Fast Algorithms for Mining Association Rules

Rakesh Agrawal
Ramakrishnan Srikant

Presenter: Dan Li
Discussion: Joel Lanir

Outline
- Introduction
- Algorithm Apriori
- Algorithm AprioriTid
- Comparison of Algorithms
- Conclusion

Data Mining
- What is data mining?
  Tell me what I want to know.
- Example:
  The legend of “diaper” and “beer”

Applications of Data Mining
- Market basket analysis
- Market segmentation
- Customer churn
- Fraud detection
- Direct marketing
- Interactive marketing
- Trend analysis

Association Rule
- Example
  - 90% of transactions that purchase diapers also purchase beer.
  - 10% of all transactions purchase diaper and/or beer.
- Formal Definition
  - $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \emptyset$
  - Confidence $c$, $c\%$ of transactions that contain $X$ also contain $Y$
  - Support $s$, $s\%$ of all transactions contain $X \cup Y$

Discussion
When looking for association rules, when would you consider support and when would you consider confidence?
What considerations should be made when setting the value of both?
For what applications would you set a high confidence value? A high support value?
Association Rule Mining

The problem of finding association rules falls within the purview of database mining.

Eg1: Find all rules that have “Coke” as consequent to boost the sale of Coke.

Eg2: Find all rules relating items located on shelves A and B in the store.

Finding association rules is valuable for

- Cross-marketing
- Catalog design
- Add-on sales
- Store layout and so on

Problem Decomposition

Mining association rules can be decomposed into two sub-problems:

- Discover large itemsets.
- Use the large itemsets to generate the desired rules.

Discovering Large Itemsets

Intuition: any subset of a large itemset must be large.

Algorithms for discovering large itemsets make multiple passes over the data.

- In the first pass, determine which individual item is large.
- In each subsequent pass,
  - Previous large itemsets are used to generate candidate itemsets.
  - Count actual support for the candidate itemsets.
  - Determine which are the real large itemsets.
- This process continues until no new large itemsets are found.

Algorithm Apriori

- The apriori-gen function has two steps
  - OJoin
  - OPrune

  - Efficiently implementing subset functions
  - OThe hash tree

Example for Apriori
Algorithm AprioriTid
1) \( L_1 \) = (large 1-itemsets);
2) \( C_1 = \text{database \( D \)}; 
3) for ( \( k = 2; L_{k-1} \neq \emptyset \); \( k++ \)) do begin 
4) \( C_k = \text{apriori-gen}(L_{k-1}); \) // New candidates 
5) \( C_k \rightarrow \emptyset \); 
6) forall itemset \( I \in C_k \) do begin 
7) // determine candidate itemsets in \( C_k \) contained 
8) if \( \text{in the transaction with identifier \( t \in D \)} \) 
9) \( \{c \in C_k \mid \{c\} \in t \} \cap \{\text{set of elements} \cap \) 
10) \( \{\text{set of elements}\} \}; 
11) \( c \in C_k \), do 
12) \( c \cap \text{const} = \emptyset; 
13)\) if \( \{c \neq \emptyset \) then \( C_k \rightarrow \{c \in t \cap C_k \} \); 
14)\) end 
15) Answer = \( \bigcup_k L_k \);
Example for AprioriTid

<table>
<thead>
<tr>
<th>( C_2 )</th>
<th>( \text{Support} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1, 2}</td>
<td>1</td>
</tr>
<tr>
<td>{1, 3}</td>
<td>1</td>
</tr>
<tr>
<td>{2, 3}</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( C_3 )</th>
<th>( \text{Support} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1, 2, 3}</td>
<td>100</td>
</tr>
<tr>
<td>{1, 2, 4}</td>
<td>50</td>
</tr>
<tr>
<td>{1, 3, 4}</td>
<td>20</td>
</tr>
<tr>
<td>{2, 3, 4}</td>
<td>50</td>
</tr>
</tbody>
</table>

Example for AprioriTid

<table>
<thead>
<tr>
<th>( C_3 )</th>
<th>( \text{Support} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1, 2, 3}</td>
<td>200</td>
</tr>
<tr>
<td>{1, 2, 4}</td>
<td>100</td>
</tr>
</tbody>
</table>

Answer = \( \bigcup_k L_k \)

Comparison of Different Algorithms

<table>
<thead>
<tr>
<th>Candidate Itemsets</th>
<th>Support Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apriori</td>
<td>( I_{k-1} )</td>
</tr>
<tr>
<td>AprioriTid</td>
<td>( I_{k-1} )</td>
</tr>
<tr>
<td>AIS</td>
<td>( I_{k-1} )</td>
</tr>
<tr>
<td>SETM</td>
<td>( I_{k-1} )</td>
</tr>
</tbody>
</table>

Example:

\( L_1 = \{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}\} \)

\( t = \{1, 2, 3, 4, 5\} \)

- Apriori/AprioriTid:
  - After join: \( C_4 = \{\{1, 2, 3, 4\}, \{1, 3, 4, 5\}\} \)
  - After prune \( C_4 = \{\{1, 2, 3, 4\}\} \)

- AIS/SETM
  - \( C_4 = \{\{1, 2, 3, 4\}, \{1, 2, 3, 5\}, \{1, 2, 4, 5\}, \{1, 3, 4, 5\}, \{2, 3, 4, 5\}\} \)

Performance

- Apriori/AprioriTid outperforms AIS/SETM
- AprioriHybrid matches Apriori and AprioriTid when either one wins.

Later Work

- Parallel versions
- Quantitative association rules
  - E.g., “10% of married people between age 50 and 60 have at least 2 cars.”
- Online association rules
Conclusion

- Two new algorithms, Apriori and AprioriTid are discussed.
- These algorithms outperform AIS and SETM.
- Apriori and AprioriTid can be combined into AprioriHybrid.
- AprioriHybrid matches Apriori and AprioriTid when either one wins.

Discussion

This paper has spun off more similar algorithms in the database world than any other data mining algorithm. (866 citations in CiteSeer)

Why do you think this is the case? Is it the algorithm? The problem? The approach? Something else?