Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

What Is Frequent Pattern Analysis?

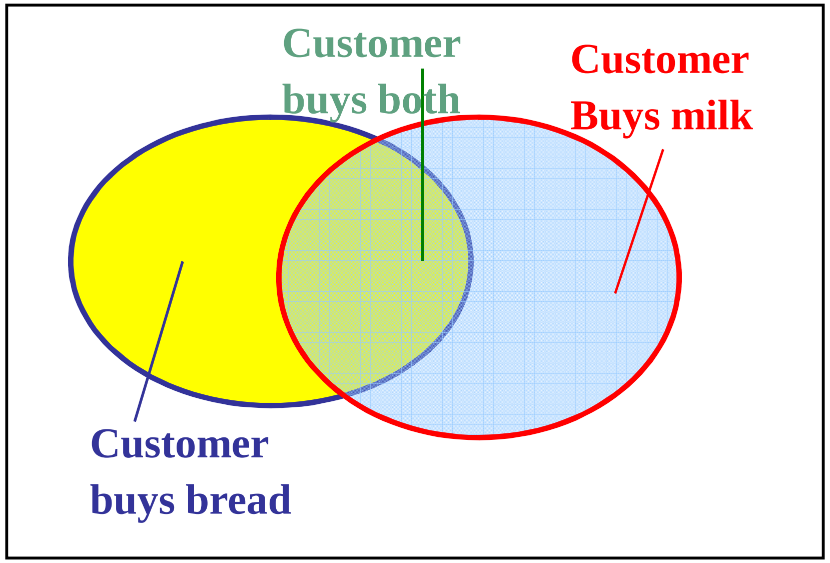
- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What products were often purchased together? — Bread and milk
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Why Is Freq. Pattern Mining Important?

- Discloses an intrinsic and important property of data sets
- Forms the foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: associative classification
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression: fascicles
  - Broad applications

Basic Concepts: Frequent Patterns and Association Rules

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F



- Itemset X = $\{x_1, \ldots, x_k\}$
- Find all the rules $X \rightarrow Y$ with minimum support and confidence
  - support, *s*, probability that a transaction contains $X \cup Y$
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

Let $sup_{min} = 50\%$, $conf_{min} = 50\%$

Freq. Pat.: {A:3, B:3, D:4, E:3, AD:3}

Association rules:

$A \rightarrow D$ (60%, 100%)

$D \rightarrow A$ (60%, 75%)

Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$ contains $({}_{100}{}^{1}) + ({}_{100}{}^{2}) + \ldots + ({}_{1}{}^{1}{}_{0}{}^{0}{}_{0}{}^{0}) = 2^{100} - 1 = 1.27 \times 10^{30}$ sub-patterns!
- Solution: *Mine closed patterns and max-patterns instead*
- An itemset X is closed if X is *frequent* and there exists *no super-pattern* Y ɔ X, *with the same support* as X
- An itemset X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ɔ X
- Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

Closed Patterns and Max-Patterns

- Exercise. DB = $\{<a_1, \ldots, a_{100}>, < a_1, \ldots, a_{50}>\}$
  - Min_sup = 1.
- What is the set of closed itemset?
  - $<a_1, \ldots, a_{100}>$: 1
  - $< a_1, \ldots, a_{50}>$: 2
- What is the set of max-pattern?
  - $<a_1, \ldots, a_{100}>$: 1
- What is the set of all patterns?
  - !!

Frequent pattern mining : A roadmap

- Based on the completeness of patterns to be mined
- Based on the levels of abstraction involved in the rule set
  - Single level, multi level association rules
- Based on the number of data dimensions involved in the rule
  - Single dimensional and multidimensional
- Based on the types of values handled in the rule
  - Boolean and quantitative
- Based on the kinds of rules to be mined
  - Association rules and correlation rules
  - Strong gradient relationships
- Based on the kinds of patterns to be mined
  - Frequent itemset, sequential, and structured pattern mining

Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Scalable Methods for Mining Frequent Patterns

- The downward closure property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Scalable mining methods: Three major approaches
  - Apriori (Agrawal & Srikant@VLDB'94)
  - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
  - Vertical data format approach (Charm—Zaki & Hsiao @SDM'02)

Apriori: A Candidate Generation-and-Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

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

1st scan $\rightarrow$

C_1

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

$\rightarrow$ L_1

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

C_2

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

2nd scan $\leftarrow$

C_2

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

$\leftarrow$ L_2

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

C_3

Itemset
{B, C, E}

3rd scan $\rightarrow$

L_3

Itemset	sup
{B, C, E}	2

The Apriori Algorithm

- Pseudo-code:

```
$C_k$: Candidate itemset of size k
$L_k$ : frequent itemset of size k

$L_1$ = {frequent items};
**for** ($k$ = 1; $L_k$ !=$\emptyset$; $k$++) **do begin**
    $C_{k+1}$ = candidates generated from $L_k$;
    **for each** transaction $t$ in database do
            increment the count of all candidates in $C_{k+1}$
        that are contained in $t$
    $L_{k+1}$ = candidates in $C_{k+1}$ with min_support
    **end**
**return** $\cup_k L_k$;
```

Important Details of Apriori

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- How to count supports of candidates?
- Example of Candidate-generation
  - L_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: $L_3 * L_3$
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - C_4={*abcd*}

How to Generate Candidates?

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

```
insert into $C_k$
select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$
from $L_{k-1}$ p, $L_{k-1}$ q
where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
```

- Step 2: pruning

```
forall itemsets c in $C_k$ do
    forall (k-1)-subsets s of c do
        if (s is not in $L_{k-1}$) then delete c from $C_k$
```

How to Count Supports of Candidates?

- Why counting supports of candidates a problem?
  - The total number of candidates can be very huge
  - One transaction may contain many candidates
- Method:
  - Candidate itemsets are stored in a *hash-tree*
  - *Leaf* node of hash-tree contains a list of itemsets and counts
  - *Interior* node contains a hash table
  - *Subset function*: finds all the candidates contained in a transaction

Generating association rules from frequent itemsets

- Strong association rules satisfy both minimum support and minimum confidence
  - Confidence(A => B) = P(B/A) = support_count(A *union* B)/support_count(A)
- Steps
  - For each frequent itemset l, generate all nonempty subsets of l
  - For every nonempty subsets of , output s=> $l - s$ if
    - support_count(l)/support_count(s) > = min_conf.

Challenges of Frequent Pattern Mining

- Challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

Partition: Scan Database Only Twice

- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns

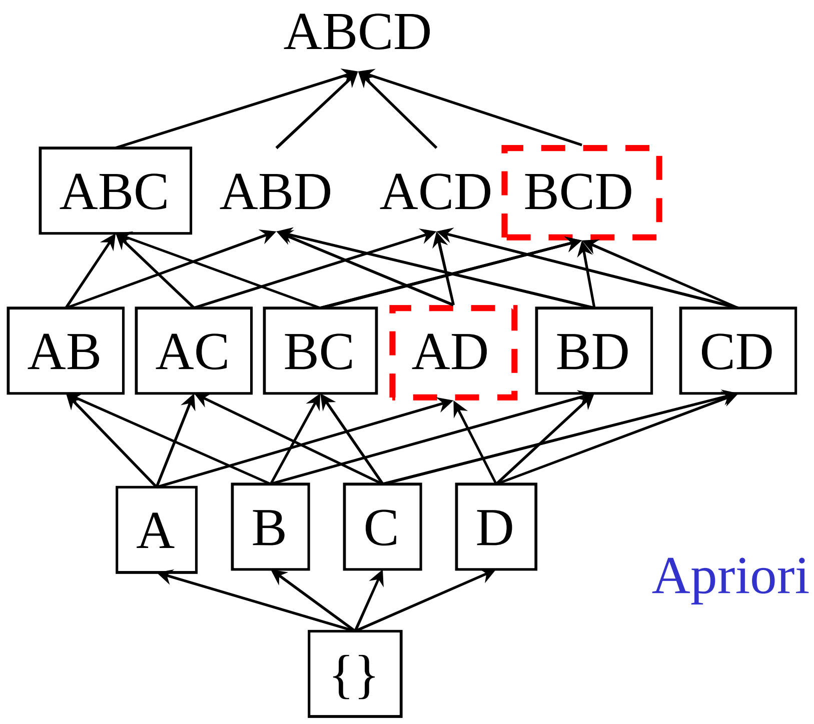
Hash based technique - Reduce the Number of Candidates

- A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
  - Candidates: a, b, c, d, e
  - Hash entries: {ab, ad, ae} {bd, be, de} …
  - Frequent 1-itemset: a, b, d, e
  - ab is not a candidate 2-itemset if the sum of count of {ab, ad, ae} is below support threshold

Sampling for Frequent Patterns

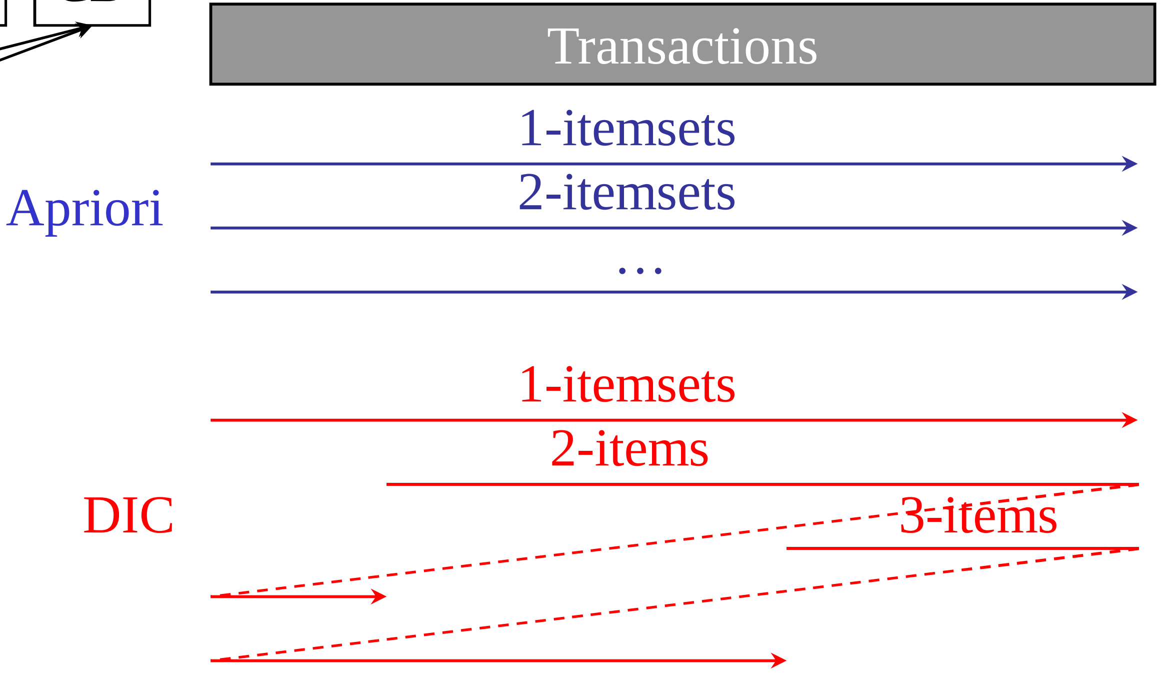
- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only *borders* of closure of frequent patterns are checked
  - Example: check *abcd* instead of *ab, ac, …, etc.*
- Scan database again to find missed frequent patterns

Dynamic itemset counting - Reduce Number of Scans



Itemset lattice

- Once both A and D are determined frequent, the counting of AD begins
- Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins



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: $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 = 1.27 * 10^{30}$!
- Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?

Mining Frequent Patterns Without Candidate Generation

- Grow long patterns from short ones using local frequent items
  - "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

Construct FP-tree from a Transaction Database

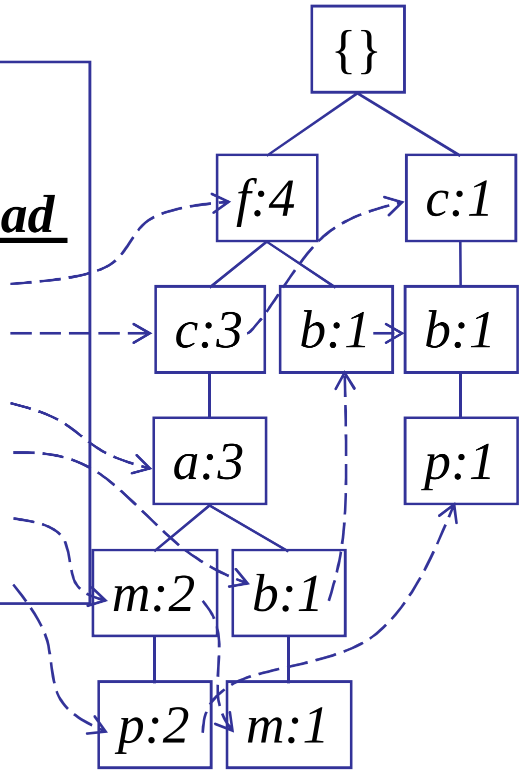
TID	*Items bought*	*(ordered) frequent items*
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o, w*}	{*f, b*}
400	{*b, c, k, s, p*}	{*c, b, p*}
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}

min_support = 3

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

Header Table

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



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

Benefits of the FP-tree Structure

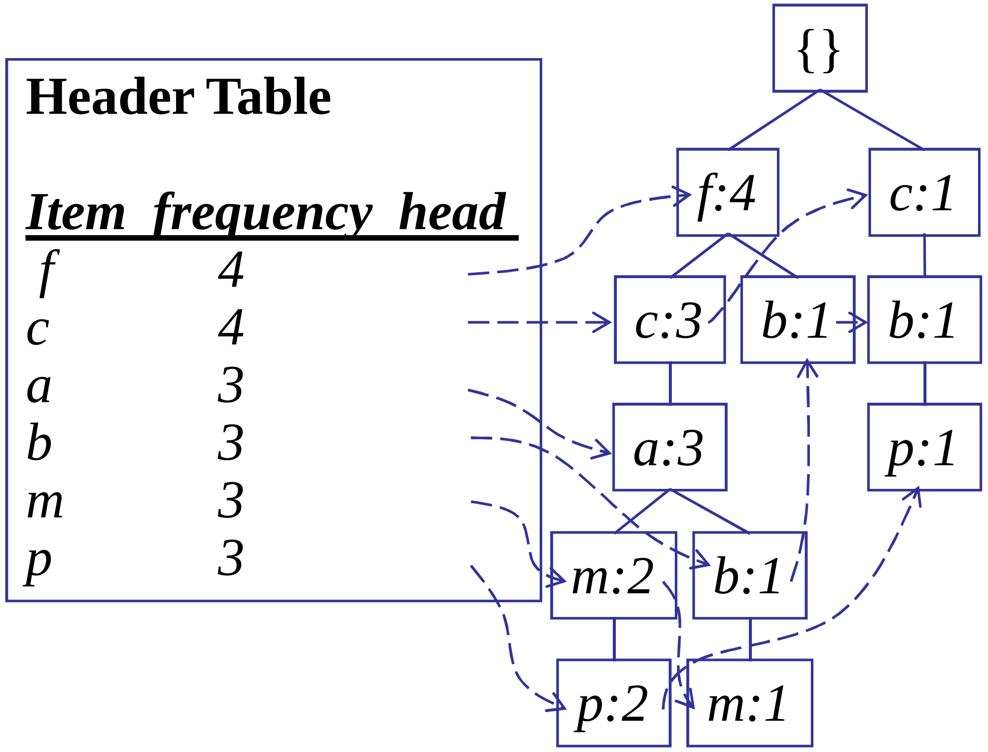
- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database

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

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

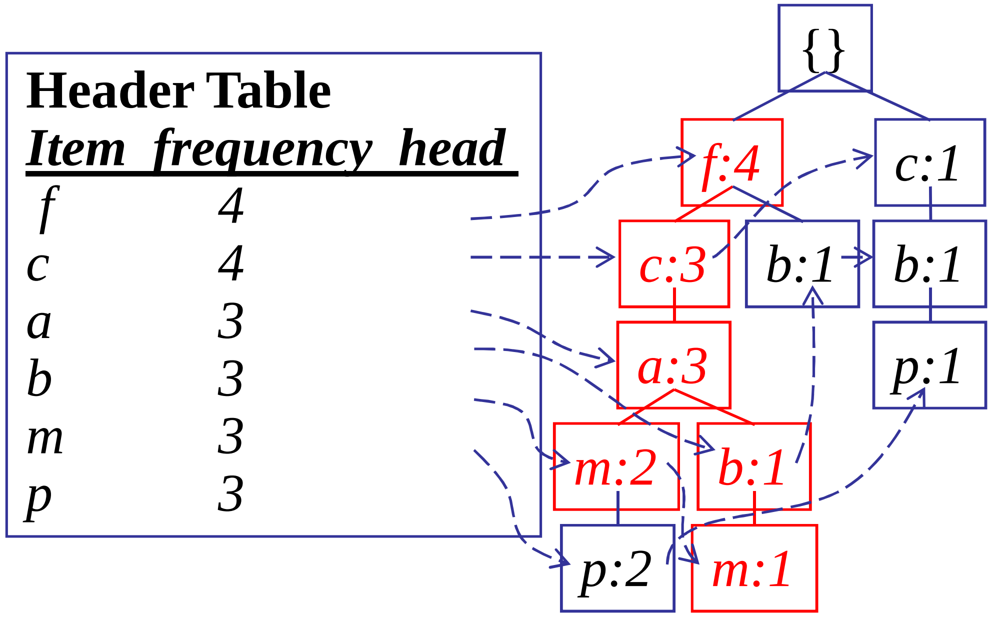
Item	*frequency*	*head*
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p	*3*	

***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*

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



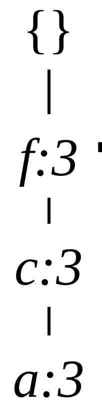
Header Table

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

m-conditional **pattern base:**

fca:2, fcab:1

$\rightarrow$



m-conditional **FP-tree**

$\rightarrow$

All frequent patterns relate to *m*

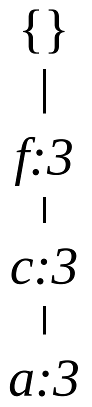
m,

fm, cm, am,

fcm, fam, cam,

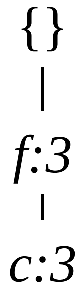
fcam

Recursion: Mining Each Conditional FP-tree



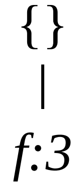
m-conditional **FP-tree**

Cond. pattern base of “am”: (fc:3)



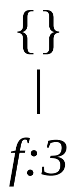
am-conditional **FP-tree**

Cond. pattern base of “cm”: (f:3)



cm-conditional **FP-tree**

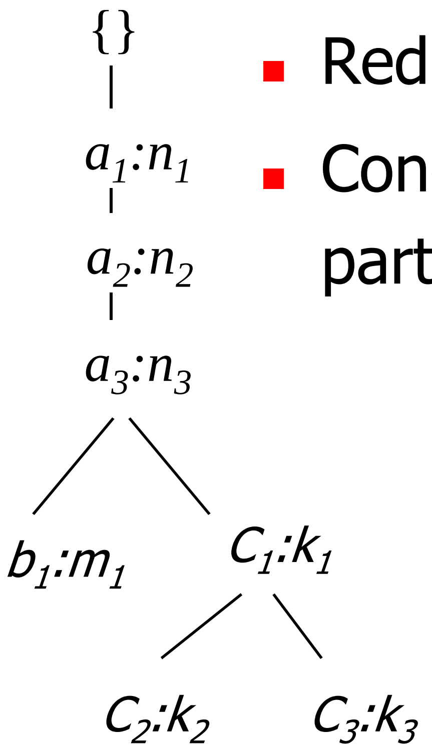
Cond. pattern base of “cam”: (f:3)

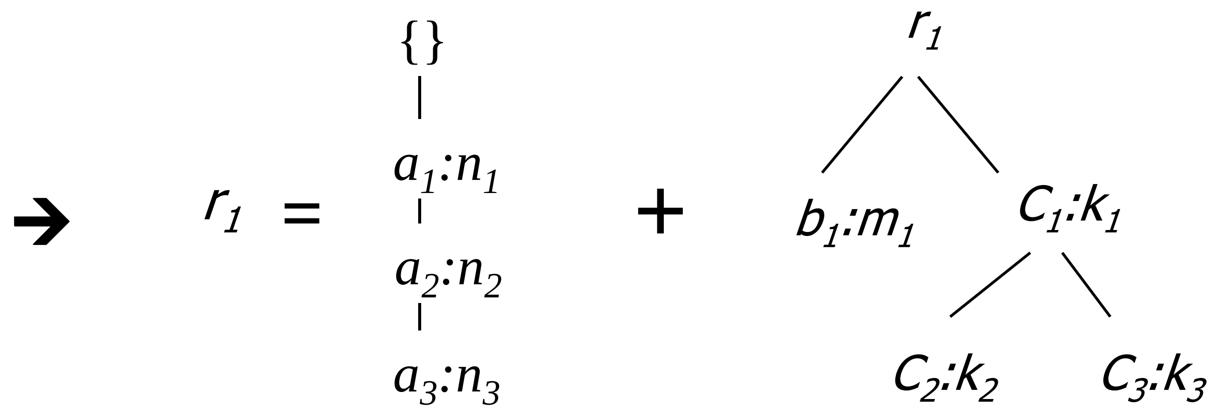


cam-conditional **FP-tree**

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

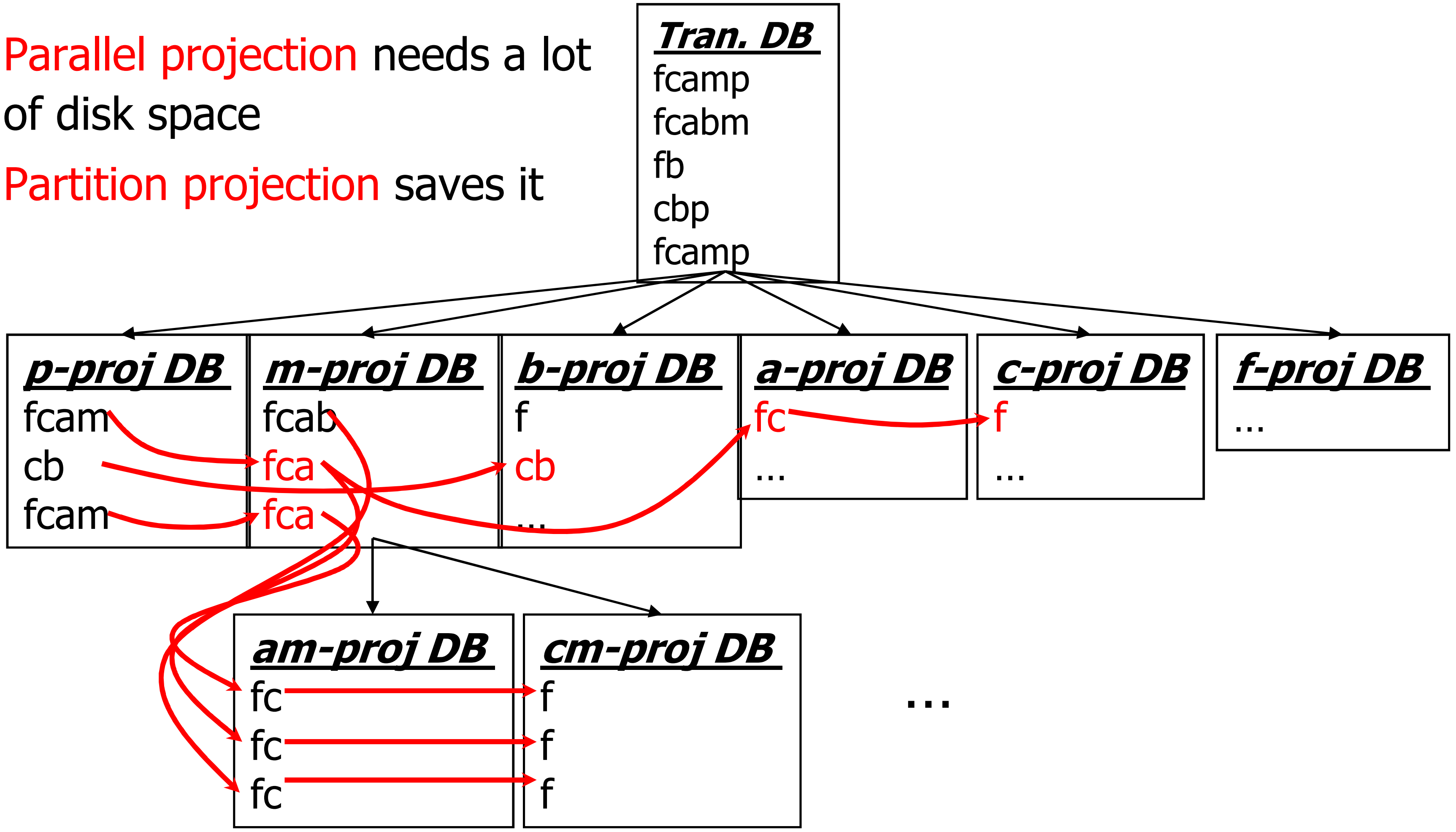
- Idea: Frequent pattern growth
  - Recursively grow frequent patterns by pattern and database partition
- Method
  - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
  - Repeat the process on each newly created conditional FP-tree
  - Until the resulting 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

Scaling FP-growth by DB Projection

- 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
- Parallel projection vs. Partition projection techniques
    - Parallel projection is space costly

Partition-based Projection

- Parallel projection needs a lot of disk space
- Partition projection saves it



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

Mining Frequent Closed Patterns

- Flist: list of all frequent items in support ascending order
  - Flist: d-a-f-e-c
- Divide search space
  - Patterns having d
  - Patterns having d but no a, etc.
- Find frequent closed pattern recursively
  - Every transaction having d also has cfa → cfad is a frequent closed pattern

Min_sup=2

TID	Items
10	a, c, d, e, f
20	a, b, e
30	c, e, f
40	a, c, d, f
50	c, e, f

Mining Closed Itemsets by Pattern-Growth

- Itemset merging: if Y appears in every occurrence of X, then Y is merged with X
- Sub-itemset pruning: if Y ɔ X, and sup(X) = sup(Y), X and all of X's descendants in the set enumeration tree can be pruned
- Item skipping: if a local frequent item has the same support in several header tables at different levels, one can prune it from the header table at higher levels
- Efficient subset checking

Mining by Exploring Vertical Data Format

- Vertical format: $t(AB) = \{T_{11}, T_{25}, \ldots\}$
  - tid-list: list of trans.-ids containing an itemset
- Deriving closed patterns based on vertical intersections
  - $t(X) = t(Y)$: X and Y always happen together
  - $t(X) \subset t(Y)$: transaction having X always has Y
- Using diffset to accelerate mining
  - Only keep track of differences of tids
  - $t(X) = \{T_1, T_2, T_3\}$, $t(XY) = \{T_1, T_3\}$
  - Diffset $(XY, X) = \{T_2\}$

Chapter 5: Mining Frequent Patterns, Association and Correlations

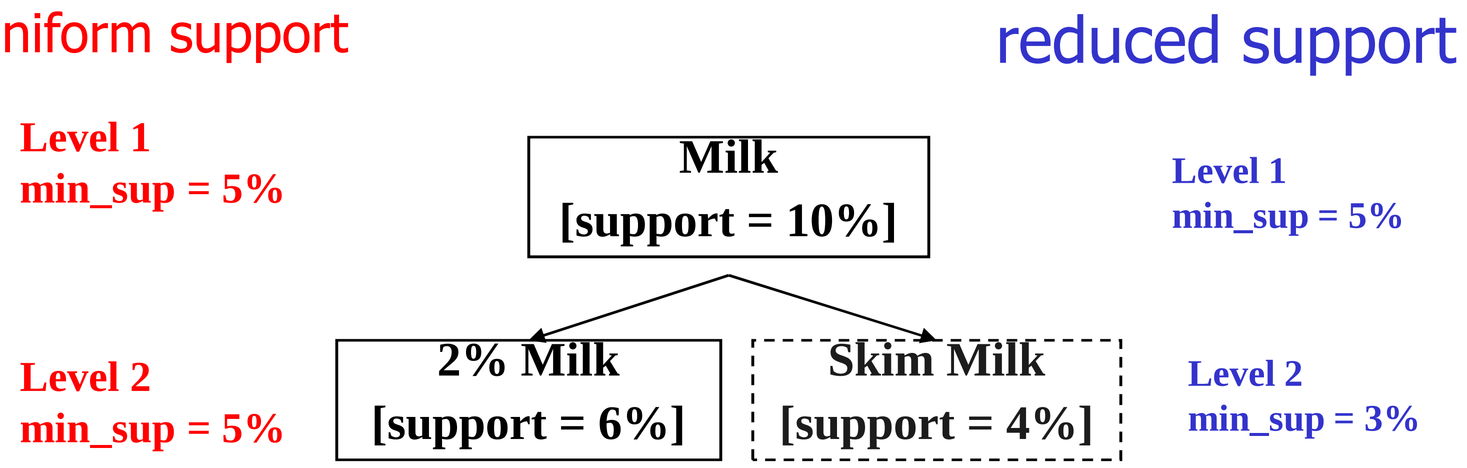
- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Mining Various Kinds of Association Rules

- Mining multilevel association
- Miming multidimensional association
- Mining quantitative association
- Mining interesting correlation patterns

Mining Multiple-Level Association Rules

- Items often form hierarchies
- Flexible support settings
  - Items at the lower level are expected to have lower support
- Exploration of *shared* multi-level mining



Multi-level Association: Redundancy Filtering

- Some rules may be redundant due to "ancestor" relationships between items.
- Example
  - milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%]
  - 2% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%]
- We say the first rule is an ancestor of the second rule.
- A rule is redundant if its support is close to the "expected" value, based on the rule's ancestor.

Mining Multi-Dimensional Association

- Single-dimensional rules:

  buys(X, "milk") $\Rightarrow$ buys(X, "bread")

- Multi-dimensional rules: $\geq$ 2 dimensions or predicates
  - Inter-dimension assoc. rules (*no repeated predicates*)

    age(X,"19-25") $\wedge$ occupation(X,"student") $\Rightarrow$ buys(X, "coke")
  - hybrid-dimension assoc. rules (*repeated predicates*)

    age(X,"19-25") $\wedge$ buys(X, "popcorn") $\Rightarrow$ buys(X, "coke")
- Categorical Attributes: finite number of possible values, no ordering among values—data cube approach
- Quantitative Attributes: numeric, implicit ordering among values—discretization, clustering, and gradient approaches

Mining Quantitative Associations

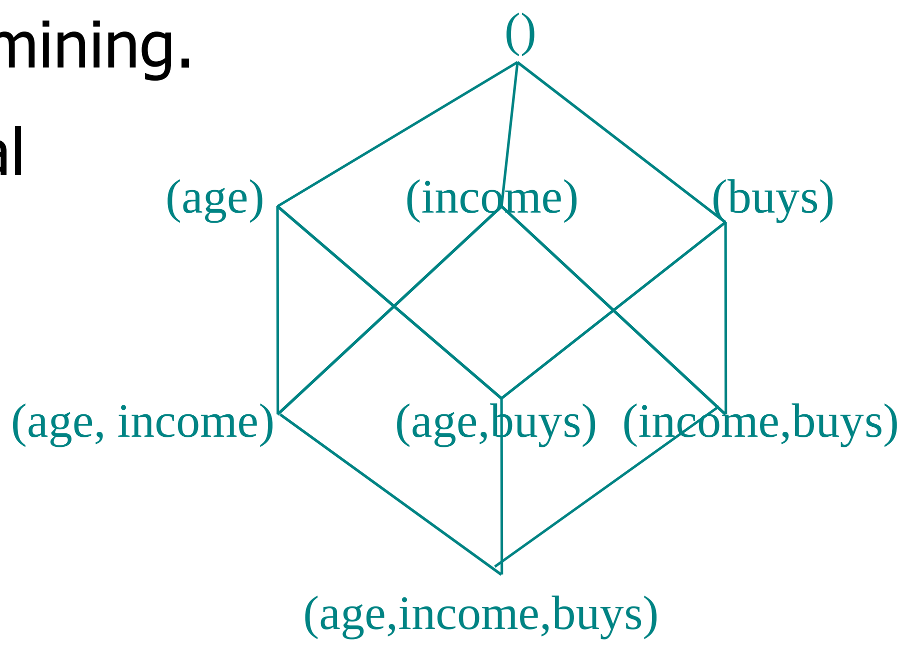
- Techniques can be categorized by how numerical attributes, such as age or salary are treated

1. Static discretization based on predefined concept hierarchies (data cube methods)
2. Dynamic discretization based on data distribution (quantitative rules, e.g., Agrawal & Srikant@SIGMOD96)
3. Clustering: Distance-based association (e.g., Yang & Miller@SIGMOD97)
   - one dimensional clustering then association
4. Deviation: (such as Aumann and Lindell@KDD99)

   Sex = female => Wage: mean=$7/hr (overall mean = $9)

Static Discretization of Quantitative Attributes

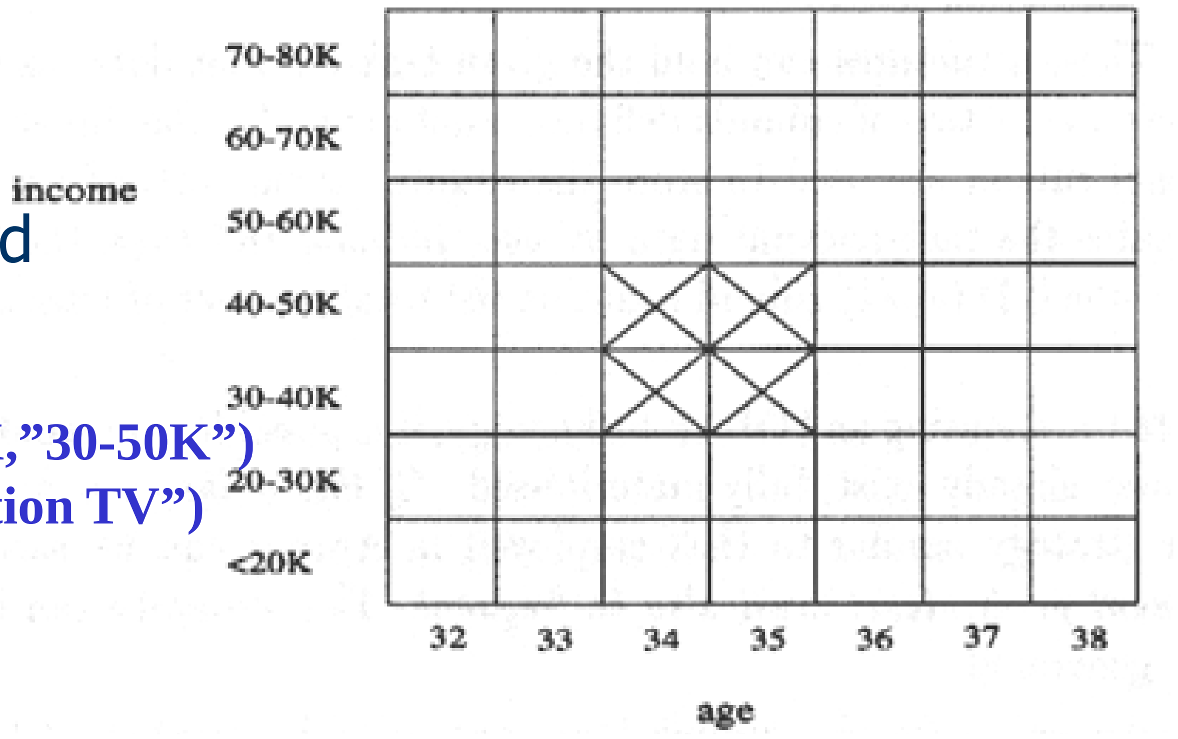
- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges.
- In relational database, finding all frequent k-predicate sets will require *k* or *k*+1 table scans.
- Data cube is well suited for mining.
- The cells of an n-dimensional cuboid correspond to the predicate sets.
- Mining from data cubes can be much faster.



Quantitative Association Rules

- Proposed by Lent, Swami and Widom ICDE'97
- Numeric attributes are *dynamically* discretized
  - Such that the confidence or compactness of the rules mined is maximized
- 2-D quantitative association rules: $A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$
- Cluster *adjacent* association rules to form general rules using a 2-D grid
- Example

age(X,"34-35") ∧ income(X,"30-50K") ⇒ buys(X,"high resolution TV")



Mining Other Interesting Patterns

- Flexible support constraints (Wang et al. @ VLDB'02)
  - Some items (e.g., diamond) may occur rarely but are valuable
  - Customized sup_{min} specification and application
- Top-K closed frequent patterns (Han, et al. @ ICDM'02)
  - Hard to specify sup_{min}, but top-k with $length_{min}$ is more desirable
  - Dynamically raise sup_{min} in FP-tree construction and mining, and select most promising path to mine

Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Interestingness Measure: Correlations (Lift)

- *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading
  - The overall % of students eating cereal is 75% > 66.7%.
- *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is more accurate, although with lower support and confidence
- Measure of dependent/correlated events: lift

$$lift = \frac{P(A \cup B)}{P(A)P(B)}$$

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

$$lift(B,C) = \frac{2000/5000}{3000/5000 * 3750/5000} = 0.89 \qquad lift(B,\neg C) = \frac{1000/5000}{3000/5000 * 1250/5000} = 1.33$$

Are *lift* and χ^2 Good Measures of Correlation?

- *"Buy walnuts ⇒ buy milk* [1%, 80%]" is misleading
  - if 85% of customers buy milk
- Support and confidence are not good to represent correlations
- So many interestingness measures? (Tan, Kumar, Sritastava @KDD'02)

$$lift = \frac{P(A \cup B)}{P(A)P(B)}$$

$$all\_conf = \frac{\sup(X)}{\max\_item\_\sup(X)}$$

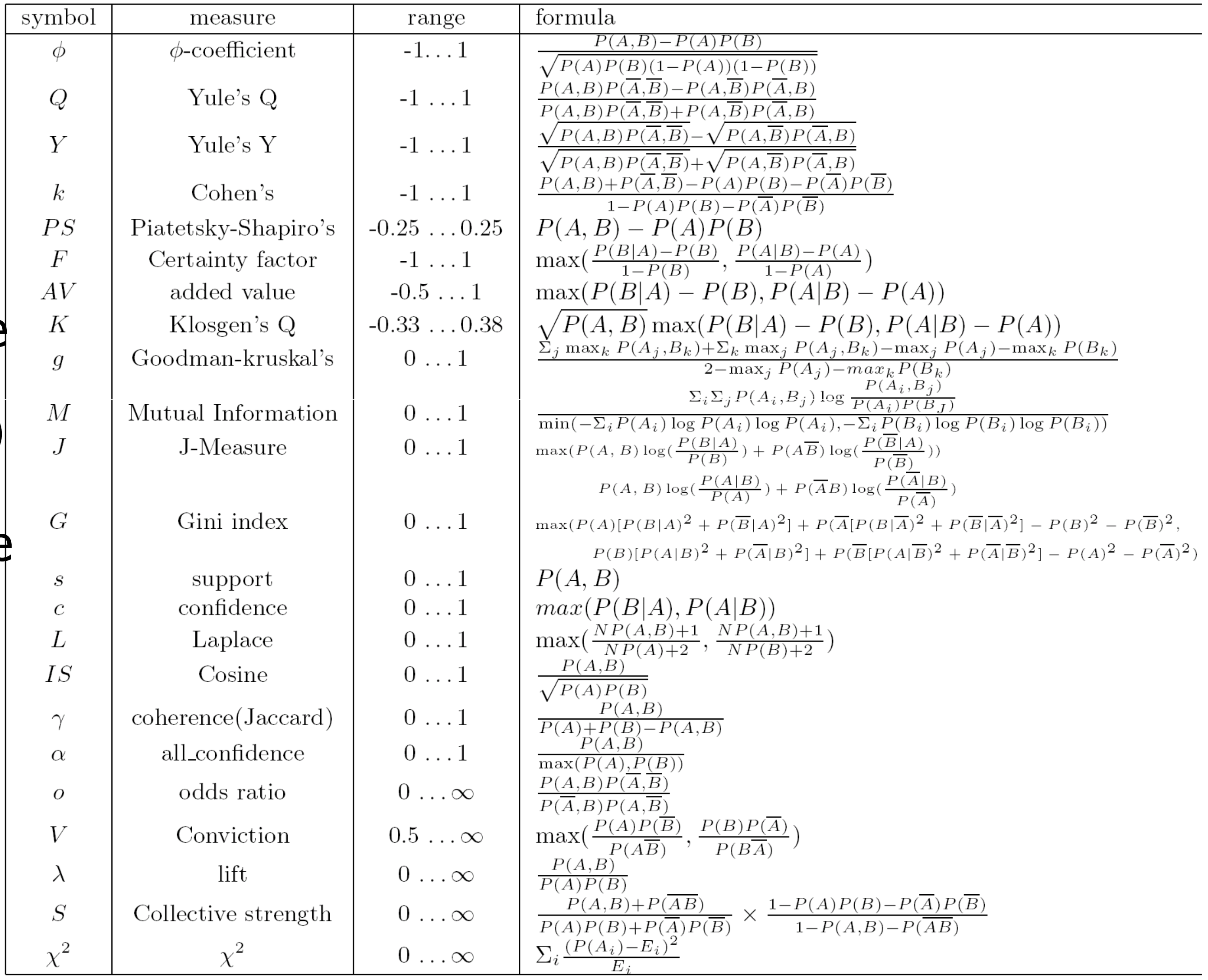
$$\cos ine = \frac{P(A \cup B)}{\sqrt{P(A)P(B)}}$$

	Milk	No Milk	Sum (row)
Coffee	m, c	~m, c	c
No Coffee	m, ~c	~m, ~c	~c
Sum(col.)	m	~m	Σ

DB	m, c	~m, c	m~c	~m~c	lift	all-conf	cosine	χ2
A1	1000	100	100	10,000	9.26	0.91	0.91	9055
A2	100	1000	1000	100,000	8.44	0.09	0.91	670
A3	1000	100	10000	100,000	9.18	0.09	0.91	8172
A4	1000	1000	1000	1000	1	0.5	0.91	0

Which Measures Should Be Used?

- **lift** *and* χ^2 are not good measures for correlations in large transactional DBs
- **all-conf** or **coherence** could be good measures (Omiecinski@TKDE'03)
- Both **all-conf** and **coherence** have the downward closure property



symbol	measure	range	formula
ϕ	ϕ-coefficient	-1 . . . 1	$\frac{P(A,B)-P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
Q	Yule's Q	-1 . . . 1	$\frac{P(A,B)P(\overline{A},\overline{B})-P(A,\overline{B})P(\overline{A},B)}{P(A,B)P(\overline{A},\overline{B})+P(A,\overline{B})P(\overline{A},B)}$
Y	Yule's Y	-1 . . . 1	$\frac{\sqrt{P(A,B)P(\overline{A},\overline{B})}-\sqrt{P(A,\overline{B})P(\overline{A},B)}}{\sqrt{P(A,B)P(\overline{A},\overline{B})}+\sqrt{P(A,\overline{B})P(\overline{A},B)}}$
k	Cohen's	-1 . . . 1	$\frac{P(A,B)+P(\overline{A},\overline{B})-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A)P(B)-P(\overline{A})P(\overline{B})}$
PS	Piatetsky-Shapiro's	-0.25 . . . 0.25	$P(A,B)-P(A)P(B)$
F	Certainty factor	-1 . . . 1	$\max(\frac{P(B\|A)-P(B)}{1-P(B)}, \frac{P(A\|B)-P(A)}{1-P(A)})$
AV	added value	-0.5 . . . 1	$\max(P(B\|A)-P(B), P(A\|B)-P(A))$
K	Klosgen's Q	-0.33 . . . 0.38	$\sqrt{P(A,B)}\max(P(B\|A)-P(B), P(A\|B)-P(A))$
g	Goodman-kruskal's	0 . . . 1	$\frac{\Sigma_j \max_k P(A_j,B_k)+\Sigma_k \max_j P(A_j,B_k)-\max_j P(A_j)-\max_k P(B_k)}{2-\max_j P(A_j)-max_k P(B_k)}$
M	Mutual Information	0 . . . 1	$\frac{\Sigma_i\Sigma_j P(A_i,B_j)\log\frac{P(A_i,B_j)}{P(A_i)P(B_J)}}{\min(-\Sigma_i P(A_i)\log P(A_i)\log P(A_i), -\Sigma_i P(B_i)\log P(B_i)\log P(B_i))}$
J	J-Measure	0 . . . 1	$\max(P(A,B)\log(\frac{P(B\|A)}{P(B)})+P(A\overline{B})\log(\frac{P(\overline{B}\|A)}{P(\overline{B})}))$ $P(A,B)\log(\frac{P(A\|B)}{P(A)})+P(\overline{A}B)\log(\frac{P(\overline{A}\|B)}{P(\overline{A})})$
G	Gini index	0 . . . 1	$\max(P(A)[P(B\|A)^2+P(\overline{B}\|A)^2]+P(\overline{A}[P(B\|\overline{A})^2+P(\overline{B}\|\overline{A})^2]-P(B)^2-P(\overline{B})^2,$ $P(B)[P(A\|B)^2+P(\overline{A}\|B)^2]+P(\overline{B}[P(A\|\overline{B})^2+P(\overline{A}\|\overline{B})^2]-P(A)^2-P(\overline{A})^2)$
s	support	0 . . . 1	$P(A,B)$
c	confidence	0 . . . 1	$max(P(B\|A), P(A\|B))$
L	Laplace	0 . . . 1	$\max(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2})$
IS	Cosine	0 . . . 1	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
γ	coherence(Jaccard)	0 . . . 1	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
α	all_confidence	0 . . . 1	$\frac{P(A,B)}{\max(P(A),P(B))}$
o	odds ratio	0 . . . ∞	$\frac{P(A,B)P(\overline{A},\overline{B})}{P(\overline{A},B)P(A,\overline{B})}$
V	Conviction	0.5 . . . ∞	$\max(\frac{P(A)P(\overline{B})}{P(A\overline{B})}, \frac{P(B)P(\overline{A})}{P(B\overline{A})})$
λ	lift	0 . . . ∞	$\frac{P(A,B)}{P(A)P(B)}$
S	Collective strength	0 . . . ∞	$\frac{P(A,B)+P(\overline{AB})}{P(A)P(B)+P(\overline{A})P(\overline{B})} \times \frac{1-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A,B)-P(\overline{AB})}$
χ^2	χ^2	0 . . . ∞	$\Sigma_i \frac{(P(A_i)-E_i)^2}{E_i}$

Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Constraint-based (Query-Directed) Mining

- Finding all the patterns in a database autonomously? — unrealistic!
  - The patterns could be too many but not focused!
- Data mining should be an interactive process
  - User directs what to be mined using a data mining query language (or a graphical user interface)
- Constraint-based mining
  - User flexibility: provides constraints on what to be mined
  - System optimization: explores such constraints for efficient mining—constraint-based mining

Constraints in Data Mining

- Knowledge type constraint:
  - classification, association, etc.
- Data constraint — using SQL-like queries
  - find product pairs sold together in stores in Chicago in Dec.'02
- Dimension/level constraint
  - in relevance to region, price, brand, customer category
- Rule (or pattern) constraint
  - small sales (price $< \$10$) triggers big sales (sum $> \$200$)
- Interestingness constraint
  - strong rules: min_support $\geq$ 3%, min_confidence $\geq$ 60%

Constrained Mining vs. Constraint-Based Search

- Constrained mining vs. constraint-based search/reasoning
  - Both are aimed at reducing search space
  - Finding all patterns satisfying constraints vs. finding some (or one) answer in constraint-based search in AI
  - Constraint-pushing vs. heuristic search
  - It is an interesting research problem on how to integrate them
- Constrained mining vs. query processing in DBMS
  - Database query processing requires to find all
  - Constrained pattern mining shares a similar philosophy as pushing selections deeply in query processing

Anti-Monotonicity in Constraint Pushing

- Anti-monotonicity
  - *When an intemset S **violates** the constraint, so does any of its superset*
  - *sum(S.Price)* $\leq v$ is anti-monotone
  - *sum(S.Price)* $\geq v$ is not anti-monotone
- Example. C: range(S.profit) $\leq$ 15 is anti-monotone
  - Itemset *ab* violates C
  - So does every superset of *ab*

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Monotonicity for Constraint Pushing

- Monotonicity
  - *When an intemset S **satisfies** the constraint, so does any of its superset*
  - *sum(S.Price)* $\geq$ *v* is monotone
  - *min(S.Price)* $\leq$ *v* is monotone
- Example. C: range(S.profit) $\geq$ 15
  - Itemset *ab* satisfies C
  - So does every superset of *ab*

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Succinctness

- Succinctness:
  - Given A_1, the set of items satisfying a succinctness constraint C, then any set S satisfying C is based on A_1 , i.e., S contains a subset belonging to A_1
  - Idea: Without looking at the transaction database, whether an itemset S satisfies constraint C can be determined based on the selection of items
  - $min(S.Price) \leq v$ is succinct
  - $sum(S.Price) \geq v$ is not succinct
- Optimization: If C is succinct, C is pre-counting pushable

Converting "Tough" Constraints

- Convert tough constraints into anti-monotone or monotone by properly ordering items
- Examine C: avg(*S*.profit) ≥ 25
  - Order items in value-descending order
    - *<a, f, g, d, b, h, c, e>*
  - If an itemset *afb* violates C
    - So does *afbh, afb**
    - It becomes anti-monotone!

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Strongly Convertible Constraints

- avg(X) ≥ 25 is convertible anti-monotone w.r.t. item value descending order R: <*a, f, g, d, b, h, c, e*>
  - If an itemset *af* violates a constraint C, so does every itemset with *af* as prefix, such as *afd*
- avg(X) ≥ 25 is convertible monotone w.r.t. item value ascending order R^{-1}: <*e, c, h, b, d, g, f, a*>
  - If an itemset *d* satisfies a constraint *C*, so does itemsets *df* and *dfa*, which having *d* as a prefix
- Thus, avg(X) ≥ 25 is strongly convertible

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Mining With Convertible Constraints

- C: avg(X) >= 25, min_sup=2
- List items in every transaction in value descending order R: <a, f, g, d, b, h, c, e>
  - C is convertible anti-monotone w.r.t. R
- Scan TDB once
  - remove infrequent items
    - Item h is dropped
  - Itemsets a and f are good, …
- Projection-based mining
  - Imposing an appropriate order on item projection
  - Many tough constraints can be converted into (anti)-monotone

Item	Value
a	40
f	30
g	20
d	10
b	0
h	-10
c	-20
e	-30

TDB (min_sup=2)

TID	Transaction
10	a, f, d, b, c
20	f, g, d, b, c
30	a, f, d, c, e
40	f, g, h, c, e

Handling Multiple Constraints

- Different constraints may require different or even conflicting item-ordering
- If there exists an order *R* s.t. both C_1 and C_2 are convertible w.r.t. *R*, then there is no conflict between the two convertible constraints
- If there exists conflict on order of items
  - Try to satisfy one constraint first
  - Then using the order for the other constraint to mine frequent itemsets in the corresponding projected database

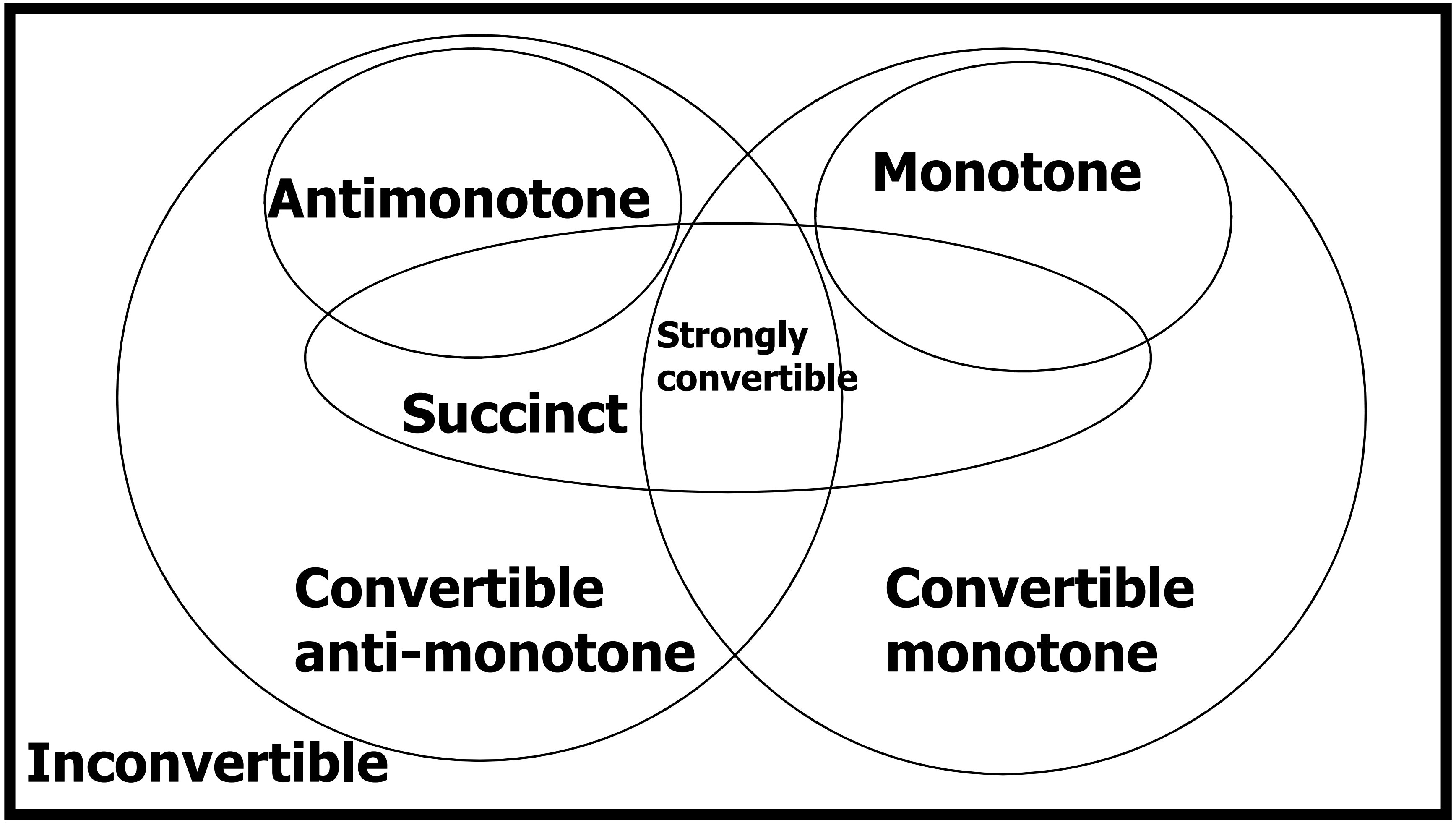
What Constraints Are Convertible?

Constraint	Convertible anti-monotone	Convertible monotone	Strongly convertible
avg(S) $\leq , \geq v$	Yes	Yes	Yes
median(S) $\leq , \geq v$	Yes	Yes	Yes
sum(S) $\leq v$ (items could be of any value, $v \geq 0$)	Yes	No	No
sum(S) $\leq v$ (items could be of any value, $v \leq 0$)	No	Yes	No
sum(S) $\geq v$ (items could be of any value, $v \geq 0$)	No	Yes	No
sum(S) $\geq v$ (items could be of any value, $v \leq 0$)	Yes	No	No
……			

Constraint-Based Mining—A General Picture

Constraint	Antimonotone	Monotone	Succinct
$v \in S$	no	yes	yes
$S \supseteq V$	no	yes	yes
$S \subseteq V$	yes	no	yes
$\min(S) \leq v$	no	yes	yes
$\min(S) \geq v$	yes	no	yes
$\max(S) \leq v$	yes	no	yes
$\max(S) \geq v$	no	yes	yes
$\text{count}(S) \leq v$	yes	no	weakly
$\text{count}(S) \geq v$	no	yes	weakly
$\text{sum}(S) \leq v$ ($a \in S, a \geq 0$)	yes	no	no
$\text{sum}(S) \geq v$ ($a \in S, a \geq 0$)	no	yes	no
$\text{range}(S) \leq v$	yes	no	no
$\text{range}(S) \geq v$	no	yes	no
$\text{avg}(S)\ \theta\ v, \theta \in \{ =, \leq, \geq \}$	convertible	convertible	no
$\text{support}(S) \geq \xi$	yes	no	no
$\text{support}(S) \leq \xi$	no	yes	no

A Classification of Constraints



Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Frequent-Pattern Mining: Summary

- Frequent pattern mining—an important task in data mining
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Projection-based (FPgrowth, CLOSET+, ...)
  - Vertical format approach (CHARM, ...)
- Mining a variety of rules and interesting patterns
- Constraint-based mining
- Mining sequential and structured patterns
- Extensions and applications

Frequent-Pattern Mining: Research Problems

- Mining fault-tolerant frequent, sequential and structured patterns
  - Patterns allows limited faults (insertion, deletion, mutation)
- Mining truly interesting patterns
  - Surprising, novel, concise, …
- Application exploration
  - E.g., DNA sequence analysis and bio-pattern classification
  - "Invisible" data mining