JIGSAW ACADEMY

Analytics for Professionals

Association Rule Mining

Association Rule Mining

- Association Rules and Frequent Patterns
- Frequent Pattern Mining Algorithms
  - Apriori
  - FP-growth
- Correlation Analysis

Association Rules

- A Frequent pattern is a pattern (a set of items) that occurs frequently in a data set.
- Motivation: Finding inherent regularities (associations) in data.
- Forms the foundation for many essential data mining tasks:
  - Association, correlation, and causality analysis
  - Classification: associative classification
  - Cluster analysis: frequent pattern-based clustering
- Extended to many different problems: graph mining, sequential pattern mining, times series pattern mining, text mining…

Association Rules

- An item (I) is:
  - For market basket data: I is an item in the store, e.g. milk.
  - For relational data: I is an attribute-value pair (numeric attributes should be discretized), e.g. salary=high, gender=male.
- A pattern (P) is a conjunction of items: $P = I_1 \wedge I_2 \wedge \ldots I_n$ (itemset)
- Pattern P' is subpattern of P if $P' \subset P$
- A rule R is $A \Rightarrow B$ where A and B are disjoint patterns.
- Support $(A \Rightarrow B) = P(A \cup B)$
- Confidence $(A \Rightarrow B) = P(B|A) =$ posterior probability

Association Rules

- Framework: find all the rules that satisfy both a minimum support (*min_sup*) and a minimum confidence (*min_conf*) thresholds.
- Association rule mining:
  - Find all frequent patterns (with support $\geq$ min_sup).
  - Generate strong rules from the frequent patterns.
- The second step is straightforward:
  - For each frequent pattern p, generate all nonempty subsets.
  - For every non-empty subset s, output the rule $s \Rightarrow (p - s)$ if conf $=$ $\text{sup}(p)/\text{sup}(s) \geq \text{min\_conf}$.
- The first step is much more difficult. Hence, we focus on frequent pattern mining.

Example for market basket data

- Items = A,B,C,D,E,F
- Transactions:

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

Let

$$\text{min\_sup} = 60\% \text{ (3 txn)}$$

$$\text{min\_conf} = 50\%$$

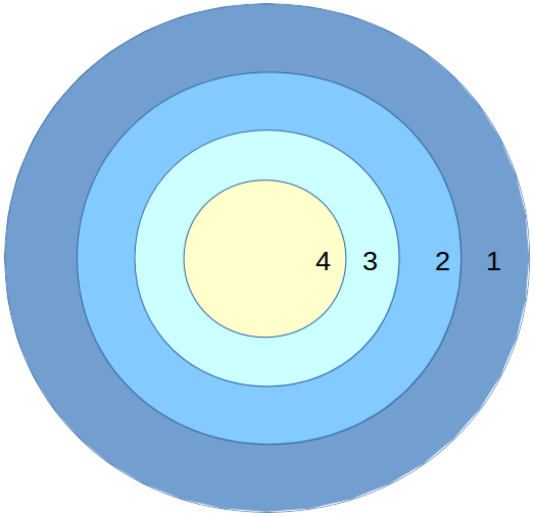
FP = {A:3, B:3, D:4, E:3, AD:3}
Association rules:

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

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

Example for relational data

1. Dataset: 200 patients
2. Smoking = T: subpatterb = super-group
3. Smoking = T $\wedge$ Family history = T (100 patients)
4. Smoke = T $\wedge$ Family history = T $\Rightarrow$ Lung cancer (60 patients)



Rule: Smoke = T $\wedge$ Family history = T $\Rightarrow$ Lung cancer = T

- sup (Smoke = T $\wedge$ Family history = T $\wedge$ Lung cancer = T) = $60/200 = 30\%$
- conf (Smoke = T $\wedge$ Family history = T $\Rightarrow$ Lung cancer = T) = $60/100 = 60\%$

Frequent Pattern Mining

Scalable mining methods: Three major approaches:

- Apriori [Agrawal & Srikant 1994]
- Frequent pattern growth (FP-growth) [Han, Pei & Yin 2000]
- Vertical data format approach [Zaki 2000]

Apriori

- The Apriori property:
  - Any subset of a frequent pattern must be frequent.
  - If {beer, chips, nuts} is frequent, so is {beer, chips}, i.e., every transaction having {beer, chips, nuts} also contains {beer, chips}.
- Apriori pruning principle: If there is any pattern which is infrequent, its superset should not be generated/tested!
- Method (level-wise search):
  - Initially, scan DB once to get frequent 1-itemset
  - For each level k:
    * Generate length (k+1) candidates from length k frequent patterns
    * Scan DB and remove the infrequent candidates
- Terminate when no candidate set can be generated

Apriori

$\text{min\_sup} = 2$

1. Database →

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

2. C_1 →

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

3. L_1 →

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

4. C_2 →

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

5. C_2 →

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

6. L_2 →

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

Apriori

7. L_3

Itemset	sup
{B, C, E}	2

$\rightarrow$

8. C_3

Itemset
{B, C, E}

Apriori

- Candidate generation: Assume we are generating $k+1$ candidates at level k
  - Step 1: self-joining two frequent k-patterns if they have the same $k-1$ prefix
  - Step 2: pruning: remove a candidate if it contains any infrequent k-pattern.
- Example: $L_3 = \{abc, abd, acd, ace, bcd\}$
  - Self-joining: $L_3 \times L_3$
    - abc and $abd \Rightarrow abcd$
    - acd and $ace \Rightarrow acde$
  - Pruning:
    - $acde$ is removed because ade is not in L_3
  - $C_4 = \{abcd\}$

Apriori

The bottleneck of *Apriori*:

- Huge candidate sets:
  - To discover a frequent 100-pattern, e.g., $\{a_1, a_2, \ldots, 100\}$, one needs to generate $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} \approx 10^{30}$ candidates!
- Multiple scans of database:
  - Needs $(n+1)$ scans, n is the length of the longest pattern.

Can we avoid candidate generation?

FP-growth

- The FP-growth algorithm: mining frequent patterns without candidate generation [Han, Pei & Yin 2000]
- Compress a large database into a compact Frequent-Pattern tree (FP- tree) structure
  - highly condensed, but complete for frequent pattern mining
  - avoid costly database scans
- Develop an efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only!

FP-growth

- FP-growth is faster than Apriori because:
  - No candidate generation, no candidate test
  - Use compact data structure
  - Eliminate repeated database scan
  - Basic operation is counting and FP-tree building (no pattern matching)
- Disadvantage: FP-tree may not fit in main memory!

Correlation analysis

- Association rule mining often generates a huge number of rules, but a majority of them either are redundant or do not reflect the true correlation relationship among data objects.
- Some strong association rules (based on support and confidence) can be misleading.
- Correlation analysis can reveal which strong association rules are interesting and useful.

Correlation analysis

Contingency table

	Basketball	Not basketball	Sum (row)
Cereal	2000 (40%)	1750 (35%)	3750 (75%)
Not cereal	1000 (20%)	250 (5%)	1250 (25%)
Sum(col.)	3000 (60%)	2000 (40%)	5000 (100%)

- *play basketball* ⇒ *eat cereal* [40%, 66.7%] is misleading
  - The overall % of students eating cereal is 75% > 66.7%.
- *play basketball* ⇒ *not eat cereal* [20%, 33.3%] is more accurate, although with lower support and confidence

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

Correlation analysis: Lift score

- Lift $= 1 \Rightarrow$ A & B are independent
- Lift $> 1 \Rightarrow$ A & B are positively correlated
- Lift $< 1 \Rightarrow$ A & B are negatively correlated

	Basketball	Not basketball	Sum (row)
Cereal	2000 (40%)	1750 (35%)	3750 (75%)
Not cereal	1000 (20%)	250 (5%)	1250 (25%)
Sum(col.)	3000 (60%)	2000 (40%)	5000 (100%)

$$\text{lift}(\text{basketball} \Rightarrow \text{cereal}) = \frac{2000/5000}{3000/5000 \times 3750/5000} = 0.89$$

$$\text{lift}(\text{basketball} \Rightarrow \neg\text{cereal}) = \frac{1000/5000}{3000/5000 \times 1250/5000} = 0.89$$

Correlation analysis: χ^2 test

- Lift calculates the correlation value, but we could not tell whether the value is statistically significant.
- Pearson Chi-square is the most common test for significance of the relationship between categorical variables

$$\chi^2 = \sum \frac{(O(r) - E(r))^2}{E(r)}$$

- If this value is larger than a cutoff value at a significance level (e.g. at 95% significance level), then we say all the variables are dependent (correlated), else we say all the variables are independent.

Correlation Analysis disadvantages

Problem: Evaluate each rule individually!

- Coronary heart disease: Pr(CHD)=30%
- R2: Family history = yes $\wedge$ Race = Caucasian $\Rightarrow$ CHD [sup=20%, conf=55%]
- R2 is interesting!
- R1: Family history = yes $\Rightarrow$ CHD [sup=50%, conf=60%]
- R2 is not interesting!

We should consider the nested structure of the rules!

Constraint-based 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
- Specify the task relevant data, the relevant attributes, rule templates, additional constraints ...
  - System optimization: explores such constraints for efficient mining – constraint-based mining.

Constraint-based Mining

- Anti-monotonic constraints are very important because they can greatly speed up the mining process.
- Anti-monotonicity exhibit an Apriori-like property:
  - When a pattern violates the constraint, so does any of its superset
  - sum(S.Price) $\leq$ v is anti-monotone
  - sum(S.Price) ge v is not anti-monotone
- Some constraints can be converted into anti-monotone constraints by properly ordering items
  - Example : avg(S.profit) $\geq$ 25
- Order items in value-descending order, it becomes anti-monotone!