SCS5623 - DATA MINING AND WAREHOUSING

UNIT 2

CONCEPT DESCRIPTION AND ASSOCIATION RULES

Attribute Oriented Induction

- Data focusing: task-relevant data, including dimensions, and the result is the *initial relation*
- Attribute-removal: remove attribute *A* if there is a large set of distinct values for *A* but (1) there is no generalization operator on *A*, or (2) *A*'s higher level concepts are expressed in terms of other attributes
- Attribute-generalization: If there is a large set of distinct values for *A*, and there exists a set of generalization operators on *A*, then select an operator and generalize *A*
- Attribute-threshold control: typical 2-8, specified/default
- Generalized relation threshold control: control the final relation/rule size

How it is done

- Collect the task-relevant data (*initial relation*) using a relational database query
- Perform generalization by attribute removal or attribute generalization
- Apply aggregation by merging identical, generalized tuples and accumulating their respective counts
- Interaction with users for knowledge presentation

Example: Describe general characteristics of graduate students in the University database

Step 1. Fetch relevant set of data using an SQL statement, e.g.,

- **Select** * (i.e., name, gender, major, birth_place, birth_date, residence, phone#, gpa)
- **from** student
- **where** student_status in {"Msc", "MBA", "PhD" }

Step 2. Perform attribute-oriented induction

Step 3. Present results in generalized relation, cross-tab, or rule forms

Basic Algorithm for Attribute-Oriented Induction

- InitialRel: Query processing of task-relevant data, deriving the *initial relation*.
- PreGen: Based on the analysis of the number of distinct values in each attribute, determine generalization plan for each attribute: removal? or how high to generalize?
- PrimeGen: Based on the PreGen plan, perform generalization to the right level to derive a "prime generalized relation", accumulating the counts.
- Presentation: User interaction: (1) adjust levels by drilling, (2) pivoting, (3) mapping into rules, cross tabs, visualization presentations.

Class Characterization: An Example

Analytical Characterization

Initial Relation

Name	Gender	Major	Birth-Place	Birth_date	Residence	Phone #	GPA
Jim Woodman	M	CS	Vancouver,BC, Canada	8-12-76	3511 Main St., Richmond	687-4598	3.67
Scott Lachance	M	CS	Montreal, Que, Canada	28-7-75	345 1st Ave., Richmond	253-9106	3.70
Laura Lee	F	Physics	Seattle, WA, USA	25-8-70	125 Austin Ave., Burnaby	420-5232	3.83
…	…	…	…	…	…	…	…
Removed	Retained	Sci,Eng, Bus	Country	Age range	City	Removed	Excl, VG,..

Prime Generalized Relation

Gender	Major	Birth_region	Age_range	Residence	GPA	Count
M	Science	Canada	20-25	Richmond	Very-good	16
F	Science	Foreign	25-30	Burnaby	Excellent	22
…	…	…	…	…	…	…

Gender \ Birth_Region	Canada	Foreign	Total
M	16	14	30
F	10	22	32
Total	26	36	62

1. Data collection
   target class: graduate student
   contrasting class: undergraduate student
2. Analytical generalization using U_i
   attribute removal
      remove *name* and *phone#*
   attribute generalization
      generalize *major*, *birth_place*, *birth_date* and *gpa*
      accumulate counts
   candidate relation: *gender*, *major*, *birth_country*, *age_range* and *gpa*

Mining ClassComparison

- Comparison: Comparing two or more classes
- Method:
    - Partition the set of relevant data into the target class and the contrasting class(es)
    - Generalize both classes to the same high level concepts
    - Compare tuples with the same high level descriptions
    - Present for every tuple its description and two measures
        - support - distribution within single class
        - comparison - distribution between classes
    - Highlight the tuples with strong discriminant features
- Relevance Analysis:
    - Find attributes (features) which best distinguish different classes

Presentation of Generalized Results

- Generalized relation:
    - Relations where some or all attributes are generalized, with counts or other aggregation values accumulated.
- Cross tabulation:
    - Mapping results into cross tabulation form (similar to contingency tables).
    - Visualization techniques:
    - Pie charts, bar charts, curves, cubes, and other visual forms.
- Quantitative characteristic rules:
    - Mapping generalized result into characteristic rules with quantitative information associated with it, e.g.,
- t-weight:
    - Interesting measure that describes the typicality of
        - each disjunct in the rule
        - each tuple in the corresponding generalized relation
        - n – number of tuples for target class for generalized relation
        - q_i … q_n – tuples for target class in generalized relation
        - q_a is in q_i … q_n

$$t\_weight = count(q_a) / \sum_{i=1}^{n} count(q_i)$$

grad(x) Λ male(x) ⇒ birth_region(x) = “Canadd[t:53%] ∨ birth_region(x) = “foreign[t:47%]

Association Rules

“An association algorithm creates rules that describe how often events have occurred together.”

Example: When a customer buys a hammer, then 90% of the time they will buy nails.

- 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 in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
    - What products were often purchased together?— Beer and diapers?!
    - 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.

<u>Support</u>: "is a measure of what fraction of the population satisfies both the antecedent and the consequent of the rule".

- Example:
    - People who buy hotdog buns also buy hotdog sausages in 99% of cases. = High Support
    - People who buy hotdog buns buy hangers in 0.005% of cases. = Low support
- Situations where there is high support for the antecedent are worth careful attention
    - E.g. Hotdog sausages should be placed in near hotdog buns in supermarkets if there is also high confidence.

<u>Confidence</u>: "is a measure of how often the consequent is true when the antecedent is true."

- Example:
    - 90% of Hotdog bun purchases are accompanied by hotdog sausages.
    - High confidence is meaningful as we can derive rules.
- Hotdog sausage, Hotdog bun
- 2 rules may have different confidence levels and have the same support.
- E.g. Hotdog bun may have a much lower confidence than Hotdog sausage, yet they both can have the same support, Hotdog bun.

<u>Apriori Algorithm</u>

It is a frequent pattern mining algorithm, and findsthe frequent item sets by generating the candidates.

- How to generate candidates?

Step 1: self-joining L_k

Step 2: pruning

- How to count supports of candidates?
    - By counting how many times it hasoccured.

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
$C_4=\{abcd\}$

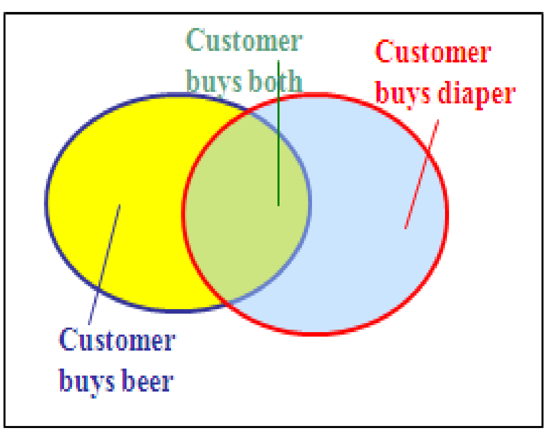
- <u>Pseudo-code</u>:

```
Ck: Candidate itemset of size k
Lk : frequent itemset of size k

L1 = {frequent items};
for (k = 1; Lk !=∅; k++) do begin
    Ck+1 = candidates generated from Lk;
   for each transaction t in database do
           increment the count of all candidates in Ck+1
     that are contained in t
   Lk+1  = candidates in Ck+1 with min_support
   end
return ∪k Lk;
```

<u>Example:</u>

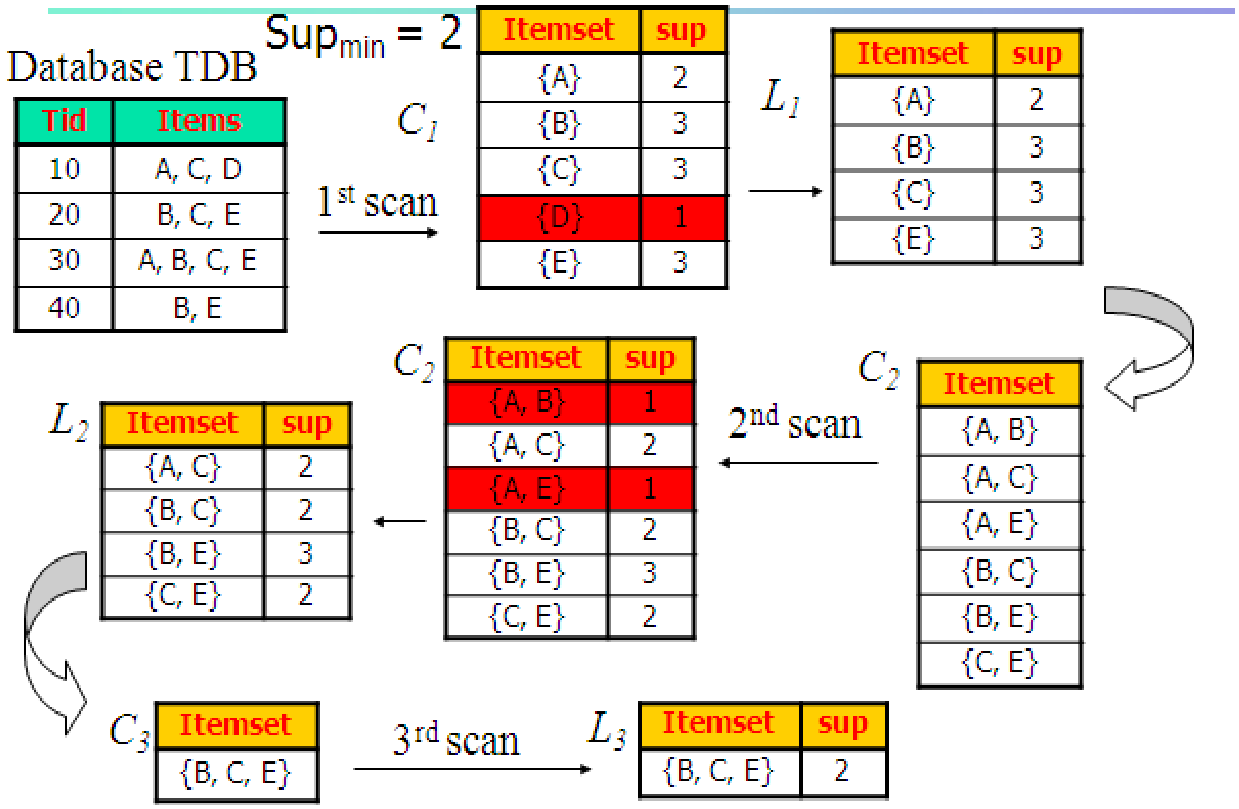
Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F



- 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%)

The Apriori Algorithm—An Example



Frequent Pattern Growth Tree Algorithm

(Mining Frequent Patterns Without Candidate Generation)

It grows long patterns from short ones using local frequent items

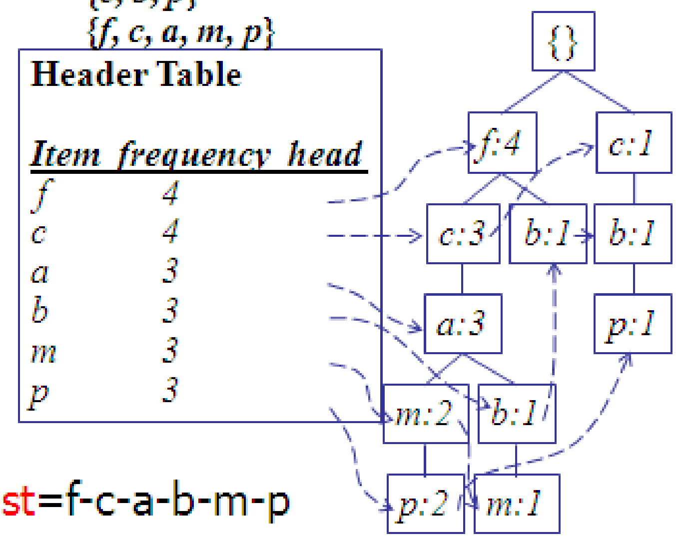
- "abc" is a frequent pattern
- Get all transactions having "abc": DB|abc
- "d" is a local frequent item in DB | abc → abcd is a frequent pattern

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

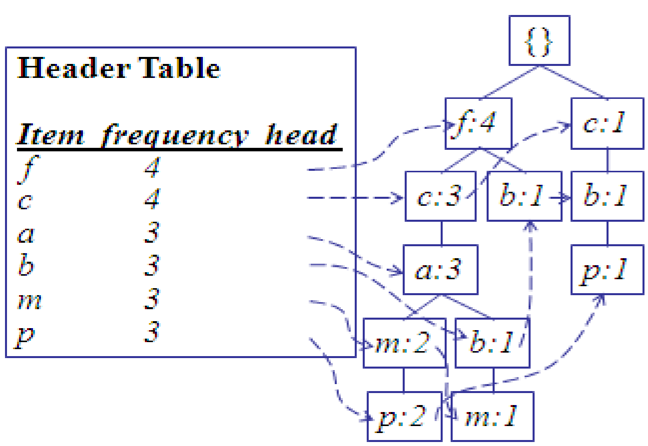
1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree



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



***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*

Mining Multi-Level Associations

- A top_down, progressive deepening approach:
    - First find high-level strong rules:
        - milk -> bread [20%, 60%].
    - Then find their lower-level “weaker” rules:
        - 2% milk -> wheat bread [6%, 50%].
- Variations at mining multiple-level association rules.
    - Level-crossed association rules:
    - *2% milk -> Wonder wheat bread*
    - Association rules with multiple, alternative hierarchies:
    - *2% milk -> Wonder bread*

Multi-level Association: Uniform Support vs. Reduced Support

- Uniform Support: the same minimum support for all levels
    - + One minimum support threshold. No need to examine itemsets containing any item whose ancestors do not have minimum support.
    - – Lower level items do not occur as frequently. If support threshold
        - too high ⇒ miss low level associations
        - too low ⇒ generate too many high level associations
- Reduced Support: reduced minimum support at lower levels
    - There are 4 search strategies:
        - Level-by-level independent
        - Level-cross filtering by k-itemset
        - Level-cross filtering by single item
        - Controlled level-cross filtering by single item

Mining Quantitative Association Rules

- Determine the number of partitions for each quantitative attribute
- Map values/ranges to consecutive integer values such that the order is preserved
- Find the support of each value of the attributes, and combine when support is less than MaxSup. Find frequent itemsets, whose support is larger than MinSup
- Use frequent set to generate association rules
- Pruning out uninteresting rules

Partial Completeness

- R : rules obtained before partition
- R’: rules obtained after partition
- Partial Completeness measures the maximum distance between a rule in R and its closest generalization in R’
- $\hat{X}$ is a generalization of itemset *X*: if

$$\forall x \in attributes(X)[< x, l, u > \in X \wedge < x, l', u' > \in \hat{X} \Rightarrow l' \le l \le u \le u']$$

- The distance is defined by the ratio of support

K-Complete

- C : the set of frequent itemsets
- For any K ≥ 1, P is K-complete w.r.t C if:
    1. $P \quad C$
    2. For any itemset X (or its subset) in C, there exists a generalization whose support is no more than K times that of X (or its subset)
- The smaller K is, the less the information lost

Constraint based Association Mining

- Interactive, exploratory mining giga-bytes of data?
    - Could it be real? — Making good use of constraints!
- What kinds of constraints can be used in mining?
    - Knowledge type constraint: classification, association, etc.
    - Data constraint: SQL-like queries
        - Find product pairs sold together in Vancouver in Dec.'98.
    - Dimension/level constraints:
        - in relevance to region, price, brand, customer category.
    - Rule constraints
        - small sales (price < $10) triggers big sales (sum > $200).
    - Interestingness constraints:
        - strong rules (min_support ≥ 3%, min_confidence ≥ 60%).
- Pattern space pruning constraints
    - Anti-monotonic: If constraint c is violated, its further mining can be terminated
    - Monotonic: If c is satisfied, no need to check c again
    - Succinct: c must be satisfied, so one can start with the data sets satisfying c
    - Convertible: c is not monotonic nor anti-monotonic, but it can be converted into it if items in the transaction can be properly ordered
- Data space pruning constraint
    - Data succinct: Data space can be pruned at the initial pattern mining process
    - Data anti-monotonic: If a transaction t does not satisfy c, t can be pruned from its further mining