S2-20_DSECFZC415: Data Mining
(Lecture #9 – Association Analysis)

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

- *The slides presented here are obtained from the authors of the books and from various other contributors. I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified a few slides to suit the requirements of the course.*

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

FP-growth Algorithm

Association Analysis *(Review)*

Association analysis measures the strength of co-occurrence between one item and another.

- The objective of this class of data mining algorithms is not to predict an occurrence of an item, like classification or regression do, but to find usable patterns in the co-occurrences of the items.
- Association rules learning is a branch of an unsupervised learning process that discovers hidden patterns in data, in the form of easily recognizable *rules*

Association algorithms are widely used in retail analysis of transactions, recommendation engines, and online clickstream analysis across web pages.

- One of the popular applications of this technique is called *market basket analysis*, which finds co-occurrences of one retail item with another item within the same retail purchase transaction

Retailer can take advantage of this association for bundle pricing, product placement, and even shelf space optimization within the store layout.

Association Rule Mining *(Review)*

Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	Items
1	Bread, Milk
2	Bread, Diaper, Butter, Beans
3	Milk, Diaper, Butter, Coke
4	Bread, Milk, Diaper, Butter
5	Bread, Milk, Diaper, Coke

Example of Association Rules

{Diaper} → {Butter},
{Milk, Bread} → {Beans, Coke},
{Butter, Bread} → {Milk},

Implication means co-occurrence, not causality!

Definition: Frequent Itemset *(Review)*

Itemset

- A collection of one or more items
  - Example: {Milk, Bread, Diaper}
- k-itemset
  - An itemset that contains k items

TID	Items
1	Bread, Milk
2	Bread, Diaper, Butter, Beans
3	Milk, Diaper, Butter, Coke
4	Bread, Milk, Diaper, Butter
5	Bread, Milk, Diaper, Coke

Support count (σ)

- Frequency of occurrence of an itemset
- E.g. σ({Milk, Bread,Diaper}) = 2

Support

- Fraction of transactions that contain an itemset
- E.g. s({Milk, Bread, Diaper}) = 2/5

Frequent Itemset

- An itemset whose support is greater than or equal to a *minsup* threshold

Definition: Association Rule *(Review)*

- **Association Rule**
  - An implication expression of the form X → Y, where X and Y are itemsets
  - Example:
    {Milk, Diaper} → {Butter}

TID	Items
1	Bread, Milk
2	Bread, Diaper, Butter, Beans
3	Milk, Diaper, Butter, Coke
4	Bread, Milk, Diaper, Butter
5	Bread, Milk, Diaper, Coke

- **Rule Evaluation Metrics**
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

Example:

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

$$s = \frac{\sigma(\text{Milk, Diaper, Butter})}{|\text{T}|} = \frac{2}{5} = 0.4$$

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

Mining Association Rules

Two-step approach:

1. Frequent Itemset Generation
   - Generate all itemsets whose support ≥ minsup

2. Rule Generation
   - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

Frequent itemset generation is computationally expensive

- Can we avoid it?

Bottleneck of Frequent-pattern Mining

Multiple database scans are costly

Mining long patterns needs many passes of scanning and generates lots of candidates

- To find frequent itemset $i_1 i_2 \ldots i_{100}$
  - # of scans: 100
  - # of Candidates: ${}^{100}C_1 + {}^{100}C_2 + \ldots + {}^{100}C_{100} = 2^{100} - 1 = 1.27*10^{30}$!

Bottleneck: candidate-generation-and-test

- Can we avoid candidate generation?

Mining Frequent Patterns Without Candidate Generation

Bottlenecks of the Apriori approach

- Breadth-first (i.e., level-wise) search
- Candidate generation and test
- Often generates a huge number of candidates

The FPGrowth Approach (J. Han, J. Pei, and Y. Yin, SIGMOD' 00)

- Depth-first search
- Avoid explicit candidate generation

Grow long patterns from short ones using local frequent items (Major philosophy behind FPGrowth)

- "abc" is a frequent pattern
- Get all transactions having "abc": DB|abc
- "d" is a local frequent item in DB|abc → abcd is a frequent pattern

FP-growth Algorithm

Use a compressed representation of the database using an FP-tree

Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent itemsets

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

Construct FP-tree from a Transaction Database

min_support = 3

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

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

Construct FP-tree from a Transaction Database

min_support = 3

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

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

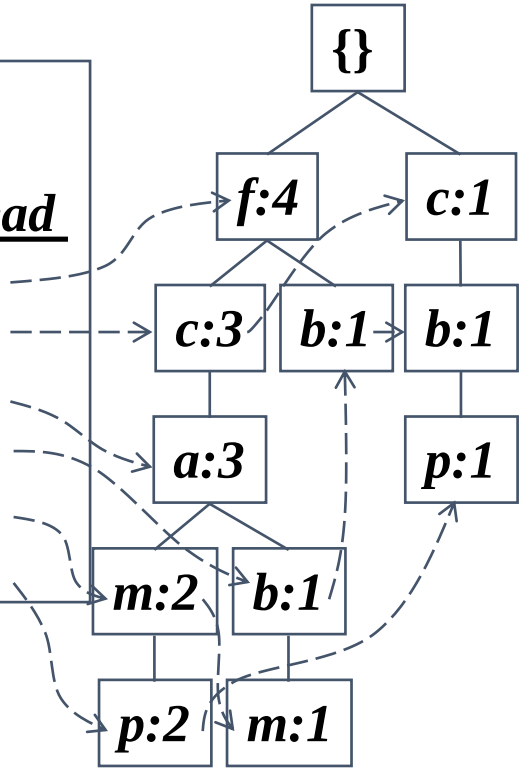
Construct FP-tree from a Transaction Database

min_support = 3

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

Header Table

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



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

Find Patterns Having P From P-conditional Database

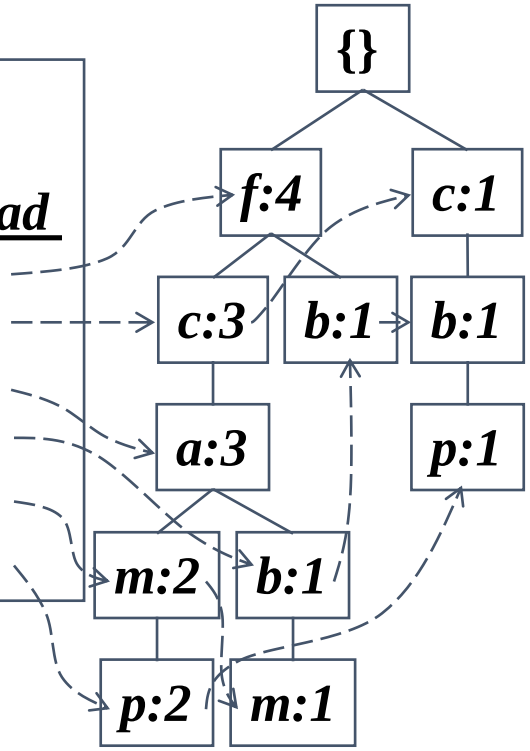
Starting at the frequent item header table in the FP-tree

Traverse the FP-tree by following the link of each frequent item *p*

Accumulate all of *transformed prefix paths* of item *p* to form *p*'s conditional pattern base

Header Table

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



***Conditional* pattern bases**

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

Benefits of the FP-tree Structure

Completeness

- Preserve complete information for frequent pattern mining
- Never break a long pattern of any transaction

Compactness

- Reduce irrelevant info—infrequent items are gone
- Items in frequency descending order: the more frequently occurring, the more likely to be shared
- Never be larger than the original database (not count node-links and the *count* field)

Partition Patterns and Databases

Frequent patterns can be partitioned into subsets according to f-list

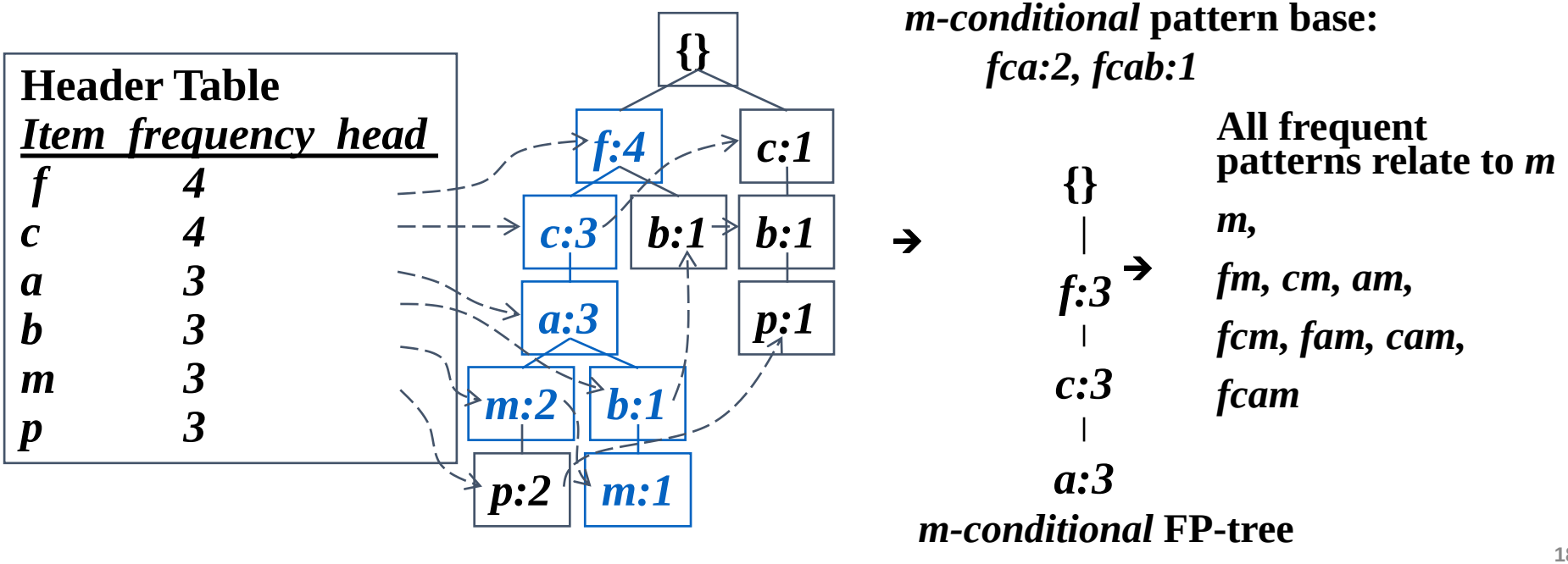
- F-list=f-c-a-b-m-p
- Patterns containing p
- Patterns having m but no p
- …
- Patterns having c but no a nor b, m, p
- Pattern f

Completeness and non-redundency

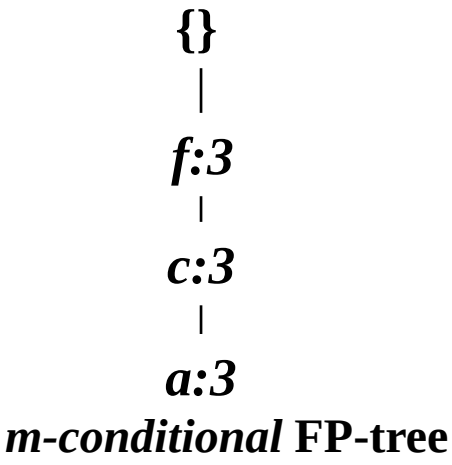
From Conditional Pattern-bases to Conditional FP-trees

For each pattern-base

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

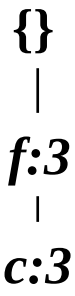


Recursion: Mining Each Conditional FP-tree



m-conditional FP-tree

Cond. pattern base of "am": (fc:3)



am-conditional FP-tree

Cond. pattern base of "cm": (f:3)

cm-conditional FP-tree

Cond. pattern base of "cam": (f:3)

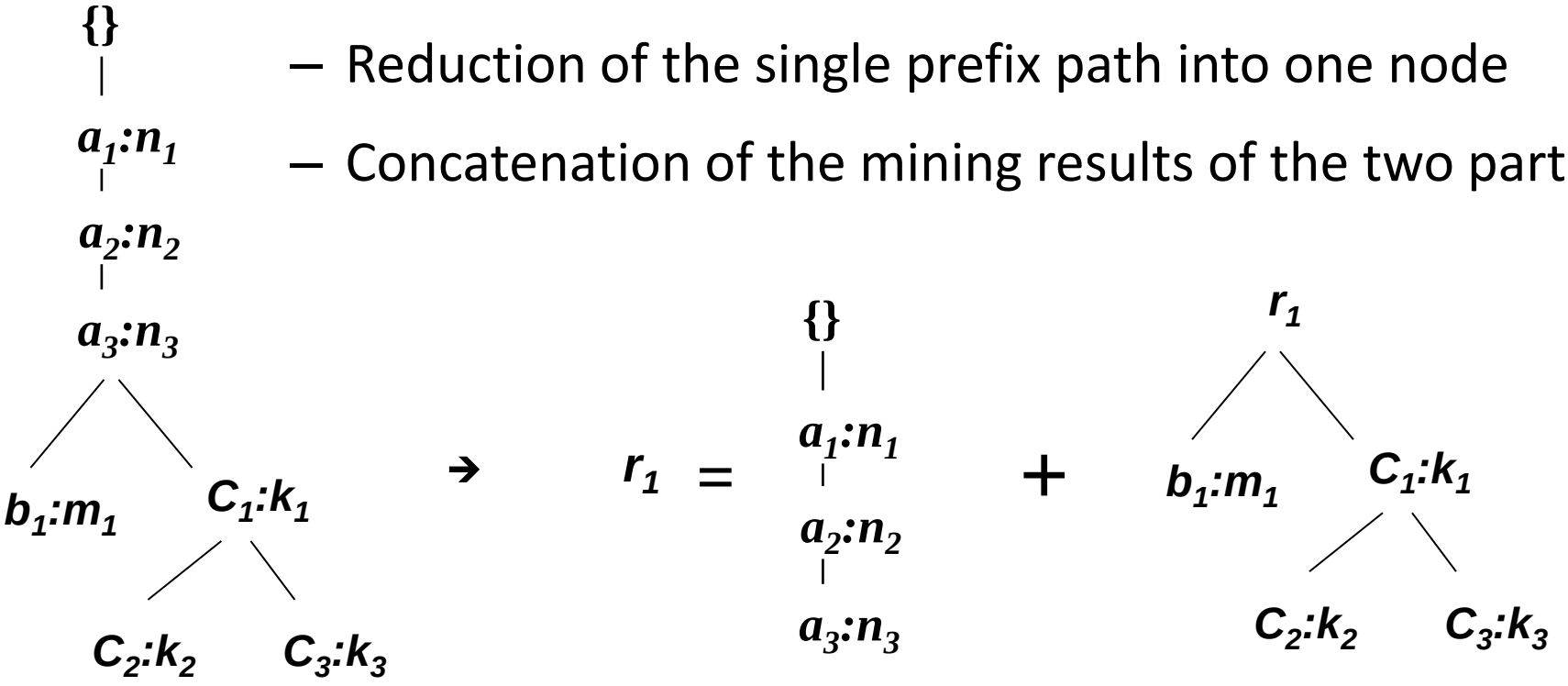
cam-conditional FP-tree

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



Mining Frequent Patterns With FP-trees

Idea: Frequent pattern growth

- Recursively grow frequent patterns by pattern and database partition

Method

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

Why Is FP-Growth the Winner?

Divide-and-conquer:

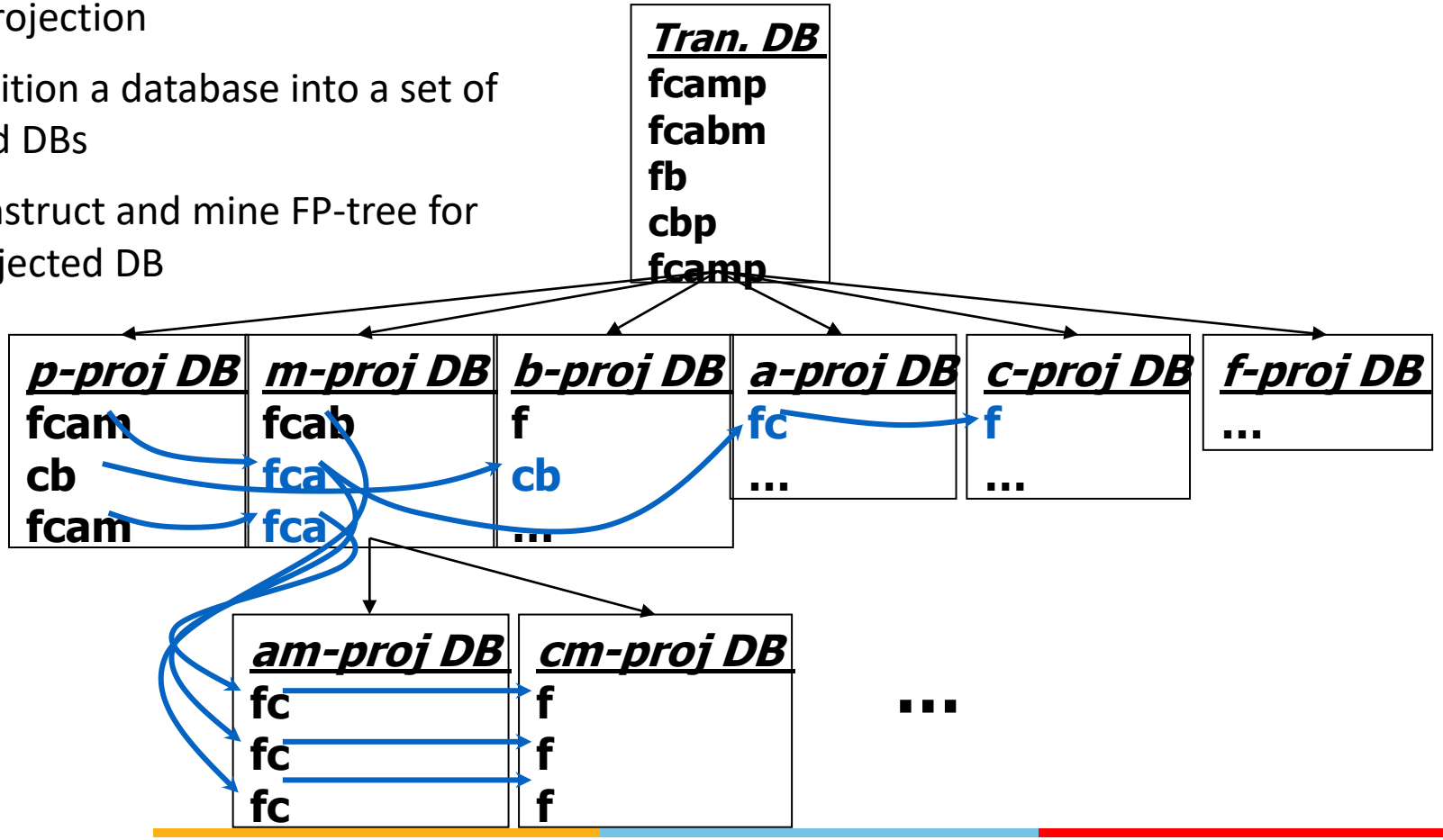
- decompose both the mining task and DB according to the frequent patterns obtained so far
- leads to focused search of smaller databases

Other factors

- no candidate generation, no candidate test
- compressed database: FP-tree structure
- no repeated scan of entire database
- basic ops—counting local freq items and building sub FP-tree, no pattern search and matching

Scaling FP-growth by Database Projection

- What about if FP-tree cannot fit in memory?
  - DB projection
- First partition a database into a set of projected DBs
- Then construct and mine FP-tree for each projected DB



innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Mining association rules

Rule Generation

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
  - If {A,B,C,D} is a frequent itemset, candidate rules:

ABC →D,	ABD →C,	ACD →B,	BCD →A,
AB →CD,	AC → BD,	AD → BC,	BC →AD,
BD →AC,	CD →AB,		
A →BCD,	B →ACD,	C →ABD,	D →ABC

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \emptyset$ and $\emptyset \rightarrow L$)

Rule Generation

- How to efficiently generate rules from frequent itemsets?
  - In general, confidence does not have an anti-monotone property

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

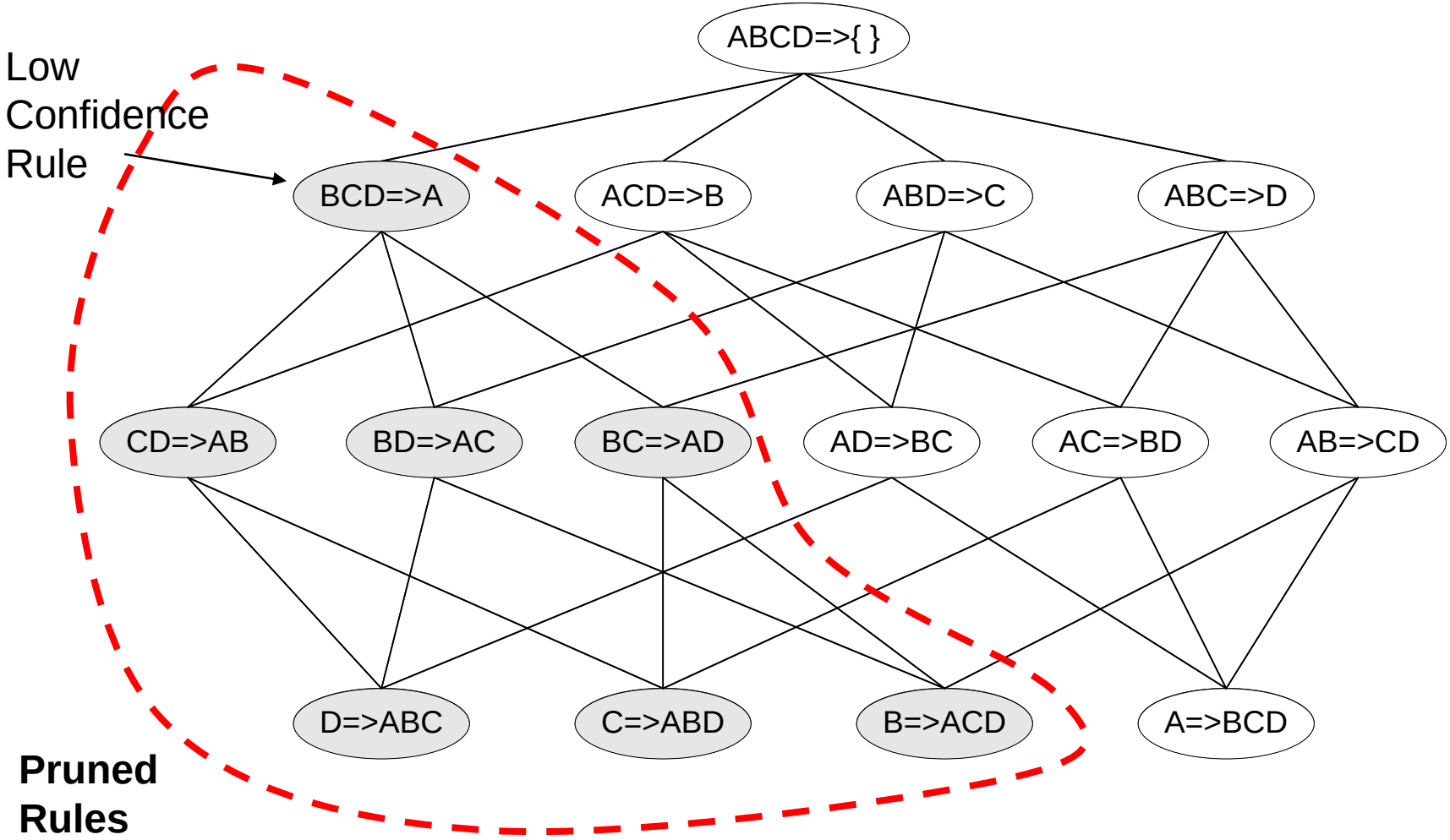
  - But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., L = {A,B,C,D}:

$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

    - Confidence is anti-monotone in this case

Rule Generation for Apriori Algorithm

Lattice of rules

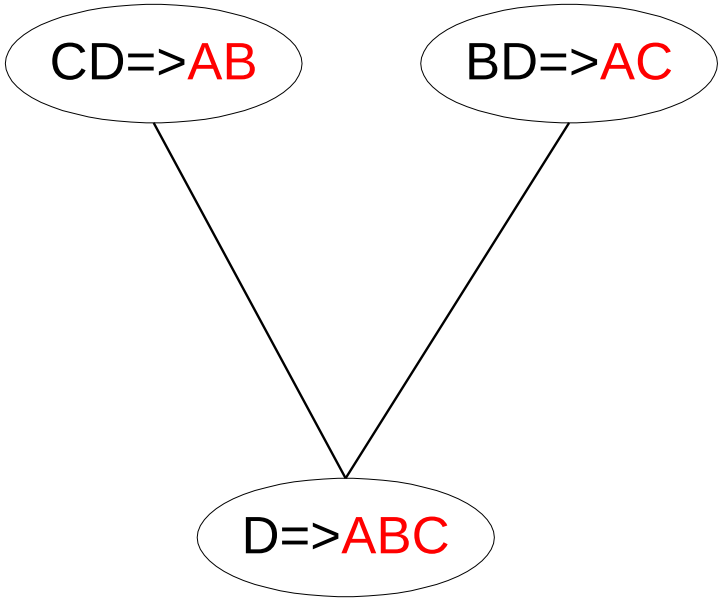


Rule Generation with Apriori Algorithm

- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent

- join(CD=>AB,BD=>AC) would produce the candidate rule D => ABC

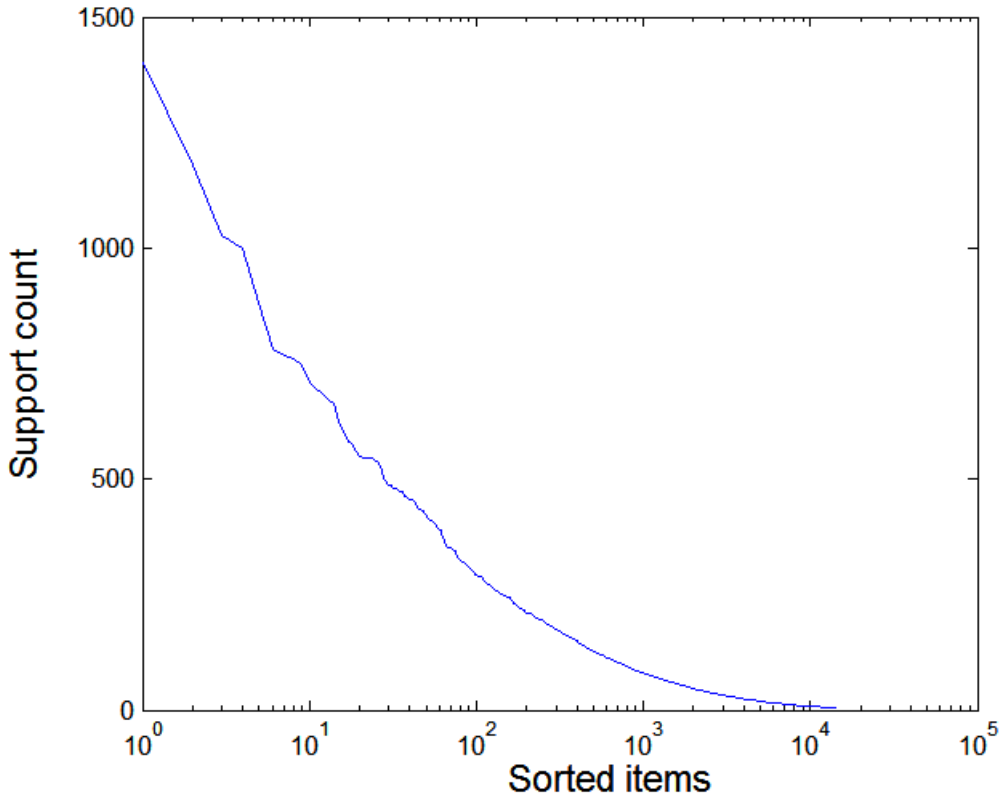
- Prune rule D=>ABC if its subset AD=>BC does not have high confidence



Effect of Support Distribution

- Many real data sets have skewed support distribution

Support distribution of a retail data set



Effect of Support Distribution

- How to set the appropriate *minsup* threshold?
  - If *minsup* is set too high, we could miss itemsets involving interesting rare items (e.g., expensive products)
  - If *minsup* is set too low, it is computationally expensive and the number of itemsets is very large

- Using a single minimum support threshold may not be effective

Multiple Minimum Support

- How to apply multiple minimum supports?
  - MS(i): minimum support for item i
  - e.g.: MS(Milk)=5%, MS(Coke) = 3%,
    MS(Broccoli)=0.1%, MS(Salmon)=0.5%
  - MS({Milk, Broccoli}) = min (MS(Milk), MS(Broccoli))
    = 0.1%
  - Challenge: Support is no longer anti-monotone
    - Suppose: Support(Milk, Coke) = 1.5% and
      Support(Milk, Coke, Broccoli) = 0.5%
    - {Milk,Coke} is infrequent but {Milk,Coke,Broccoli} is frequent

Pattern Evaluation

- Association rule algorithms tend to produce too many rules
  - many of them are uninteresting or redundant
  - Redundant if {A,B,C} → {D} and {A,B} → {D} have same support & confidence

- Interestingness measures can be used to prune/rank the derived patterns

- In the original formulation of association rules, support & confidence are the only measures used

Computing Interestingness Measure

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

Contingency table for $X \rightarrow Y$

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

f_{11}: support of X and Y
f_{10}: support of $\underline{X}$ and $\overline{Y}$
f_{01}: support of $\underline{X}$ and $\underline{Y}$
f_{00}: support of $\overline{X}$ and $\overline{Y}$

Used to define various measures

- support, confidence, lift, Gini, J-measure, etc.

Drawback of Confidence

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

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.9

$\Rightarrow$ Although confidence is high, rule is misleading

$\Rightarrow$ P(Coffee|$\overline{\text{Tea}}$) = 0.9375

Statistical Independence

- Population of 1000 students
  - 600 students know how to swim (S)
  - 700 students know how to bike (B)
  - 420 students know how to swim and bike (S,B)

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

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

Statistical-based Measures

- Measures that take into account statistical dependence

$$Lift = \frac{P(Y \mid X)}{P(Y)}$$

$$Interest = \frac{P(X,Y)}{P(X)P(Y)}$$

Example: Lift/Interest

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

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.9

$\Rightarrow$ Lift = 0.75/0.9= 0.8333 (< 1, therefore is negatively associated)

Drawback of Lift & Interest

	Y	$\bar{Y}$	
X	10	0	10
$\bar{X}$	0	90	90
	10	90	100

	Y	$\bar{Y}$	
X	90	0	90
$\bar{X}$	0	10	10
	90	10	100

$$Lift = \frac{0.1}{(0.1)(0.1)} = 10$$

$$Lift = \frac{0.9}{(0.9)(0.9)} = 1.11$$

Statistical independence:

If P(X,Y)=P(X)P(Y) => Lift = 1

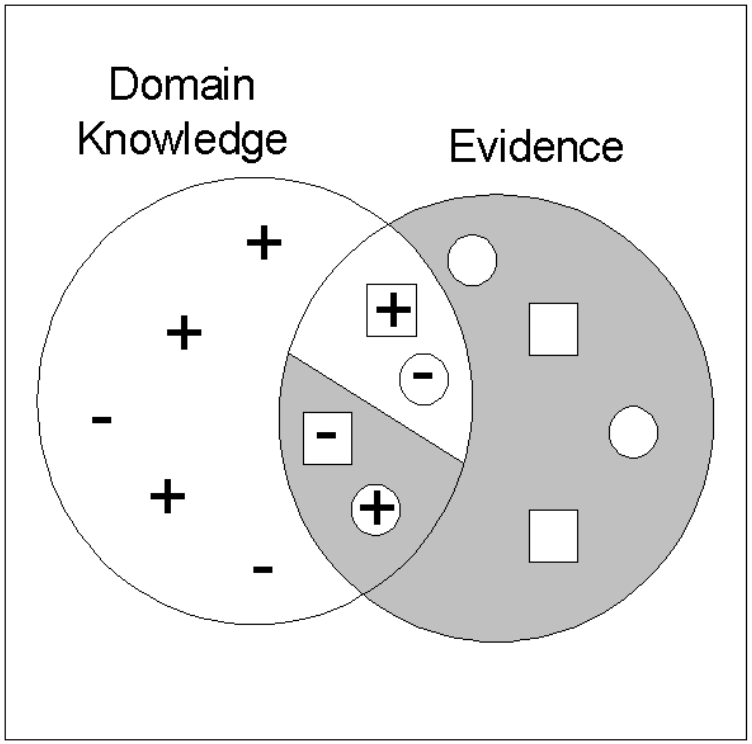
Subjective Interestingness Measure

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

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

Interestingness via Unexpectedness

- Need to model expectation of users (domain knowledge)



+ Pattern expected to be frequent

- Pattern expected to be infrequent

Pattern found to be frequent

Pattern found to be infrequent

+ - Expected Patterns

- + Unexpected Patterns

- Need to combine expectation of users with evidence from data (i.e., extracted patterns)

Prescribed Text Books

	Author(s), Title, Edition, Publishing House
T1	Data Mining: Concepts and Techniques, Third Edition by Jiawei Han, Micheline Kamber and Jian Pei Morgan Kaufmann Publishers
R1	Tan P. N., Steinbach M & Kumar V. “Introduction to Data Mining” Pearson Education
R2	Predictive Analytics and Data Mining: Concepts and Practice with RapidMiner by Vijay Kotu and Bala Deshpande Morgan Kaufmann Publishers