

Introduction to Data Mining

Chapter 4 Association

Fall 2009



Zhi-Hua Zhou

**Department of Computer Science & Technology
Nanjing University**

What is association rule?

Association rule mining searches for interesting relationships among items in a given data set

association rule mining is motivated for *market basket analysis*, where the customer buying habits are analyzed by finding associations between the different items that customers place in their shopping baskets

association rules are in the form of:

Body \rightarrow Head [support, confidence]

example:

$buys(X, \text{"diapers"}) \Rightarrow buys(X, \text{"beers"}) [0.5\%, 60\%]$

$major(X, \text{"CS"}) \wedge takes(X, \text{"DB"}) \Rightarrow grade(X, \text{"A"}) [1\%, 75\%]$

Formal definition of association rule

- $\mathcal{I} = \{i_1, i_2, \dots, i_m\}$ is an itemset, i.e., a set of items
- DB is the task-relevant data, which is a set of transactions where each transaction T is an itemset such that $T \subseteq \mathcal{I}$
- if A is an itemset, we say T contains A iff $A \subseteq T$

an association rule is of the form $A \Rightarrow B$, where $A \subset \mathcal{I}$, $B \subset \mathcal{I}$, and $A \cap B = \emptyset$

the rule holds in DB with

$$\text{support}(A \Rightarrow B) = \mathbf{P}(A \text{ and } B)$$

$$\text{confidence}(A \Rightarrow B) = \mathbf{P}(B / A)$$

- rules satisfying both min_sup and min_conf are *strong*
- an itemset contains k items is a *k-itemset*
- the number of transactions that contain an itemset is the *frequency* (or *support count*) of the itemset
- if an itemset satisfies min_sup , then it is a *frequent itemset* (or *large itemset*)

Categories of association rules

- **Based on the types of values**

- **Boolean association rule**

- $buys(X, \text{"diapers"}) \Rightarrow buys(X, \text{"beers"}) [0.5\%, 60\%]$

- **quantitative association rule**

- $age(X, \text{"30-39"}) \wedge income(X, \text{"42-48K"}) \Rightarrow buys(X, \text{"computer"}) [1\%, 75\%]$

- **Based on the dimensions of data**

- **single-dimensional association rule**

- $buys(X, \text{"diapers"}) \Rightarrow buys(X, \text{"beers"}) [0.5\%, 60\%]$

- **multidimensional association rule**

- $age(X, \text{"30-39"}) \wedge income(X, \text{"42-48K"}) \Rightarrow buys(X, \text{"computer"}) [1\%, 75\%]$

- **Based on the levels of abstractions**

- **single-level association rule**

- $age(X, \text{"30-34"}) \Rightarrow buys(X, \text{"computer"}) [1\%, 75\%]$

- **multilevel association rule**

- $age(X, \text{"30-32"}) \Rightarrow buys(X, \text{"laptop computer"}) [0.5\%, 80\%]$

- $age(X, \text{"30-34"}) \Rightarrow buys(X, \text{"computer"}) [1\%, 75\%]$

Two-step process of association rule mining

- **step1: find all frequent itemsets**
- **step2: generate strong association rules from the frequent itemsets**

step2 is easier, therefore most works on association rule mining focus on step1

a solution for step2:

considering $\text{confidence}(A \Rightarrow B) = P(B|A) = \frac{P(A \text{ and } B)}{P(A)} = \frac{\text{support}(A \Rightarrow B)}{\text{support}(A)}$

- **for each frequent itemset l , generate all non-empty subsets of l**
- **for every non-empty subset s of l , output the rule “ $s \Rightarrow (l - s)$ ” if**

$$\frac{\text{support}(l)}{\text{support}(s)} \geq \text{min_conf}$$



Apriori (I)

- in initial is for finding frequent itemsets for Boolean association rules
[Agrawal & Srikant, VLDB94]
- level-wise search, where k -itemsets are used to explore $(k+1)$ -itemsets

the key of Apriori is the *a priori* knowledge:

all non-empty subsets of a frequent itemset must also be frequent

from L_k to L_{k+1} :

- **join**: a set of candidate frequent $(k+1)$ -itemsets, C_{k+1} , is generated by joining L_k with itself (the set of frequent k -itemsets is denoted by L_k)

$$L_k \triangleright \triangleleft L_k = \{A \triangleright \triangleleft B \mid A, B \in L_k, |A \cap B| = k - 1\}$$

- **prune**: a scan of the database for determining the count of each candidate in C_{k+1} would result in the determination of L_{k+1}

a priori knowledge is used to reduce the size of C_{k+1}

Apriori (II)

an example

Suppose $\text{min_sup_count} = 2$

Database DB

TID	Items
T100	I1, I2, I5
T200	I2, I3, I4
T300	I3, I4
T400	I1, I2, I3, I4

C_1

itemset	sup
{I1}	2
{I2}	3
{I3}	3
{I4}	3
{I5}	1

Scan DB

L_1

itemset	sup
{I1}	2
{I2}	3
{I3}	3
{I4}	3

C_2

itemset	sup
{I1, I2}	2
{I1, I3}	1
{I1, I4}	1
{I2, I3}	2
{I2, I4}	2
{I3, I4}	3

Scan DB

C_2

itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I2, I3}
{I2, I4}
{I3, I4}

L_2

itemset	sup
{I1, I2}	2
{I2, I3}	2
{I2, I4}	2
{I3, I4}	3

C_3

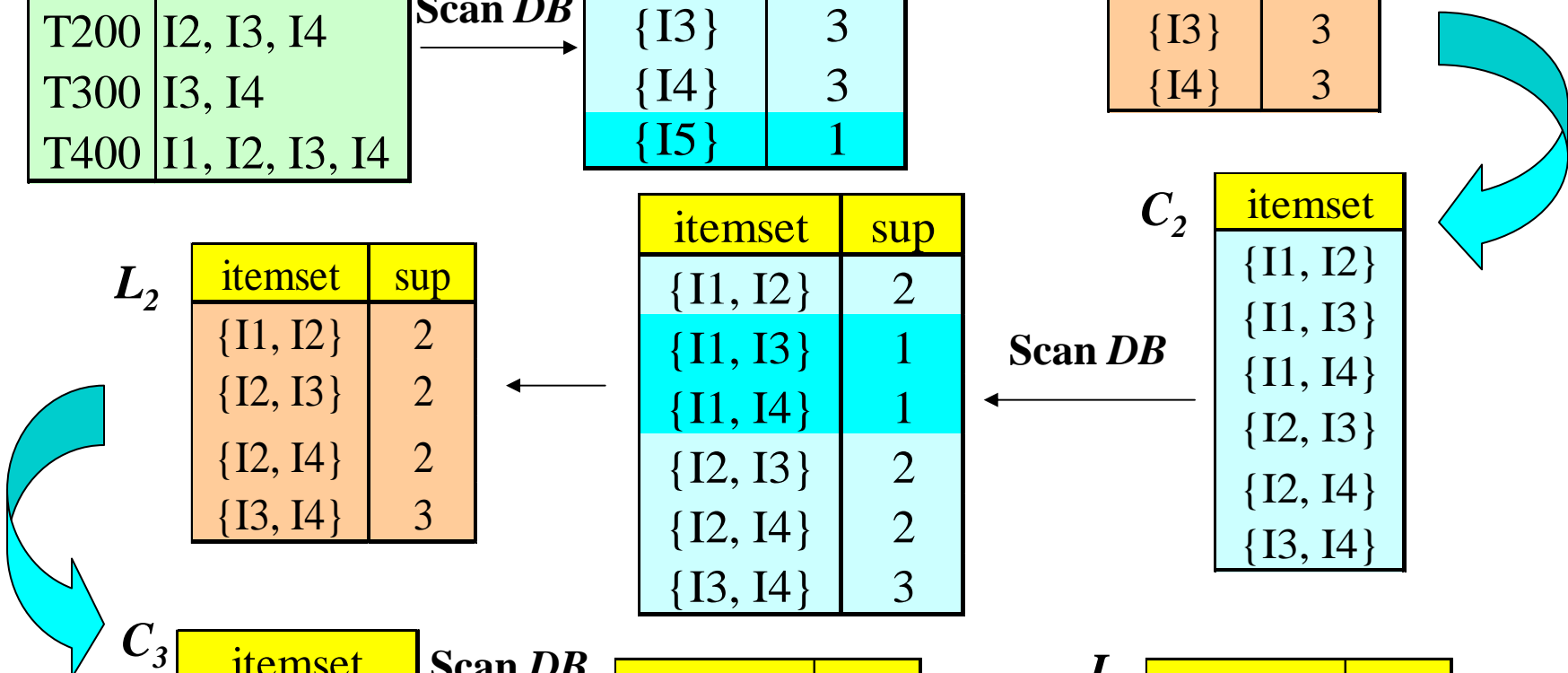
itemset
{I2, I3, I4}

Scan DB

itemset	sup
{I2, I3, I4}	2

L_3

itemset	sup
{I2, I3, I4}	2



Apriori (III)

Input: DB : database of transactions; min_sup : minimum support threshold

Output: L : frequent itemsets in DB

Method:

$L_1 = \text{find_frequent_1-itemsets}(DB);$

for ($k = 1; L_k \neq \emptyset; k++$) {

$C_{k+1} = \text{apriori_gen}(L_k, min_sup);$ // generate candidate frequent ($k+1$)-itemsets

for each transaction $t \in DB$ { // scan DB for counting

$C_t = \text{subset}(C_{k+1}, t);$ // get the subsets of t that are candidates

for each candidate $c \in C_t$

$c.count ++$

}

$L_{k+1} = \{c \in C_{k+1} \mid c.count \geq min_sup\}$

}

return $L = \cup_k L_k;$

Apriori (IV)

generate candidate frequent $(k+1)$ -itemset

```
procedure apriori_gen( $L_k$ : frequent  $k$ -itemsets;  $min\_sup$ : minimum support)
  for each itemset  $l_1 \in L_k$ 
    for each itemset  $l_2 \in L_k$ 
      if  $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \dots \wedge (l_1[k-1] = l_2[k-1]) \wedge (l_1[k] < l_2[k])$  then {
         $c = l_1 \triangleright \triangleleft l_2$ ; // join step: generate candidates why?
        if has_infrequent_subset( $c, L_k$ ) then
          delete  $c$ ; // prune step: remove unfruitful candidate
        else add  $c$  to  $C_{k+1}$ ;
      }
  return  $C_{k+1}$ ;
```

use knowledge to prune C_{k+1}

```
procedure has_infrequent_subset( $c$ : candidate  $(k+1)$ -itemset;  $L_k$ : frequent  $k$ -itemsets)
  for each  $k$ -subset  $s$  of  $c$ 
    if  $s \notin L_k$  then
      return TRUE;
  return FALSE;
```

Apriori (v)

How about the joinable L_k are not ordered and constrained?

for example, without those constraints, {I1, I3, I5} and {I1, I4, I5} will be joined to {I1, I3, I4, I5}

SETM algorithm do so [Houtsma & Swami, Research Report at IBM Almaden Research Center 1993]

however, in Apriori, if {I1, I3, I4, I5} is a frequent itemset, it will be generated from {I1, I3, I4} and {I1, I3, I5}. Otherwise it will not be generated

therefore those constraints are useful in generating a relatively small set of candidate frequent itemsets

moreover, those constraints can avoid the cost of generating the same candidate frequent itemsets for several times

AIS

the first algorithm for association rule mining

in initial, AIS algorithm did not have a name [Agrawal et al., SIGMOD93]

later, it was named after the abbreviations of the names of the authors, i.e. R. Agrawal, T. Imielinski, and A. Swami

Main differences:

AIS: after obtaining frequent k -itemsets, for each tuple in the database, find all the frequent k -itemsets contained in the tuple, and then expand those k -itemsets to candidate frequent $(k+1)$ -itemsets by adding other items contained in the tuple to those k -itemsets

Apriori: generate candidate frequent $(k+1)$ -itemsets directly from frequent k -itemsets. The tuples in the database are only used for counting

AprioriTid

an variant of Apriori [Agrawal & Srikant, VLDB94]



Main differences to Apriori:

AprioriTid **does not count the support of itemsets from the tuples of the database**, instead, a specific data structure \overline{C}_k is maintained, whose content is the frequent k -itemsets contained by every tuple

therefore, as k increases, the tuples and the frequent k -itemsets contained by each tuple in \overline{C}_k decrease, so that the contents required by scanning for counting is reduced

DHP (I)

an variant of Apriori [Park et al., SIGMOD95]

DHP is the abbreviation of *Direct Hashing and Pruning*

- based on the observation that **the processes in the initial iterations of Apriori dominates the total execution cost**
- the general idea of DHP is to reduce the number of transactions to be scanned and trim the number of items in each transaction for the initial candidate set generation, especially for the frequent 2-itemsets
- the key of DHP is a set of four *a priori* knowledge



DHP (II)

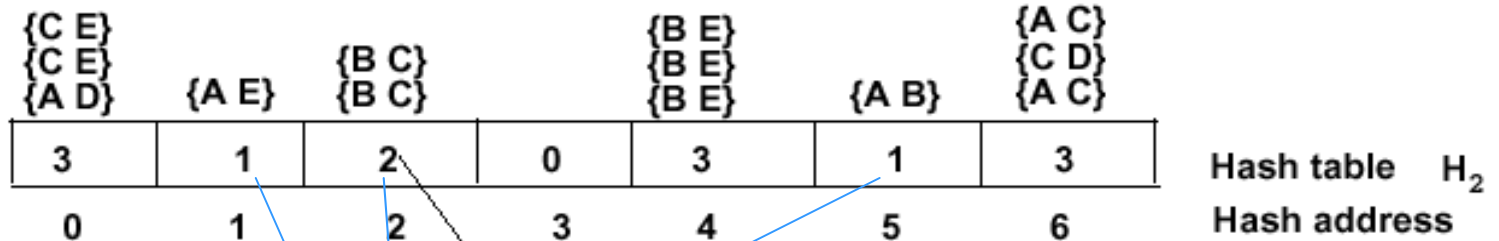
Knowledge 1: any member of the candidate frequent itemset must be hashed to a bucket whose count is not smaller than the min support count

useful for reducing the size of the set of candidate frequent itemset

Making a hash table

100 {A C}, {A D}, {C D}
 200 {B C}, {B E}, {C E}
 300 {A B}, {A C}, {A E}, {B C}, {B E}, {C E}
 400 {B E}

$$h\{\{x y\}\} = ((\text{order of } x) * 10 + (\text{order of } y)) \bmod 7;$$



The number of items hashed to bucket 2

suppose the min support count is 3, then itemsets hashed to those buckets will not be considered in further processing

DHP (III)

Knowledge 2: any tuple useful in determining frequent $(k+1)$ -itemsets must contain at least $(k+1)$ candidate frequent k -itemsets

useful for reducing the number of the tuples for scanning

Knowledge 3: for any items contained in a tuple, if it is useful in determining frequent $(k+1)$ -itemsets, it must appear in at least k candidate frequent k -itemsets

useful for reducing the size of the tuples for scanning

Knowledge 4: for any items contained in a tuple, if it is useful in determining frequent $(k+1)$ -itemsets, it must appear in at least one $(k+1)$ -itemset whose k -itemsets are all candidate frequent k -itemsets

useful for reducing the size of the tuples for scanning

All these rules are used for making preparation for C_{k+1} when the database is scanned to determine L_k from C_k

Techniques for improving Apriori

- **Hash-based itemset counting**

a k -itemset whose corresponding hashing bucket count is below the threshold cannot be frequent

- **Transaction reduction**

a transaction that does not contain any frequent k -itemset is useless in subsequent scans

- **Partitioning**

any itemset that is potentially frequent in a database must be frequent in at least one of the partitions of the database

- **Sampling**

mining a sampled subset of database with a lower support threshold, and using some mechanism to ensure the completeness

- **Dynamic itemset counting**

adding new candidate itemsets when all of their subsets are estimated to be frequent



FP-growth (I)

In situations with a large number of frequent patterns, long patterns, or quite low min support thresholds, an Apriori-style algorithm may suffer from:

- *it is costly to handle a huge number of candidates*
- *it is tedious to repeatedly scan the database and check a large set of candidates by pattern matching*

Can we avoid candidate generation-and-test?

- **mining frequent patterns without candidate generation**
[Han et al., SIGMOD2000; Han et al., DMKD04]
- **utilizing the FP-tree structure**
- **two steps: constructing FP-tree; mining frequent patterns using FP-tree**

FP-growth (II)

Suppose $\text{min_sup_count} = 3$

Database *DB*

TID	Items
T100	<i>f, a, c, d, g, i, m, p</i>
T200	<i>a, b, c, f, l, m, o</i>
T300	<i>b, f, h, j, o, w</i>
T400	<i>b, c, k, s, p</i>
T500	<i>a, f, c, e, l, p, m, n</i>

Scan
DB

F-list	
	$\langle (f:4), (c:4), (a:3), (b:3), (m:3), (p:3) \rangle$

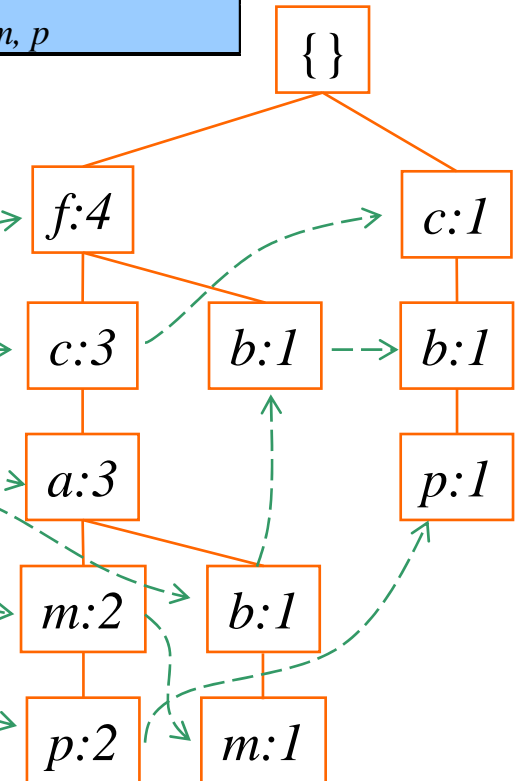
Scan
DB

TID	(Ordered) frequent items
T100	<i>f, c, a, m, p</i>
T200	<i>f, c, a, b, m</i>
T300	<i>f, b</i>
T400	<i>c, b, p</i>
T500	<i>f, c, a, m, p</i>

FP-tree construction:

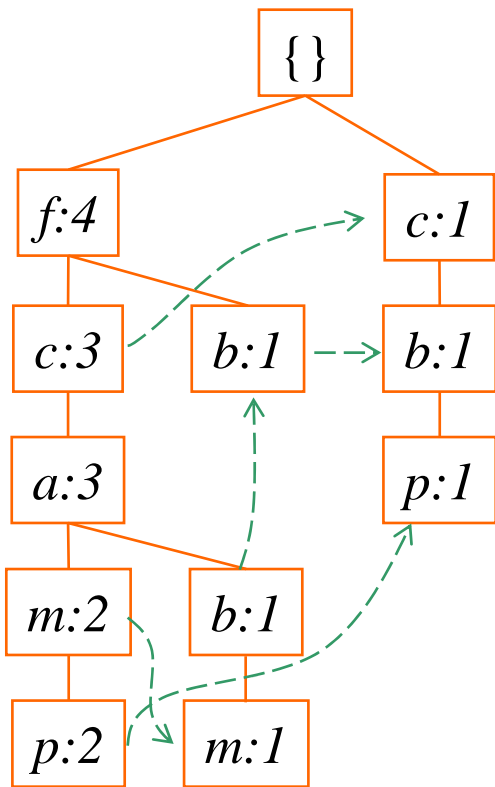
- 1) Scan *DB* once, find frequent 1-itemset
- 2) Sort frequent items in frequency descending order, F-list
- 3) Scan *DB* again, construct FP-tree

Header Table		
item	frequency	head
<i>f</i>	4	
<i>c</i>	4	
<i>a</i>	3	
<i>b</i>	3	
<i>m</i>	3	
<i>p</i>	3	



FP-growth (III)

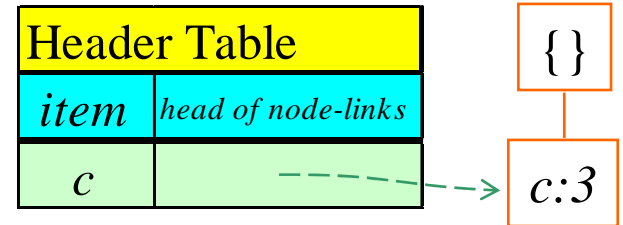
For each frequent item, construct its *conditional pattern-base* and *conditional FP-tree*



Conditional pattern-base of p

$(fcam:2), (cb:1)$

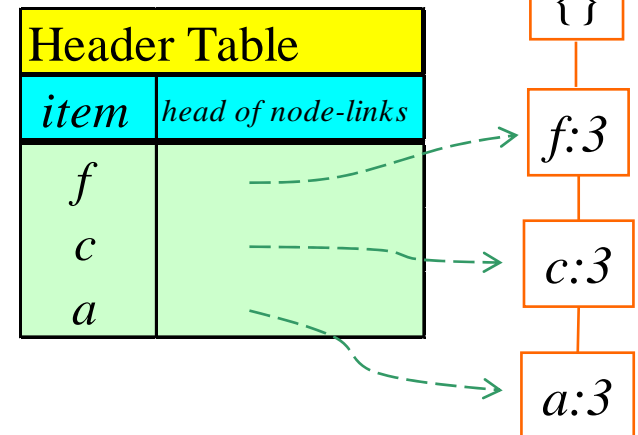
Conditional FP-tree of p



Conditional pattern-base of m

$(fca:2), (fcab:1)$

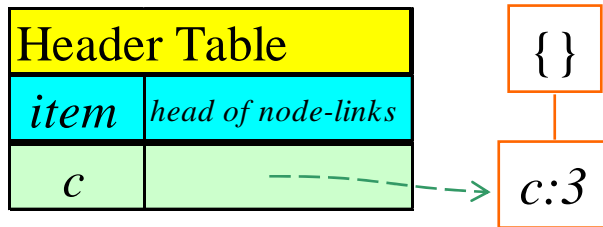
Conditional FP-tree of m



FP-growth (IV)

Generate frequent patterns from *conditional FP-tree*

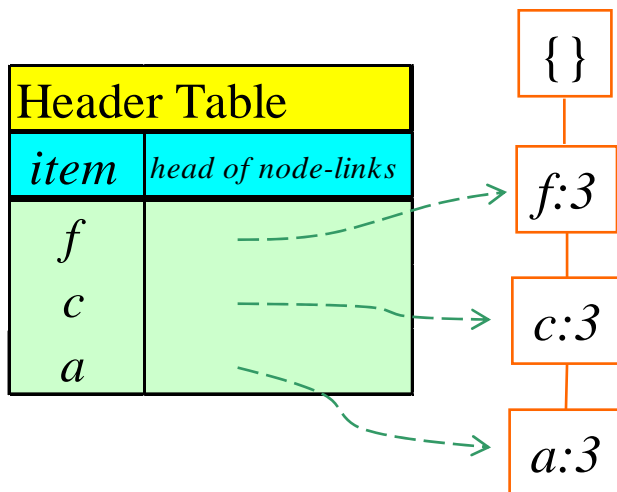
Conditional FP-tree of p



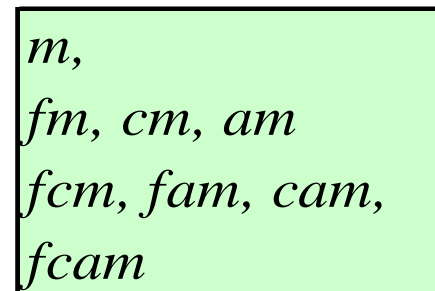
Frequent patterns involving p



Conditional FP-tree of m



Frequent patterns involving m



FP-growth (v)

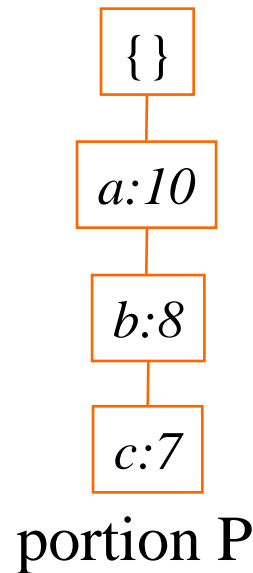
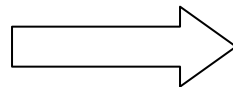
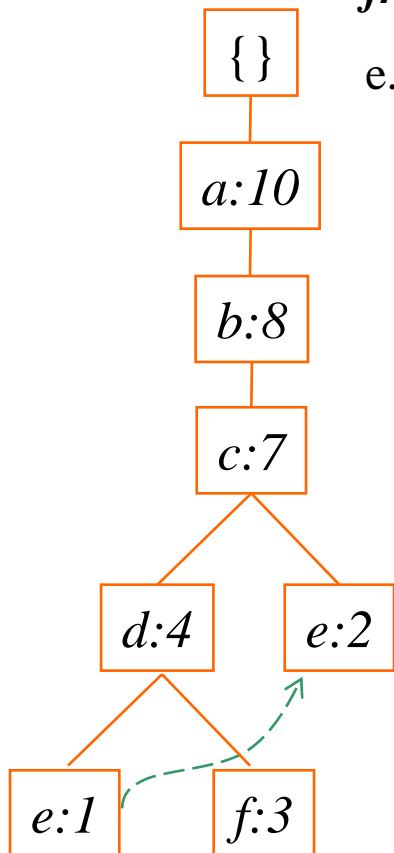
General method for generating frequent patterns from *FP-tree*

$$\text{freq_pattern_set}(T) = \text{freq_pattern_set}(P) \times \text{freq_pattern_set}(Q)$$

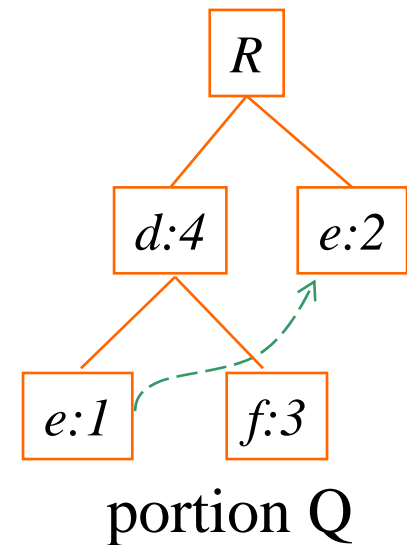
e.g. $\{(a:10)\} \times \text{freq_pattern_set}(Q) = \{(ad:4), (ae:3), (af:3), (adf:3)\}$

$$\text{freq_pattern_set}(P) = \{(a:10), (b:8), (c:7), (ab:8), (ac:7), (bc:7), (abc:7)\}$$

$$\text{freq_pattern_set}(Q) = \{(d:4), (e:3), (f:3), (df:3)\}$$



AND



FP-growth (VI)

Summary of the algorithm:

- Scan the DB twice to construct the FP-tree. All the following processes are performed on the FP-tree (if the tree cannot be held in the main memory, techniques such as *partition projection* can be used)
- 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. In the latter case, generate all the combinations of the sub-paths, each of which is a frequent pattern

Such an idea can be applied to the mining of sequential patterns, episodes, etc. (either Apriori or FP-growth can be used as the basis for the mining of many other types of patterns)

FP-growth (VII)

Why FP-growth is efficient?

- The mining process works on a set of pattern-bases and conditional FP-trees that are usually much smaller than DB
- The mining operations consist of mainly prefix count adjustment, counting local frequent items, and pattern fragment concatenation, which is much less costly than generation and test of a very large number of candidate patterns

FP-growth is about an order of magnitude faster than Apriori, especially when the data set is dense (containing many patterns) and/or when the frequent patterns are long

Closed/maximal frequent itemset (I)

The number of frequent itemsets is usually very big. For example, a frequent itemset of length 100, such as $\{a_1, a_2, \dots, a_{100}\}$, contains $C_{100}^1 + C_{100}^2 + \dots + C_{100}^{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}$ number of frequent itemsets

In fact, usually we do not need to generate all the frequent itemsets. Generating **closed frequent itemsets** or **maximal frequent itemset** is enough in many cases

- An itemset X is a **closed frequent itemset** in S if X is both *closed* and *frequent* in S
 - ✓ An itemset X is **closed** in a data set S if there exists no proper super-itemset Y such that Y has the same support count as X in S
- An itemset X is a **maximal frequent itemset** in S if X is frequent, and there exists no proper super-itemset Y such that Y is frequent in S

Closed/maximal frequent itemset (II)

An example: Suppose a database contains only two transactions: $\{\langle a_1, a_2, \dots, a_{100} \rangle, \langle a_1, a_2, \dots, a_{50} \rangle\}$, and let $min_sup = 1$. Then,

- The set of closed frequent itemsets $\mathbf{C} = \{\{a_1, a_2, \dots, a_{100}\}: 1, \{a_1, a_2, \dots, a_{50}\}: 2\}$
- The maximal frequent itemset $\mathbf{M} = \{\{a_1, a_2, \dots, a_{100}\}: 1\}$

➤ **The set of closed frequent itemsets contains complete information regarding frequent itemsets**

All frequent itemsets and their actual support counts

➤ **The maximal frequent itemset contains all frequent itemsets, but no actual support counts**

Multilevel association rule

Rules generated from association rule mining with concept hierarchies are called multilevel association rules



strong associations discovered at very high concept levels may represent common sense knowledge

however, common sense knowledge to one person may be novel to another person

Mining multilevel association rules

A top-down, progressive deepening approach:

- **Find high-level strong rules**

e.g. $buys(X, \text{"computer"}) \Rightarrow buys(X, \text{"printer"})$ [20%, 60%]

- **Then find their lower-level “weaker” rules**

e.g. $buys(X, \text{"IBM computer"}) \Rightarrow buys(X, \text{"HP printer"})$ [6%, 50%]

Choices:

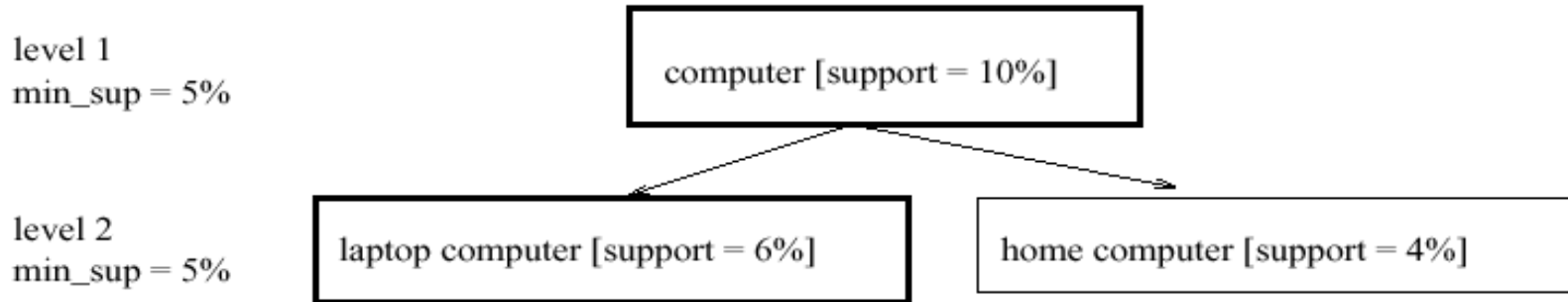
- **uniform support**
- **reduced support**



Mining multilevel association rules with uniform support

use a same `min_sup` for all levels

an example



Disadvantage: it is unlikely that items at lower levels of abstraction will occur as frequently as those at higher levels of abstraction

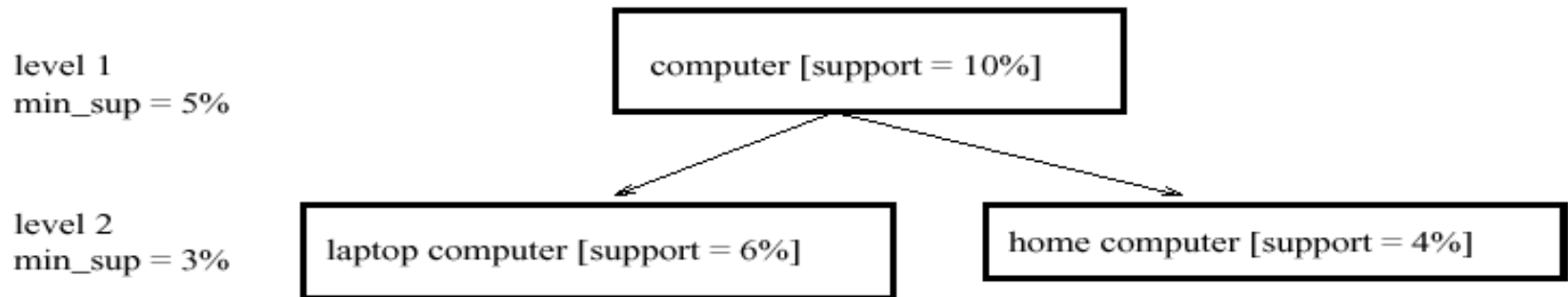
- if the `min_sup` threshold is too high, interesting low level associations will be missed
- if the `min_sup` threshold is too low, uninteresting high level associations will be generated

Mining multilevel association rules with reduced support (I)

- **Level-by-level independent**

each node is examined, regardless of whether or not its parent node is found to be frequent

an example



Advantage: won't miss any interesting associations

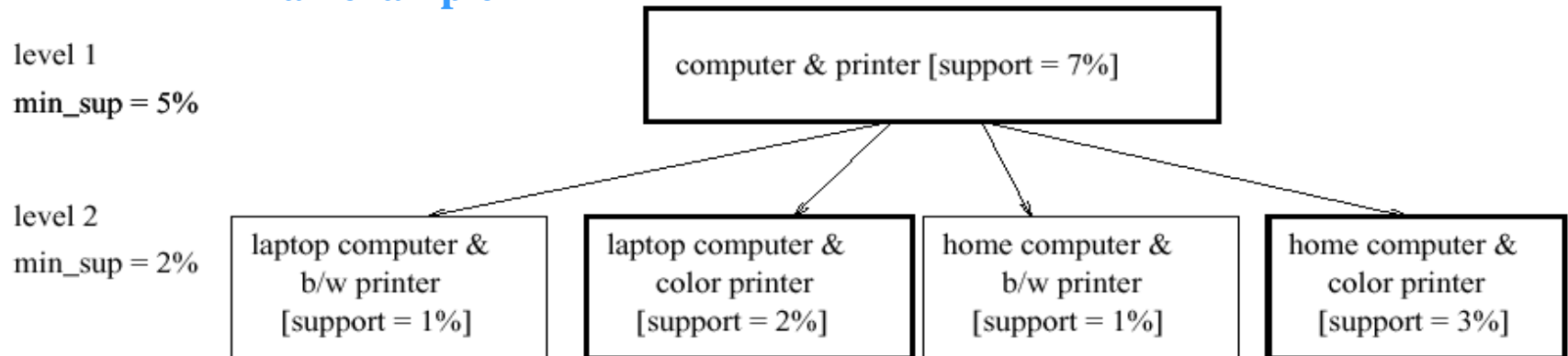
Disadvantage: examine numerous infrequent items at low levels, find associations between items of little importance

Mining multilevel association rules with reduced support (II)

- **Level-cross filtering by k -itemset**

k -itemset at the i -th level is examined if and only if its corresponding parent k -itemset at the $(i-1)$ -th level is frequent

an example



Advantage: examine only the children of frequent k -itemsets

Disadvantage: many valuable patterns may be filtered out

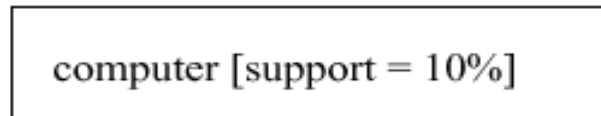
Mining multilevel association rules with reduced support (III)

- **Level-cross filtering by single item**

a node at the i -th level is examined if and only if its parent node at the $(i-1)$ -th level is frequent

an example

level 1
min_sup = 12%



level 2
min_sup = 3%

laptop (not examined)

home computer (not examined)

Advantage: compromise between *level-by-level independent* and *level-cross filtering by k -itemset*

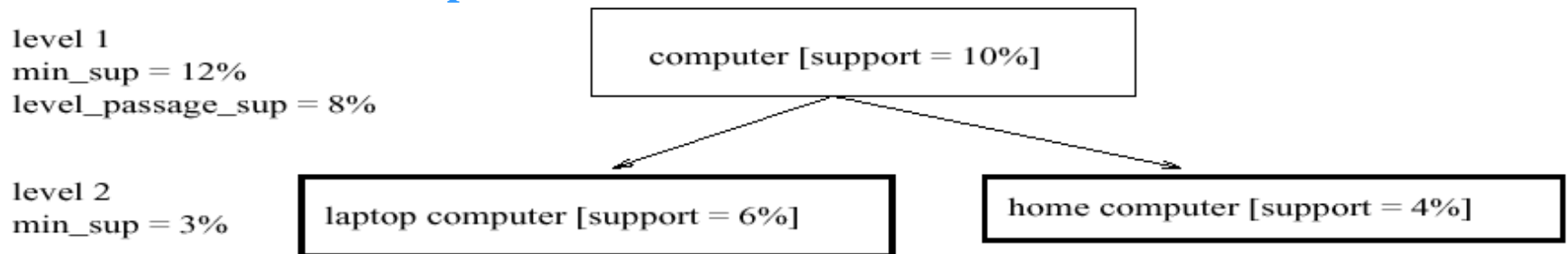
Disadvantage: may miss associations between low level items that are frequent based on a reduced min_sup, but whose parent node are not frequent

Mining multilevel association rules with reduced support (IV)

- **Controlled level-cross filtering by single item**

a variant of *level-cross filtering by single item*. Each level has a *level passage threshold* which is used for passing down sub-frequent items to lower levels

an example



Advantage: allow the children of items that do not satisfy min_sup to be examined if these items satisfy the *level passage threshold*

Disadvantage: the setting of the *level passage threshold* is tricky

Cross-level association rule

Rules whose items do not belong to a same concept level are called cross-level association rules

e.g. *buys*(*X*, “computer”) \Rightarrow *buys*(*X*, “HP printer”) [4%, 70%]



Mining cross-level association rule:

If mining associations from concept levels *i* and *j*, where level *j* is more specific than *i*, then the reduced minimum support threshold of level *j* should be used overall

Redundancy filtering

when multilevel association rules are mined, some of the rules found may be redundant due to “ancestor” relationships between items

a rule $R1$ is an *ancestor* of a rule $R2$ if $R1$ can be obtained by replacing the items in $R2$ by their ancestors in a concept hierarchy

a rule can be considered redundant if its support and confidence are close to their “expected” values based on its ancestor

example:

$buys(X, \text{“computer”}) \Rightarrow buys(X, \text{“HP printer”})$ [8%, 70%]

$buys(X, \text{“IBM computer”}) \Rightarrow buys(X, \text{“HP printer”})$ [4%, 72%]

if the sales of IBM computers is half of that of computers, then the second rule is redundant because it does not offer any additional information

Multidimensional association rule

- **Single-dimensional association rule**

also called *intra-dimensional association rule*

an association rule contains a single predicate with multiple occurrences

e.g. $buys(X, \text{"IBM computer"}) \Rightarrow buys(X, \text{"printer"})$

- **Multidimensional association rule**

an association rule contains two or more predicates or dimensions

- **inter-dimension association rule**

multidimensional association rule without repeated predicates

e.g. $age(X, \text{"19-24"}) \wedge occupation(X, \text{"student"}) \Rightarrow buys(X, \text{"computer"})$

- **hybrid-dimension association rule**

multidimensional association rule with repeated predicates

e.g. $age(X, \text{"19-24"}) \wedge buys(X, \text{"computer"}) \Rightarrow buys(X, \text{"printer"})$

Mining multidimensional association rule

Multidimensional association rule mining searches for frequent predicatesets instead of frequent itemsets

***k*-predicateset is a set containing *k* conjunctive predicates**

multidimensional association rule mining techniques can be categorized by how quantitative attributes are treated:

- **using static discretization of quantitative attributes**
quantitative attributes are statically discretized using predefined concept hierarchies
- **quantitative association rules**
quantitative attributes are dynamically discretized into “bins” based on the distribution of the data, which may be further combined
- **distance-based association rules**
quantitative attributes are clustered to capture the semantic meaning of data

Static discretization of quantitative Attributes

Discretization prior to mining

numeric values are replaced by intervals

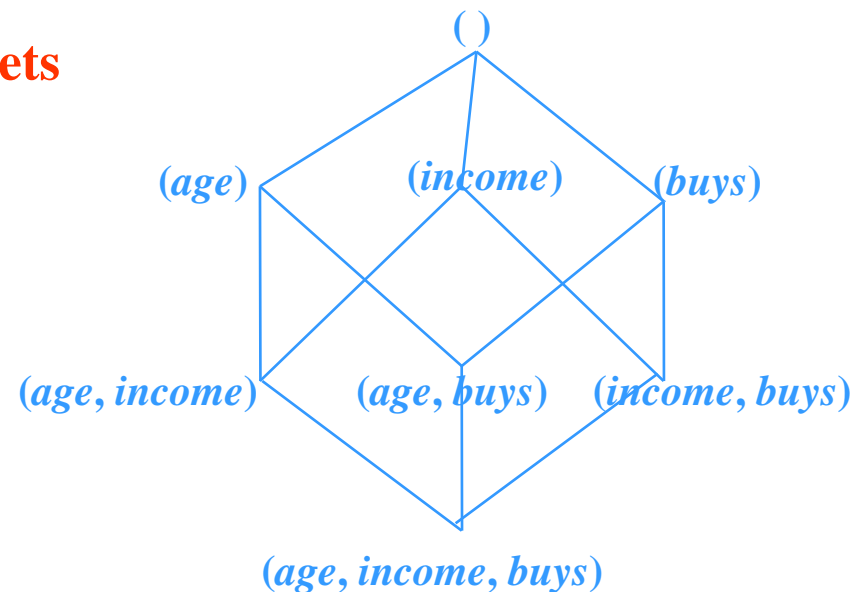
in relational database, finding all frequent k -predicatesets will require k or $(k+1)$ scans

every subset of frequent predicatesets must also be frequent

data cube is well suited for mining

the cells of an n -dimensional cuboid correspond to the predicatesets

mining from data cubes can be more faster



Quantitative association rule (I)

Quantitative attributes are dynamically discretized during the mining process so as to satisfy some mining criteria

a typical procedure (used in ARCS)

map pairs of quantitative attributes onto a 2-D grid for tuples satisfying a given categorical attribute condition. Then search the grid for clusters of points, from which the association rules are generated

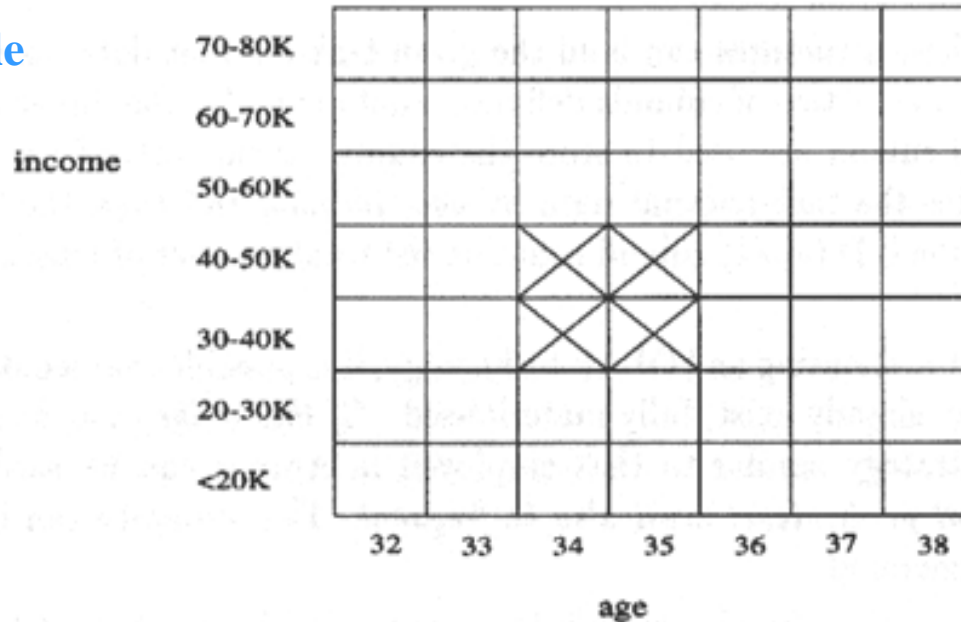
step1: binning partition the ranges of quantitative attributes into intervals that may be combined during the mining process

step2: finding frequent predicatesets scan the grid to find frequent predicatesets that satisfy min_conf, then generate association rules from those predicatesets

step3: clustering association rules map the strong association rules to the grid, and cluster the rules based on proximity

Quantitative association rule (II)

an example

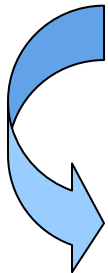


$age(X, "34") \wedge income(X, "30-40K") \Rightarrow buys(X, "TV")$

$age(X, "35") \wedge income(X, "30-40K") \Rightarrow buys(X, "TV")$

$age(X, "34") \wedge income(X, "40-50K") \Rightarrow buys(X, "TV")$

$age(X, "35") \wedge income(X, "40-50K") \Rightarrow buys(X, "TV")$



$age(X, "34-35") \wedge income(X, "30-50K") \Rightarrow buys(X, "TV")$

Distance-based association rule

binning methods do not work well in some cases

an example

Price(\$)	Equi-width (width \$10)	Equi-depth (depth 2)	Distance- based
7	[0,10]	[7,20]	[7,7]
20	[11,20]	[22,50]	[20,22]
22	[21,30]	[51,53]	[50,53]
50	[31,40]		
51	[41,50]		
53	[51,60]		

distance-based discretization is more meaningful because it considers

- **the density or number of points in an interval**
- **the “closeness” of points in an interval**

Mining distance-based association rules

step1: employ clustering techniques to find the intervals or clusters

step2: search for groups of clusters that occur frequently together

support-confidence framework are not used here:

support $\xrightarrow{\text{replaced by}}$ **density threshold**

confidence $\xrightarrow{\text{replaced by}}$ **degree of association**

Criticism to support-confidence framework

Strong association rules may not be interesting

an example:

10,000 transactions, among which 6,000 include *computer games*, 7,500 include *videos*, and 4,000 include both *computer games* and *videos*

suppose $\text{min_sup}=30\%$ and $\text{min_conf}=60\%$, then following association rule will be generated:

$\text{buys}(X, \text{"computer games"}) \Rightarrow \text{buys}(X, \text{"videos"}) [40\%, 66\%]$

however, the probability of purchasing *videos* is 75%, which is even larger than 66%. Therefore the above rule is misleading

In fact, for *computer games* and *videos*, the purchase of one item will decrease the likelihood of purchasing the other one

Correlation analysis

Correlation analysis can be used in analyzing the correlation of itemsets

recall that correlation analysis has been used in handling redundancy in data integration. Now it is used in an alternative of support-confidence framework

the *lift* of the association rule $A \Rightarrow B$ is defined as

$$\text{lift}(A \Rightarrow B) = \frac{P(A \text{ and } B)}{P(A)P(B)} = \frac{P(B|A)}{P(B)}$$

- $\text{lift} = 1$ A and B are independent (under some assumption)
- $\text{lift} > 1$ the occurrence of A encourages the occurrence of B
- $\text{lift} < 1$ the occurrence of A discourages the occurrence of B

the bigger the value of *lift*, the stronger the correlation between A and B

a correlation rule is of the form $\{e_1, e_2, \dots, e_m\}$ where the occurrences of the items $\{e_1, e_2, \dots, e_m\}$ are correlated

Metarule-guided association mining

Metarules allow users to specify the syntactic form of rules that they are interested in mining

example:

a metarule $P_1(X, Y) \wedge P_2(X, W) \Rightarrow buys(X, \text{“educational software”})$

can be matched by

$age(X, \text{“35-45”}) \wedge income(X, \text{“40-60K”}) \Rightarrow buys(X, \text{“educational software”})$



Now let's go to
Chapter 5

