_1(X) < \sigma|TDB_1| \quad \text{sup}_2(X) < \sigma|TDB_2| \quad \dots \quad \text{sup}_k(X) < \sigma|TDB_k| \quad \text{sup}(X) < \sigma|TDB|$$

- 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 (why?)
 - 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

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: $v = \text{hash}(\text{itemset})$
- 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 the transaction to its bucket
 - {ab, ad, ce}
 - {bd, be, de}
 - ...
 - At the end of the first scan,
 - if $\text{minsup} = 80$, remove ab, ad, ce, since $\text{count}\{\text{ab, ad, ce}\} < 80$

| Itemsets | Count |
|--------------|-------|
| {ab, ad, ce} | 35 |
| {bd, be, de} | 298 |
| | ... |
| {yz, qs, wt} | 58 |

Hash Table

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
 - $t(e) = \{T_{10}, T_{20}, T_{30}\}$, $t(ce) = \{T_{10}, T_{30}\} \rightarrow \text{Diffset } (ce, e) = \{T_{20}\}$

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 |

Why 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 develop a *depth-first search* mining algorithm?
 - ❑ For a frequent itemset ρ , can subsequent search be confined to only those transactions that containing ρ ?
- ❑ Such thinking leads to a frequent pattern growth approach:
 - ❑ FP-Growth (J. Han, J. Pei, Y. Yin, “Mining Frequent Patterns without Candidate Generation,” SIGMOD 2000)

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"

- Scan DB once, find single item frequent pattern:

Let min_support = 3

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

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

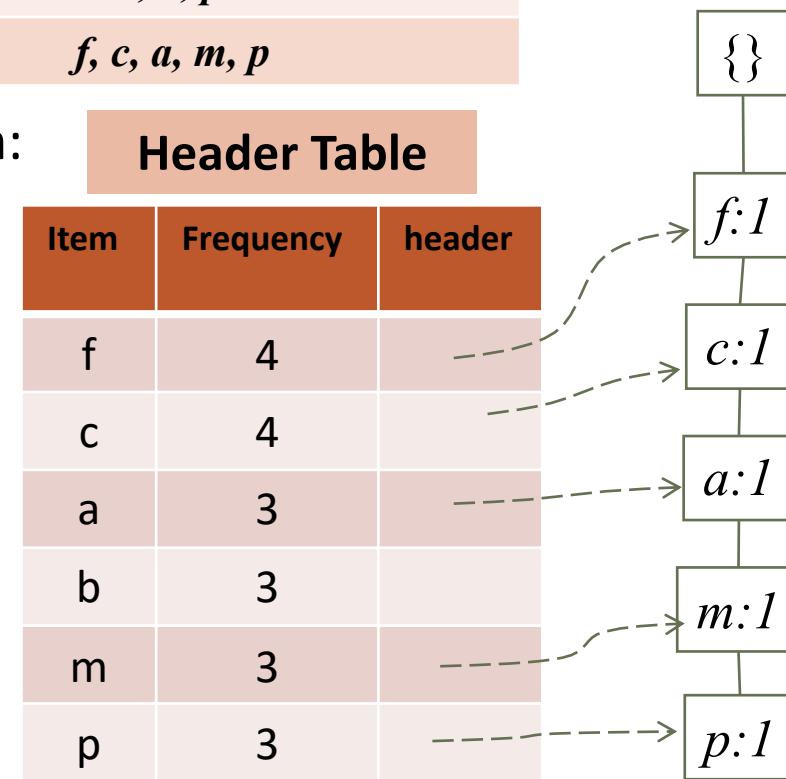
F-list = f-c-a-b-m-p

- 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"

- Scan DB once, find single item frequent pattern:

Let min_support = 3

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

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

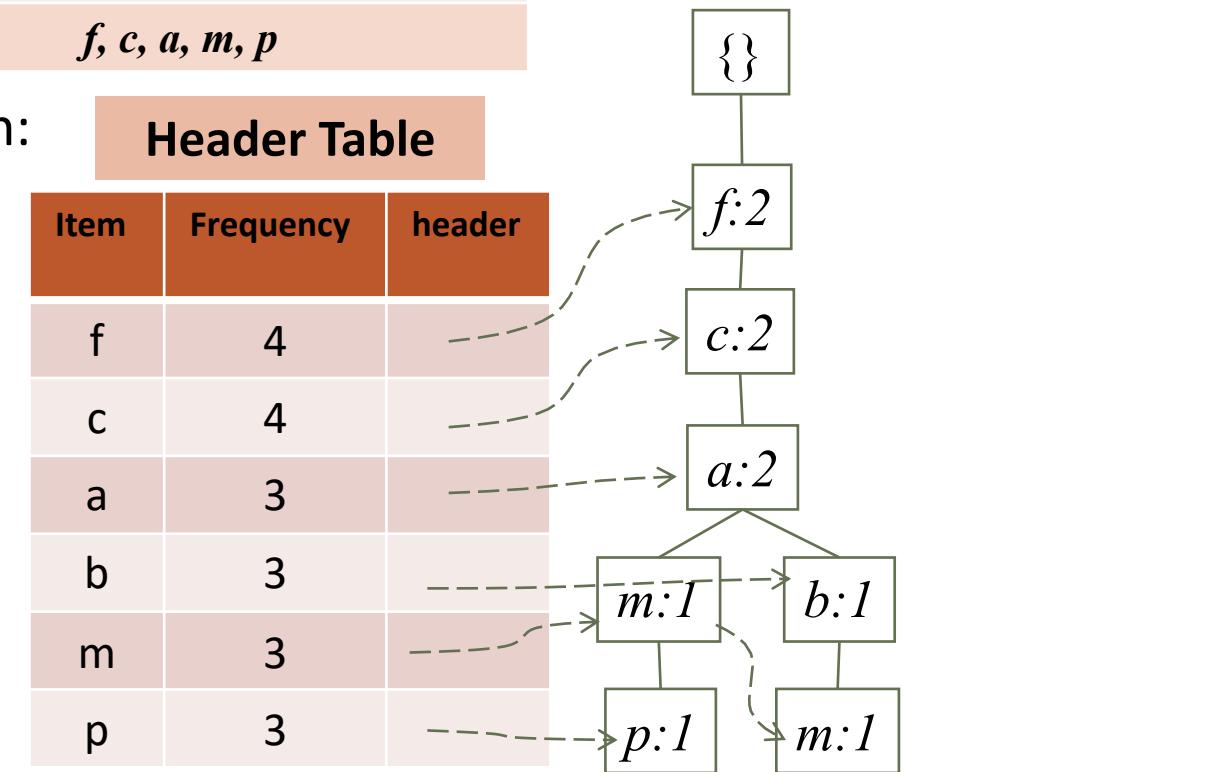
F-list = f-c-a-b-m-p

- 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 all the frequent itemlists

- Scan DB once, find single item frequent pattern:

Let min_support = 3

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

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

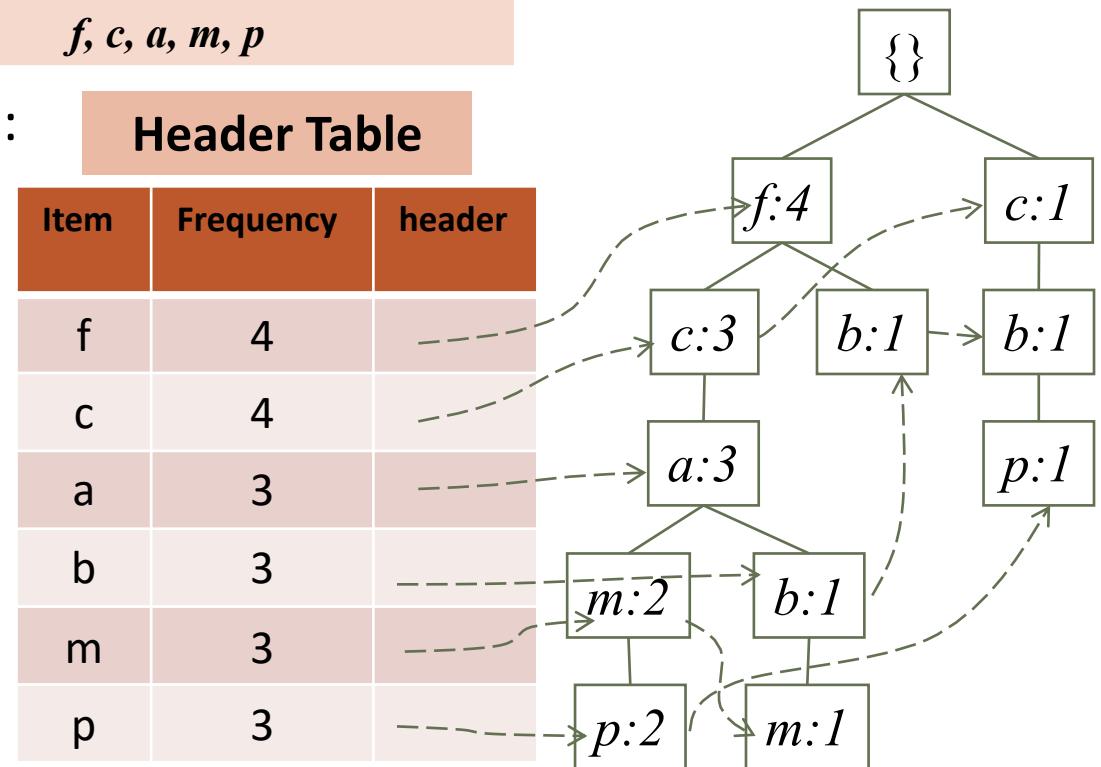
F-list = f-c-a-b-m-p

- Scan DB again, construct FP-tree

- The frequent itemlist of each transaction is inserted as a branch, with shared sub-branches merged, counts accumulated

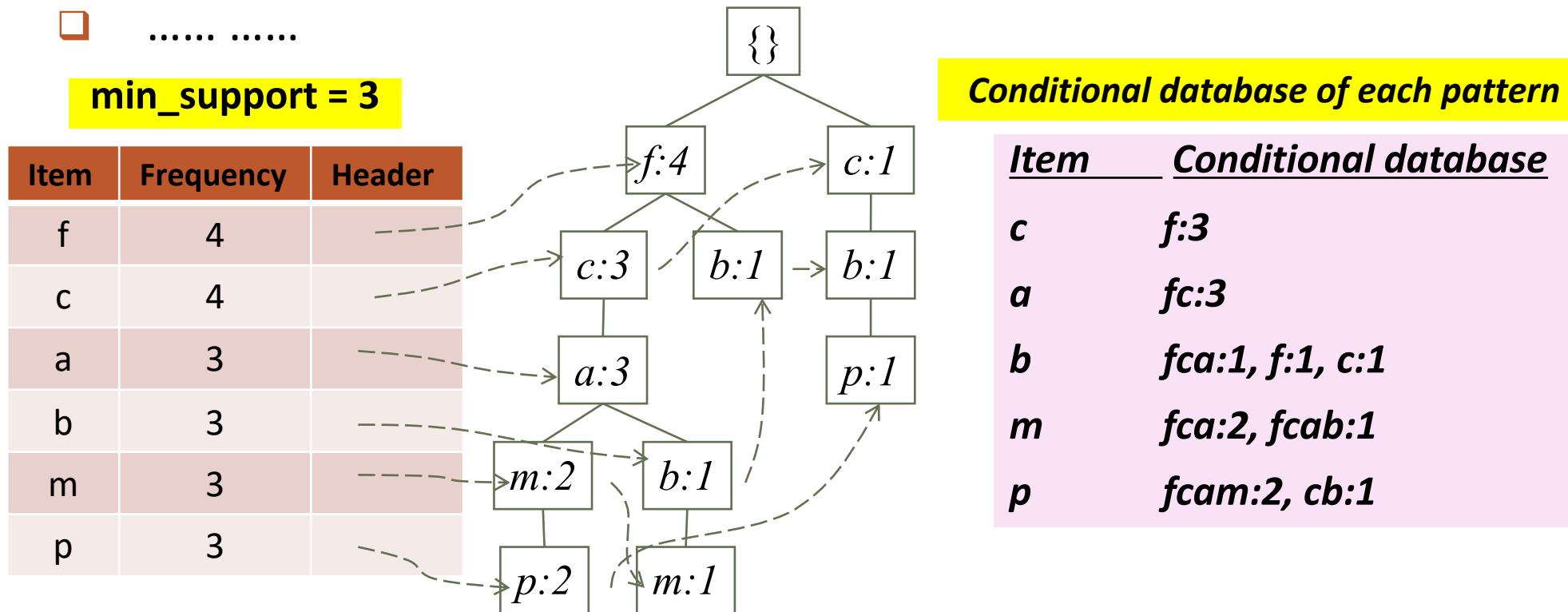
Header Table

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



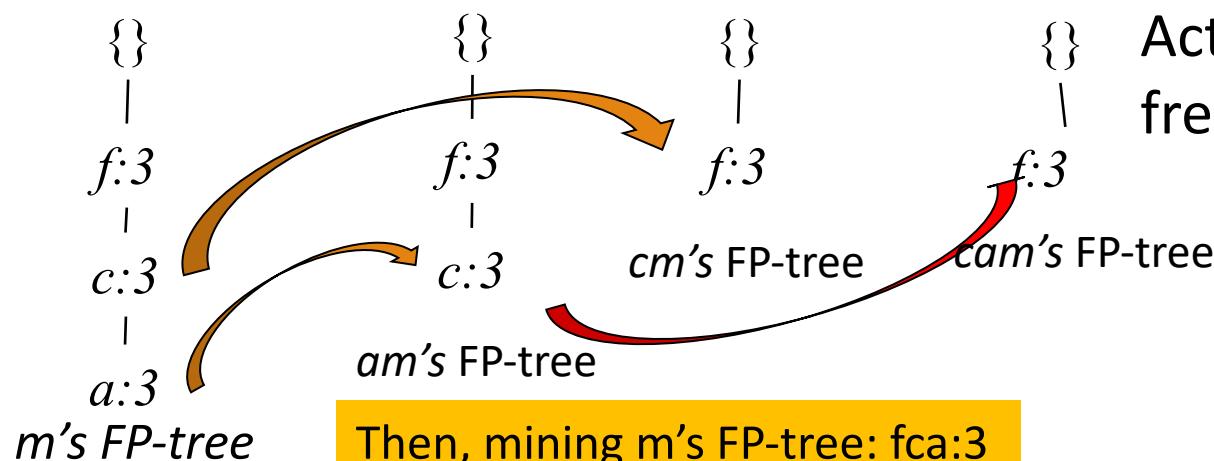
Mining FP-Tree: Divide and Conquer Based on Patterns and Data

- 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: $fca:2, fcab:1$
 -



Mine Each Conditional Database Recursively

| min_support = 3 | |
|------------------------|------------------------|
| Conditional Data Bases | |
| <u>item</u> | <u>cond. data base</u> |
| 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: $fcam:2, cb:1 \rightarrow c: 3$

m's conditional DB: $fca:2, fcab:1 \rightarrow fca: 3$

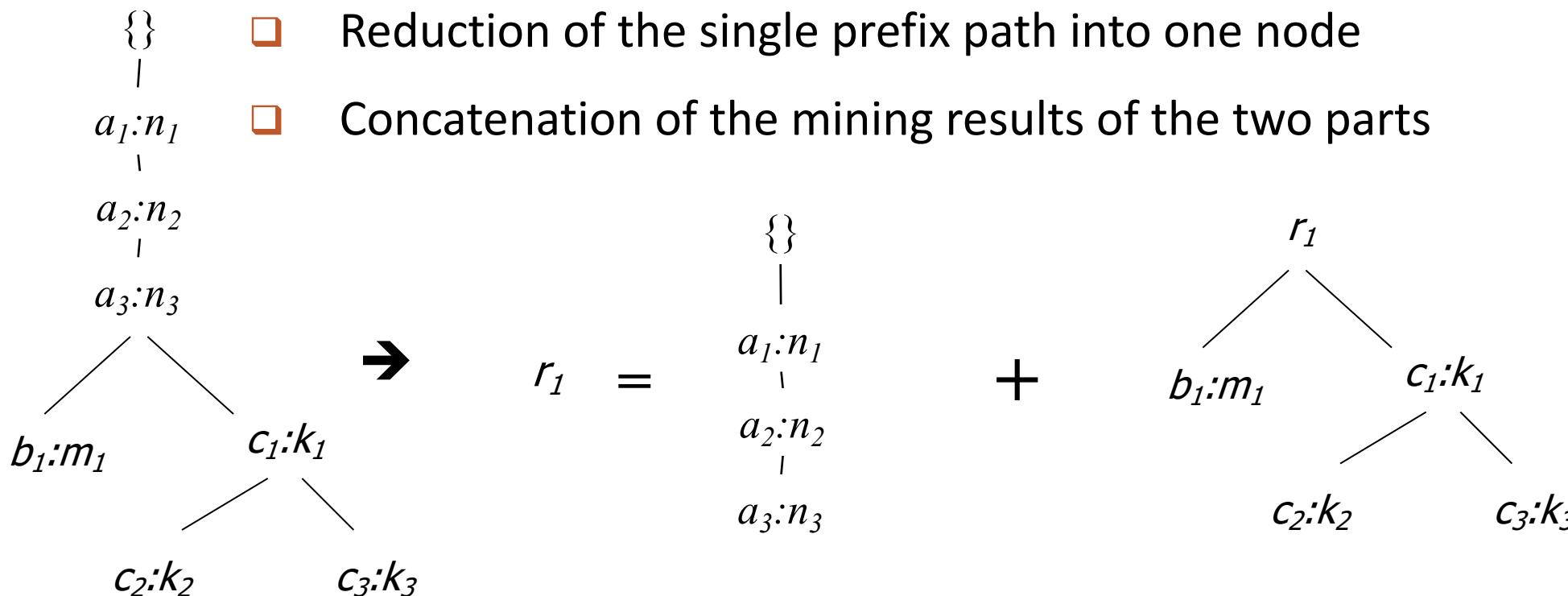
b's conditional DB: $fca:1, f:1, c:1 \rightarrow \emptyset$

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

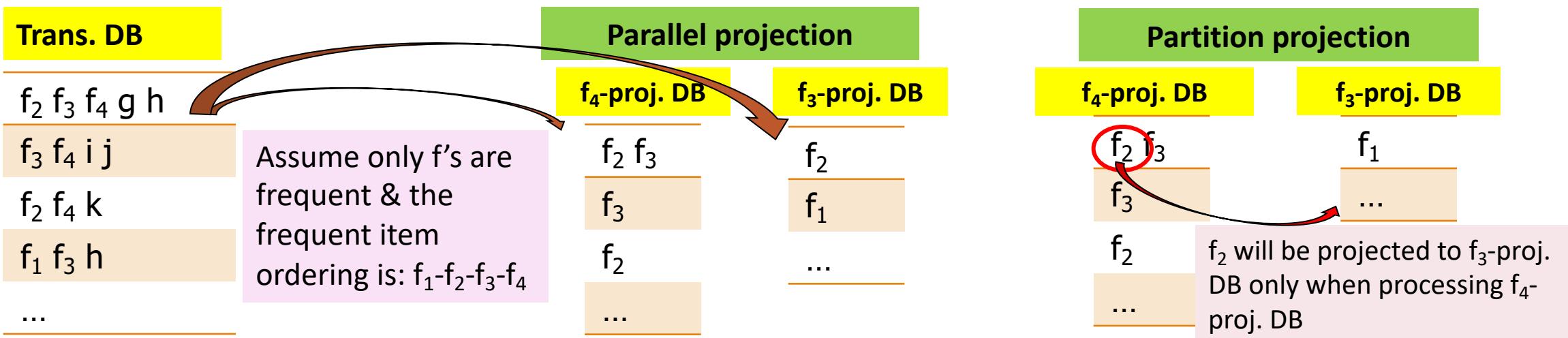


FPGrowth: Mining Frequent Patterns by Pattern Growth

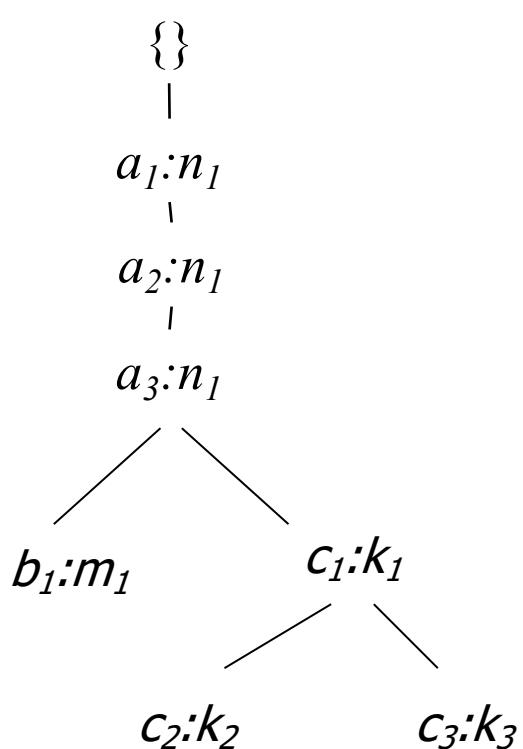
- Essence of frequent pattern growth (FPGrowth) methodology
 - Find frequent single items and partition the database based on each such single item pattern
 - Recursively grow frequent patterns by doing the above for each *partitioned database* (also called the pattern's *conditional database*)
 - To 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



CLOSET+: Mining Closed Itemsets by Pattern-Growth



- ❑ Efficient, *direct* mining of closed itemsets
- ❑ Intuition:
 - ❑ If an FP-tree contains a single branch as shown left
 - ❑ “ a_1, a_2, a_3 ” should be merged
- ❑ Itemset merging: If Y appears in every occurrence of X, then Y is merged with X
 - ❑ d -proj. db: {acef, acf} \rightarrow acfd-proj. db: {e}
 - ❑ Final closed itemset: acfd:2
- ❑ There are many other tricks developed
 - ❑ For details, see J. Wang, et al., “CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets”, KDD'03

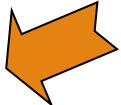
| TID | Items |
|-----|-------|
| 1 | acdef |
| 2 | abe |
| 3 | cefg |
| 4 | acdf |

Let minsupport = 2
a:3, c:3, d:2, e:3, f:3

F-List: a-c-e-f-d

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

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]?
- 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:
 - $play\text{-}basketball \Rightarrow eat\text{-}cereal$ [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:
 - $\neg play\text{-}basketball \Rightarrow eat\text{-}cereal$ [35%, 87.5%] (high s & c)

Interestingness Measure: Lift

If \Rightarrow ສົມບັດຫຼຸດ
ລົງຈະລູນ

ຄ່າການເຊື້ອກ
Correlation

- 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(B, C) may tell how B and C are correlated

Lift(B, C) = 1: B and C are independent \Rightarrow ສົມບັດຫຼຸດ

> 1: positively correlated

< 1: negatively correlated

- For 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 $\neg C$ are positively correlated since lift(B, $\neg C$) > 1

Lift is more telling than s & c

| | B | $\neg B$ | Σ_{row} |
|------------------------|-----|----------|-----------------------|
| C | 400 | 350 | 750 |
| $\neg C$ | 200 | 50 | 250 |
| $\Sigma_{\text{col.}}$ | 600 | 400 | 1000 |

Interestingness Measure: χ^2

- Another measure to test correlated events: χ^2

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

- For the table on the right,

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

| | B | $\neg B$ | Σ_{row} |
|----------------|-----------|-----------|----------------|
| C | 400 (450) | 350 (300) | 750 |
| $\neg C$ | 200 (150) | 50 (100) | 250 |
| Σ_{col} | 600 | 400 | 1000 |

Expected value
Observed value

- By consulting a table of critical values of the χ^2 distribution, one can conclude that the chance for B and C to be independent is very low (< 0.01)
- χ^2 -test shows B and C are negatively correlated since the expected value is 450 but the observed is only 400
- Thus, χ^2 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!
 - ❑ But, Lift(B, C) = 8.44 >> 1 (Lift shows B and C are strongly positively correlated!)
 - ❑ $\chi^2 = 670$: Observed(BC) >> expected value (11.85)
 - ❑ *Too many null transactions may “spoil the soup”!*



| | B | $\neg B$ | Σ_{row} |
|------------------------|------|----------|-----------------------|
| C | 100 | 1000 | 1100 |
| $\neg C$ | 1000 | 100000 | 101000 |
| $\Sigma_{\text{col.}}$ | 1100 | 101000 | 102100 |

null transactions

Contingency table with expected values added

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

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 |
| $Kulczynski(A, B)$ | $\frac{1}{2} \left(\frac{s(A \cup B)}{s(A)} + \frac{s(A \cup B)}{s(B)} \right)$ | $[0, 1]$ | Yes |
| $MaxConf(A, B)$ | $\max\left\{\frac{s(A \cup B)}{s(A)}, \frac{s(A \cup B)}{s(B)}\right\}$ | $[0, 1]$ | Yes |

 χ^2 and lift are not null-invariant

Jaccard, cosine, 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

| | <i>milk</i> | $\neg\text{milk}$ | Σ_{row} |
|-----------------------|-------------|-------------------|-----------------------|
| <i>coffee</i> | <i>mc</i> | $\neg\text{mc}$ | <i>c</i> |
| $\neg\text{coffee}$ | $m\neg c$ | $\neg m\neg c$ | $\neg c$ |
| Σ_{col} | <i>m</i> | $\neg m$ | 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 | <i>mc</i> | $\neg\text{mc}$ | $m\neg c$ | $\neg m c$ | 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

| | <i>milk</i> | $\neg\text{milk}$ | Σ_{row} |
|-----------------------|-------------------|-------------------|-----------------------|
| <i>coffee</i> | <i>mc</i> | $\neg mc$ | <i>c</i> |
| $\neg coffee$ | <i>m</i> $\neg c$ | $\neg m$ $\neg c$ | $\neg c$ |
| Σ_{col} | <i>m</i> | $\neg m$ | Σ |

All 5 are null-invariant

| Data set | <i>mc</i> | $\neg mc$ | <i>m</i> $\neg c$ | $\neg m$ $\neg c$ | <i>AllConf</i> | 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

Analysis of DBLP Coauthor Relationships

- DBLP: Computer science research publication bibliographic database
 - > 3.8 million entries on authors, paper, venue, year, and other information

| ID | Author A | Author B | $s(A \cup B)$ | $s(A)$ | $s(B)$ | Jaccard | Cosine | Kulc |
|----|----------------------|----------------------|---------------|--------|--------|------------|------------|------------|
| 1 | Hans-Peter Kriegel | Martin Ester | 28 | 146 | 54 | 0.163 (2) | 0.315 (7) | 0.355 (9) |
| 2 | Michael Carey | Miron Livny | 26 | 104 | 58 | 0.191 (1) | 0.335 (4) | 0.349 (10) |
| 3 | Hans-Peter Kriegel | Joerg Sander | 24 | 146 | 36 | 0.152 (3) | 0.331 (5) | 0.416 (8) |
| 4 | Christos Faloutsos | Spiros Papadimitriou | 20 | 162 | 26 | 0.119 (7) | 0.308 (10) | 0.446 (7) |
| 5 | Hans-Peter Kriegel | Martin Pfeifle | 18 | 146 | 18 | 0.123 (6) | 0.351 (2) | 0.562 (2) |
| 6 | Hector Garcia-Molina | Wilbert Labio | 16 | 144 | 18 | 0.110 (9) | 0.314 (8) | 0.500 (4) |
| 7 | Divyakant Agrawal | Wang Hsiung | 16 | 120 | 16 | 0.133 (5) | 0.365 (1) | 0.567 (1) |
| 8 | Elke Rundensteiner | Murali Mani | 16 | 104 | 20 | 0.148 (4) | 0.351 (3) | 0.477 (6) |
| 9 | Divyakant Agrawal | Oliver Po | 12 | 120 | 12 | 0.100 (10) | 0.316 (6) | 0.550 (3) |
| 10 | Gerhard Weikum | Martin Theobald | 12 | 106 | 14 | 0.111 (8) | 0.312 (9) | 0.485 (5) |

Advisor-advisee relation: Kulc: high, Jaccard: low, cosine: middle

- Which pairs of authors are strongly related?
- Use Kulc to find Advisor-advisee, close collaborators

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 D₄ through D₆
 - D₄ is neutral & balanced; D₅ is neutral but imbalanced
 - D₆ is neutral but very imbalanced

| Data set | <i>mc</i> | $\neg mc$ | <i>m</i> $\neg c$ | $\neg m$ <i>c</i> | 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
- ❑ Exercise: Mining research collaborations from research bibliographic data
 - ❑ Find a group of frequent collaborators from research bibliographic data (e.g., DBLP)
 - ❑ Can you find the likely advisor-advisee relationship and during which years such a relationship happened?
 - ❑ Ref.: C. Wang, J. Han, Y. Jia, J. Tang, D. Zhang, Y. Yu, and J. Guo, "Mining Advisor-Advisee Relationships from Research Publication Networks", KDD'10

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

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

Summary

- ❑ Basic Concepts
 - ❑ What Is Pattern Discovery? Why Is It Important?
 - ❑ Basic Concepts: Frequent Patterns and Association Rules
 - ❑ Compressed Representation: Closed Patterns and Max-Patterns
- ❑ 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
 - ❑ FP-Growth: A Frequent Pattern-Growth Approach
 - ❑ Mining Closed Patterns
- ❑ Pattern Evaluation
 - ❑ Interestingness Measures in Pattern Mining
 - ❑ Interestingness Measures: Lift and χ^2
 - ❑ Null-Invariant Measures
 - ❑ Comparison of Interestingness Measures

Recommended Readings (Basic Concepts)

- R. Agrawal, T. Imielinski, and A. Swami, “Mining association rules between sets of items in large databases”, in Proc. of SIGMOD'93
- R. J. Bayardo, “Efficiently mining long patterns from databases”, in Proc. of SIGMOD'98
- N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, “Discovering frequent closed itemsets for association rules”, in Proc. of ICDT'99
- J. Han, H. Cheng, D. Xin, and X. Yan, “Frequent Pattern Mining: Current Status and Future Directions”, Data Mining and Knowledge Discovery, 15(1): 55-86, 2007

Recommended Readings (Efficient Pattern Mining Methods)

- R. Agrawal and R. Srikant, “Fast algorithms for mining association rules”, VLDB'94
- A. Savasere, E. Omiecinski, and S. Navathe, “An efficient algorithm for mining association rules in large databases”, VLDB'95
- J. S. Park, M. S. Chen, and P. S. Yu, “An effective hash-based algorithm for mining association rules”, SIGMOD'95
- S. Sarawagi, S. Thomas, and R. Agrawal, “Integrating association rule mining with relational database systems: Alternatives and implications”, SIGMOD'98
- M. J. Zaki, S. Parthasarathy, M. Ogihsara, and W. Li, “Parallel algorithm for discovery of association rules”, Data Mining and Knowledge Discovery, 1997
- J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation”, SIGMOD'00
- M. J. Zaki and Hsiao, “CHARM: An Efficient Algorithm for Closed Itemset Mining”, SDM'02
- J. Wang, J. Han, and J. Pei, “CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets”, KDD'03
- C. C. Aggarwal, M.A., Bhuiyan, M. A. Hasan, “Frequent Pattern Mining Algorithms: A Survey”, in Aggarwal and Han (eds.): Frequent Pattern Mining, Springer, 2014

Recommended Readings (Pattern Evaluation)

- C. C. Aggarwal and P. S. Yu. A New Framework for Itemset Generation. PODS'98
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97
- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94
- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE'03
- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02
- T. Wu, Y. Chen and J. Han, Re-Examination of Interestingness Measures in Pattern Mining: A Unified Framework, Data Mining and Knowledge Discovery, 21(3):371-397, 2010

