Abstract: With the development of the information technology, the amount of information growing explosively, it is difficult to find the useful information which hide behind those mass information. Data mining is a new technology to obtain useful information from a large number of data. Association rule mining is a data mining methods. This project discusses association rule mining in detail based on Apriori. This report is organized as follows: First introduces the basic concepts of apriori, Then analysis the problem based on apriori, Finally test our algorithm to verify the correctness of our design.

Keywords: data mining; association rules; Apriori algorithm

1. Introduction

Association rule mining tries to “find frequent patterns, associations, correlations, or casual structures sets of items or objects in transaction database, relational database, etc.” that is to say, to find out the relation or dependency of occurrence of one of one item based on occurrence of other items. Apriori algorithm is a basic algorithm for association rule mining. In this project, we will introduce Apriori algorithm and how we implement it.

2. System Design

2.1 Question Statement

A supermarket wants to implement a bundling sale. They need to find the items purchased together frequently. It’s a typical market basket analysis problem. This process analyzes customer buying habits by finding associations between the different items that customers place in their “shipping baskets”. The result can help retailers develop marketing strategies by getting to know which items are frequently purchases together by customers. Apriori is a good solution to this Association Rules Mining problem.
Data sets: Transaction records are downloaded from http://wenku.baidu.com/view/972ef7c66137ee06eff91824.html. In this dataset, the total number of transactions of the data used in this experiment is 42, and there are total 17 items.

Concepts:
An item: an item/article in a basket
I: the set of all items sold in the store
A transaction: items purchased in a basket, it may have TID(transaction ID)
A transaction dataset: A set of transactions.

Data model:
I = {i1, i2, ..., im}: a set of items.
Transaction t :
t means a set of items, and t ⊆ l.
Transaction Database T: a set of transactions T = {t1, t2, ..., tn}.
Below are some samples of the dataset:
T1: 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1
T2: 1 0 1 1 0 1 1 0 0 0 0 0 0 0 1
1 means the customer in some transaction buys a kind of item.
0 means the customer in some transaction doesn’t buy a kind of item.

2.2 Implementation environment
We implement and run our code on matlab.

2.3 Apriori Algorithm
In computer science and data mining, Apriori is a classic algorithm for learning associate rules, Apriori is designed to operate on database containing transactions (for example, collections of items bought by customers, or details of a website frequentation). Other algorithms are designed for finding association rules in data having no transactions, or having no timestamps (DNA sequencing).
Apriori is an influential algorithm for mining frequent itemsets for Boolean association rules. The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties, as we shall see below. Apriori employs an iterative approach known as a level-wise search,
where k-itemsets are used to explore (k+1)-itemsets. First, the item of frequent 1-itemsets is found, this set is denoted L₁. L₁ is used to find L₂, the frequent 2-itemsets, which is used to find L₃, and so on, until no more frequent k-itemsets can be found. The finding of each Lₖ require one full scan of the database.

The principle of Apriori algorithm is an example of finding itemsets by reducing the number of candidates, the principle states that if an itemset is frequent, then all of its subsets must also be frequent. This principle holds true because of the anti-monotone property of support.

This means that when I is anti-monotone (or downward closed) and X is a subset of Y, then the support of X, s(X) must not exceed the support of Y, s(Y).

The converse is also true when f is monotone (or upward closed), this means that when X is a subset of Y, then the support of Y, s(Y) must not exceed the support of X, s(X).

We use the Apriori principle mentioned on the previous page in an iterative approach known as a level-wise to reduce the number of itemsets that we have to count their support.

For example: For the transaction database D as follows. There are 9 transactions in this database. |D|=9.

<table>
<thead>
<tr>
<th>Tid</th>
<th>List of item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I, I₂, I₃</td>
</tr>
<tr>
<td>2</td>
<td>I₂, I₄</td>
</tr>
<tr>
<td>3</td>
<td>I₂, I₃</td>
</tr>
<tr>
<td>4</td>
<td>I₂, I₄, I₅</td>
</tr>
<tr>
<td>5</td>
<td>I₃, I₅</td>
</tr>
<tr>
<td>6</td>
<td>I₃, I₅</td>
</tr>
<tr>
<td>7</td>
<td>I₃, I₅</td>
</tr>
<tr>
<td>8</td>
<td>I₂, I₃, I₅</td>
</tr>
<tr>
<td>9</td>
<td>I₂, I₃, I₅</td>
</tr>
</tbody>
</table>

We use the following figure to illustrate the Apriori algorithm to find frequent itemsets in D.
1. The first iteration of the algorithm, each item is a member of the set of candidate 1-itemsets, C1.
   The algorithm scans all of the transaction to count the number of occurrences of each item.

2. Suppose the minimum support count is 2. The set of frequent 1-itemsets, L1, can then be determined. It consists of the candidate 1-items satisfying minimum support.

3. To get frequent 2-itemsets L2, the algorithm uses L1 join L2 to get a candidate set of 2-itemsets, C2. Note that no candidates are removed from C2 during the prune step because each subset of the candidates is also frequent.

4. Next, the transactions in D are scanned and the support count of each candidate item set in C2 is accumulated, as shown in the middle table of the second row in next figure.

5. The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate 2-itemsets in C2 having minimum support.
6. The generation of the set of candidate 3-itemsets, C3, is detailed in next figure. From the join step, we first get \( C_3 = L_2 \text{ join } L_2 = \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}. \) Based on the apriori property that all subsets of a frequent itemset must also be frequent, we can determine that the four latter candidates cannot possibly be frequent. Remove them from C3, thereby saving the effort of unnecessarily obtaining their counts during the subsequent scan of D to determine \( L_3 \). Note that when given a candidate k-itemset, we only need to check if its (k-1)-subsets are frequent since the apriori algorithm uses a level-wise search strategy. The resulting pruned version of C3 is shown in the following figure.

7. The transactions in D are scanned in order to determine \( L_3 \), consisting of those candidate 3-itemsets in C3 having minimum support.

8. Try to find \( C_4 \) until \( C_k = \)

**Pseudocode: Apriori.** Find frequent itemsets using an iterative level-wise approach based on candidate generation.

\( L_k \): Set of frequent itemsets of size \( k \) (those with min support)

\( C_k \): Set of candidate itemset of size \( k \) (potentially frequent itemsets)

\( L_1 = \{\text{frequent items}\}; \)

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

\( C_{k+1} = \text{candidates generated from } L_k; \)

**for each** transaction \( t \) in database do

increment the count of all candidates in \( C_{k+1} \) that are contained in \( t \)

\( L_{k+1} = \text{candidates in } C_{k+1} \text{ with min support} \)

**end**

**return** \( \cup_k L_k; \)

Fig 3: The Apriori Algorithm Pseudocode

Apriori, while historically significant, suffers from a number of inefficiencies or trade-offs, which have spawned other algorithms. Candidate generation generates large numbers of subsets (the algorithm attempts to load up the candidate set with as many as possible before each scan). Bottom-up subset exploration (essentially a breadth-first traversal of the subset lattice) finds any maximal subset \( S \) only after all of its proper subsets.

3. Result
The output of this project is a cell matrix, each cell stores itemsets with a certain number of items. We can get all itemsets based on data sources, support and confidence, which proves that apriori algorithm is effective in association rules mining.

4. Conclusion

This project uses apriori algorithm to resolve association rules mining problem. Apriori algorithm interacts with a transaction record database can get itemsets which customers buy together frequently.

Reference:

4. Data Mining by Dr. Hall (http://www.cse.usf.edu/~hall/dm/)