The Comparative of Boolean Algebra Compress and Apriori Rule Techniques for New Theoretic Association Rule Mining Model

Somboon Anekritmongkol, Kulthon Kasamsan

International Journal of Advancements in Computing Technology, Volume 3, Number 1, February 2011

Abstract

The Data Mining refers to extracting or “mining” knowledge from large amounts of data. The Association Rule the one of technique to knowledge discovery. The Association Rule learning is a popular and well researched method for discovering interesting relations between variables in large databases. One of the most famous association rule learning algorithms is Apriori rule. Apriori algorithm is one of algorithms for generation of association rules. The drawback of Apriori Rule algorithm is the number of time to read data in the database equal number of each candidate is generated. Many research papers have been published trying to reduce the amount of time needed to read data from the database. In this paper, we propose a new algorithm that will work rapidly. Boolean Algebra Compress technique for Association Rule Mining (B-Compress). This algorithm adopts three major ideas. Firstly, compress data. Secondly, reduce the amount of times to scan database tremendously. Thirdly, reduce file size. The construction method of Boolean Algebra Compress technique for association rule mining has ten times higher mining efficiency in execution time than Apriori Rule.

Keywords: Association Rule, Apriori Rule, Boolean Algebra

1. Introduction

Data mining is the process of extracting patterns from data. Data mining is seen as an increasingly important tool by modern business to transform data into an informational advantage. One of the most popular technique in data mining is Apriori rule [1][2][3][6]. The Data mining is usually associated with huge information. Association rules exhaustively look for hidden patterns, making them suitable for discovering predictive rules involving subsets of data set attributes. Association rule learners are used to discover elements that co-occur frequently within a data set consisting of multiple independent selections of elements (such as purchasing transactions), and to discover rules. In my point of view, Firstly, most of information in data set is same pattern. Secondly, amount of time to read the whole database. Thirdly, the pruning candidate in each step of process. This paper proposes the development of algorithm to discover association rules from large amounts of information that is faster than Apriori rule by using Boolean algebra and compressed technique. The improvement focuses on compress data and reducing the number of times to read data from the database.

2. Basic in Association Rule

Let D = {T1, T2, . . . , Tn} [2] be a set of n transactions and let I be a set of items, I = {i1, i2 . . . im}. Each transaction is a set of items, i.e. Ti ⊆ I. An association rule is an implication of the form X ⇒ Y, where X, Y ⊆ I, and X ∩ Y = ∅, X is called the antecedent and Y is called the consequent of the rule. In general, a set of items, such as X or Y, is called an itemset. In this work, a transaction record transformed into a binary format where only positive binary values are included as items. This is done for efficiency purposes because transactions represent sparse binary vectors. Let P(X) be the probability of appearance of itemset X in D and let P(Y |X) be
the conditional probability of appearance of itemset $Y$ given itemset $X$ appears. For an itemset $X \subseteq I$, $\text{support}(X)$ is defined as the fraction of transactions $T_i \in D$ such that $X \subseteq T_i$. That is $P(X) = \text{support}(X)$. The support of a rule $X \Rightarrow Y$ is defined as $\text{support}(X \Rightarrow Y) = P(X) \cap Y$. An association rule $X \Rightarrow Y$ has a measure of reliability called $\text{confidence}(X \Rightarrow Y)$ defined as $P(Y | X) = \text{support}(X \cup Y) / \text{support}(X)$. The standard problem of mining association rules [1] is to find all rules whose metrics are equal to or greater than some specified minimum support and minimum confidence thresholds. A $k$-itemset with support above the minimum threshold is called frequent. We use a third significance metric for association rules called $\text{lift}$ [25]: $\text{lift}(X \Rightarrow Y) = P(Y | X) / P(Y) = \text{confidence}(X \Rightarrow Y) / \text{support}(Y)$. Lift quantifies the predictive power of $X \Rightarrow Y$; we are interested in rules such that $\text{lift}(X \Rightarrow Y) > 1$.

3. Apriori Rule

Finding frequency itemsets using candidate generation. Apriori is a algorithm proposed by R. Agrwal and R. Srikant in 1994. Apriori rule employs an iterative approach know as a level-wise search, where $k$-itemsets are used to explore $(k+1)$-itemsets. First, the set of frequency 1-itemsets is found by scanning the database to accumulate the count of each time and collecting those items satisfy minimum support. The resulting set is $L_1$. Next $L_j$ used to find the set of frequency 2-itemsets, which is used to find, and so on, until no more frequency $k$-itemsets can be found. The finding of each $L_k$ requires one full scan of database.

Algorithm: Apriori rule. Find frequent itemsets using an iterative level-wide approach based on candidate generation.

Input: 1. $D$, a database of transaction; 2. min_sup, The minimum support count threshold.

Output: $L$, frequent itemsets in $D$

Method:

$L_1 = \text{FIND\_FREQUENT\_1\_-ITEMSET}(D)$; for (K=2; $L_{k-1}$ <> 0; K++) {
    $C_k = \text{Apriori\_GEN}(L_{k-1})$;
    For each transaction $T \in D$ { // scan D for count
        $C_t = \text{subset}(C_k)$; // Get subset of t that are candidate
        for each candidate $c \in C_t$
            c.count++;
    }
    $L_k = \{ c \in C_k | c.count \geq \text{min} \_ \text{sup} \}$
}
return $L = \bigcup_k L_k$

Procedure $\text{apriori\_Gen}(L_{k-1}$, frequent $(k-1)$-itemsets)
for each itemset $l_1 \in L_{k-1}$
    for each itemset $l_2 \in L_{k-1}$
        if $l_1[1] = l_2[1]$ and $l_2[2] \neq l_2[2] \neq \ldots \neq l_2[k-2] = l_2[k-2] \neq l_2[k-1]$ then {
            $C = l_1 \bowtie l_2$; // join step : generate candidates
            if $\text{has\_in frequent\_subset}(c, L_{k-1})$ then
                delete $c$; // prune step : remove unfruitful candidate
            else add $c$ to $C_k$;
        }
    }
return $C_k$

Procedure $\text{has\_in frequent\_subset}(c$: candidate $k$-itemset;
$L_{k-1}$ : frequent $(k-1)$-itemsets); // use prior knowledge
for each $(k-1)$-subset $s$ of $c$
    if $s \not\subset L_{k-1}$ then
4. Boolean Algebra

Boolean algebra developed in 1854 by George Boole in his book An Investigation of the Laws of Thought. Some operations of ordinary algebra, in particular multiplication $xy$, addition $x+y$, and negation $-x$, have their counterparts in Boolean algebra, respectively the Boolean Operations AND, OR, and NOT also called conjunction $x \land y$, disjunction $x \lor y$, and negation or complement $\neg x$ sometime $!x$. Some authors use instead the same arithmetic operations as ordinary algebra reinterpreted for Boolean algebra, treating $xy$ as synonymous with $x \land y$ and $x+y$ with $x \lor y$.

**Conjunction** $x \land y$ behaves on 0 and 1 as exactly multiplication does for ordinary algebra: if either $x$ or $y$ is 0, but if both are 1 then $x \land y$ is 1.

**Disjunction** $x \lor y$ works almost like addition, with $0 \lor 0 = 0$ and $1 \lor 0 = 0 \lor 1 = 1$. However, there is a difference: $1 \lor 1$ is not 2 but 1.

**Complement** resembles ordinary negation in that it exchanges values. Complement can be defined arithmetically as $\neg x = 1 - x$.

5. Boolean algebra and data compression

Algorithm of Boolean algebra and Data compression, 1. Find frequent itemsets $\geq$ minimum support. 2. Compress data by create a new structure are same pattern of transactions. 3. Candidate 2-itemsets are generated and count $\geq$ minimum support. 4. Using Candidate 2-Itemsets to generate candidate 3-
itemsets. 5. Generate candidate 4-itemsets until k-itemsets by generate on actual data in pattern transactions.

Algorithm: Pseudo-code of Boolean algebra and Data compression.

Tid-cl, a table of transaction
Tid-cl2, a table contain candidate
Final Candidate, the result of candidates $\geq$ Min_sup
$T_k$, Itemset
Tid, transaction ID
Min_sup, Minimum support count all transactions

Procedure Find L1
(1) for each itemset count $T_k \in$ Tid-cl;
(2) Delete $T_k <$ Min_sup;
(3) add to Final Candidate;

Procedure Compress Structure
(1) FOR EACH TID, TID_k $\in$ L1 {
(2) IF SAME TID_k THEN
(3) New_structurek = New_structurek + $T_k$;
(5) else
(6) New_Structure.countk++;
(7) }
(8) for each New_structurek
(9) Update Tid_cl.feq;
(10) for each New_structurek
(11) delete Tid_cl.feq = 0;

Procedure Generate Candidate-2
(1) for each itemset $T_1 \in$ Tid-cl,$T_{k-1}$
(2) for each itemset $T_2 \in$ Tid-cl,$T_{k-1}$
(3) if($T_2_i > T_1_i$) then
(4) Add $T_2 \cup T_1$ to Tid-cl2;

Procedure Generate Associate Data
(1) FIND L1;
(2) COMPRESS STRUCTURE;
(3) Generate Candidate-2;
(4) for each (k=2; L_k; k++){
(5) for each Tid_cl2 Delete Tid_cl2.$T_k <$ Min_sup
(6) for each Tid_cl2 add to final Candidate
(7) for each itemset (j=1 ; Tid_cl2.eof ; j++)
(8) add Tid_cl2.$T_j$ $\cup$ Tid_cl.$T_i$;
(9) Delete Tid_cl2.$T_k$;
(10) Delete Tid_cl.No_of_Attribute = k-1;
(11) }
The Comparative of Boolean Algebra Compress and Apriori Rule Techniques for New Theoretic Association Rule Mining Model

Somboon Anekritmongkol, Kulthon Kasamsan

International Journal of Advancements in Computing Technology, Volume 3, Number 1, February 2011

Table 1. Transaction Data

<table>
<thead>
<tr>
<th>Trans#</th>
<th>Item</th>
<th>Trans#</th>
<th>Item</th>
<th>Trans#</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>T001</td>
<td>I1</td>
<td>T006</td>
<td>I2</td>
<td>T011</td>
<td>I1</td>
</tr>
<tr>
<td>T001</td>
<td>I2</td>
<td>T006</td>
<td>I3</td>
<td>T011</td>
<td>I2</td>
</tr>
<tr>
<td>T001</td>
<td>I5</td>
<td>T007</td>
<td>I2</td>
<td>T011</td>
<td>I3</td>
</tr>
<tr>
<td>T002</td>
<td>I2</td>
<td>T007</td>
<td>I3</td>
<td>T012</td>
<td>I2</td>
</tr>
<tr>
<td>T002</td>
<td>I4</td>
<td>T008</td>
<td>I1</td>
<td>T012</td>
<td>I3</td>
</tr>
<tr>
<td>T003</td>
<td>I2</td>
<td>T008</td>
<td>I2</td>
<td>T013</td>
<td>I1</td>
</tr>
<tr>
<td>T003</td>
<td>I3</td>
<td>T008</td>
<td>I3</td>
<td>T013</td>
<td>I2</td>
</tr>
<tr>
<td>T004</td>
<td>I1</td>
<td>T009</td>
<td>I1</td>
<td>T013</td>
<td>I4</td>
</tr>
<tr>
<td>T004</td>
<td>I2</td>
<td>T009</td>
<td>I2</td>
<td>T014</td>
<td>I1</td>
</tr>
<tr>
<td>T004</td>
<td>I3</td>
<td>T009</td>
<td>I3</td>
<td>T014</td>
<td>I2</td>
</tr>
<tr>
<td>T004</td>
<td>I4</td>
<td>T010</td>
<td>I1</td>
<td>T014</td>
<td>I3</td>
</tr>
<tr>
<td>T005</td>
<td>I1</td>
<td>T010</td>
<td>I2</td>
<td>T015</td>
<td>I2</td>
</tr>
<tr>
<td>T005</td>
<td>I3</td>
<td>T010</td>
<td>I4</td>
<td>T015</td>
<td>I3</td>
</tr>
</tbody>
</table>

Step 1: Find L1
Find frequency itemset of each item and remove frequency itemsets less than minimum support.

Table 2. Support count

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Support Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{I1}</td>
<td>9</td>
</tr>
<tr>
<td>{I2}</td>
<td>14</td>
</tr>
<tr>
<td>{I3}</td>
<td>11</td>
</tr>
<tr>
<td>{I4}</td>
<td>4</td>
</tr>
<tr>
<td>{I5}</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3. Result of candidate-1 Itemset

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Support Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{I1}</td>
<td>9</td>
</tr>
<tr>
<td>{I2}</td>
<td>14</td>
</tr>
<tr>
<td>{I3}</td>
<td>11</td>
</tr>
<tr>
<td>{I4}</td>
<td>4</td>
</tr>
</tbody>
</table>

Step 2: Compress Data
Count transactions are same item such as T003 = {I2, I3}, T006 = {I2, I3}, T007 = {I2, I3}, T012 = {I2, I3} and T015 = {I2, I3} or {I2, I3} = {(T003), (T006), (T007), (T012), (T015)} = 5. Create pattern is same structure as T001 → create pattern1, T002 → pattern2, {T003, T006, T007, T012, T015} → pattern3, T004 → pattern4, {T010,T013} → pattern 5, T005 → pattern6 and {T008,T009,T011, T014} → pattern7.

Table 4. Data Compression

<table>
<thead>
<tr>
<th>Pattern-1</th>
<th>I1</th>
<th>I2</th>
<th>I3</th>
<th>I4</th>
<th>Count</th>
<th>No. Of Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern-2</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Pattern-3</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Pattern-4</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Pattern-5</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Pattern-6</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Pattern-7</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

Step 3: Find Candidate-2 Items
To discover the set of frequency 2-itemsets, Generate candidate from Pattern table itemsets, to itemsets, and count number of candidate. Starting from candidate-2 itemsets by Tid cl.Itemset, ∪ (Tid cl.Itemset > Tid cl.Itemset), scan for all pattern transactions table. Then remove each candidate-2 itemsets < minimum support.
The Comparative of Boolean Algebra Compress and Apriori Rule Techniques for New Theoretic Association Rule Mining Model
Somboon Anekritmongkol, Kulthon Kasamsan
International Journal of Advancements in Computing Technology, Volume 3, Number 1, February 2011

\[
\begin{align*}
I_1, I_2 &= \{P_1, P_4, P_5, P_7\} = 1 + 1 + 2 + 4 = 8 \\
I_1, I_3 &= \{P_4, P_6, P_7\} = 1 + 1 + 4 = 6 \\
I_1, I_4 &= \{P_4, P_5\} = 1 + 2 = 3 \\
I_2, I_3 &= \{P_3, P_4, P_7\} = 5 + 1 + 4 = 10 \\
I_2, I_4 &= \{P_2, P_4, P_5\} = 1 + 1 + 2 = 4
\end{align*}
\]

**Figure 3.** Find candidate-2 itemsets

<table>
<thead>
<tr>
<th>Itemsets ((C_2))</th>
<th>Support Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{I_1, I_2}</td>
<td>8</td>
</tr>
<tr>
<td>{I_1, I_3}</td>
<td>6</td>
</tr>
<tr>
<td>{I_1, I_4}</td>
<td>3</td>
</tr>
<tr>
<td>{I_2, I_3}</td>
<td>10</td>
</tr>
<tr>
<td>{I_2, I_4}</td>
<td>4</td>
</tr>
</tbody>
</table>

**Table 5.** Result of candidate-1 itemsets

Step 4: Find Candidate-3 Itemsets
Generate Candidate-3 Itemsets by Itemset, \( \cup C_2 \), scan for all candidate-2 itemsets, each candidate-2 Itemset scan all pattern transactions table and count. Then remove each candidate-3 itemsets < minimum support.

\[
\begin{align*}
\{I_3\} &\cup \{I_4\} = \{P_4\} = 1 \\
\{I_1, I_2, I_3\} &= \{P_4, P_7\} = 1 + 4 = 5 \\
\{I_1, I_2, I_4\} &= \{P_4, P_5\} = 1 + 2 = 3 \\
\{I_2, I_3, I_4\} &= \{P_4\} = 1
\end{align*}
\]

**Figure 4.** Candidate-3 itemset

<table>
<thead>
<tr>
<th>Itemsets ((C_2))</th>
<th>Support Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{I_1, I_2, I_3}</td>
<td>5</td>
</tr>
<tr>
<td>{I_1, I_2, I_4}</td>
<td>3</td>
</tr>
</tbody>
</table>

**Table 6.** Result of candidate-3 itemsets

**Table 7.** Final result (Support count > minimum support)

<table>
<thead>
<tr>
<th>Itemsets ((C_2))</th>
<th>Support Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{I_1}</td>
<td>9</td>
</tr>
<tr>
<td>{I_2}</td>
<td>14</td>
</tr>
<tr>
<td>{I_3}</td>
<td>11</td>
</tr>
<tr>
<td>{I_4}</td>
<td>4</td>
</tr>
<tr>
<td>{I_1, I_2}</td>
<td>8</td>
</tr>
<tr>
<td>{I_1, I_3}</td>
<td>6</td>
</tr>
<tr>
<td>{I_1, I_4}</td>
<td>3</td>
</tr>
<tr>
<td>{I_2, I_3}</td>
<td>10</td>
</tr>
<tr>
<td>{I_2, I_4}</td>
<td>4</td>
</tr>
<tr>
<td>{I_1, I_2, I_3}</td>
<td>5</td>
</tr>
<tr>
<td>{I_1, I_3, I_4}</td>
<td>3</td>
</tr>
</tbody>
</table>
6. Experiments

In this section, we performed a set of experiments to evaluate the effectiveness of B-Compress. The experiment dataset consists of two kinds of data, First, data from Phranakorn Yontrakarn Co., Ltd. This company sales and offer car services to discover association data. Second, generate sampling data. The experiment of four criteria, Firstly, Increase amount of records from 10,000 to 50,000 records and fixed 10 itemsets (Fig.5, Fig.6). Secondly, increase of itemsets and fixed amount of records = 50,000(Fig.7, Fig.8). Thirdly, increase amount of records from 10,000 to 190,000 records (Fig.9). Fourthly, Increase of minimum support and fixed itemsets and amount of records (Fig.10).

Experiment 1: Increase number of records. Step 10,000 records. Fixed 10 itemsets. Compare Apriori Rule with Boolean Algebra Compress Technique for Association Rule Mining. Apriori rule, with increasing amount of record will take longer time. B-Compress, with increasing amount of record will take a little time.

![Figure 5. Data from Phranakorn Yontrakarn Co., Ltd., Increase number of records](image_url)

![Figure 6. Sampling Data, Itemset =10, Increase number of records](image_url)

Experiment 2: Increase itemsets. Fixed number of records 50,000 records. Compare Apriori Rule with Boolean Algebra Compress Technique for Association Rule Mining. Apriori rule, with increasing itemsets will take longer time. B-Compress, with increasing itemsets will take a shorter time. Fig. 8 sampling Data, B-compress is slightly increase because of density of information is 50 percent. Each record have 50 percent of itemsets such as ten itemsets, each transaction ID will have 5 itemsets.
The Comparative of Boolean Algebra Compress and Apriori Rule Techniques for New Theoretic Association Rule Mining Model
Somboon Anekritmongkol, Kulthon Kasamsan
International Journal of Advancements in Computing Technology, Volume 3, Number 1, February 2011

Experiment 3: Increase number of records from 10,000 to 190,000 records. Compare Apriori Rule with Boolean Algebra Compress Technique for Association Rule Mining. From actual data from Phranakorn Yontrakarn Co., Ltd. shows effective performance between Apriori rule and B-Compress as 190,000 records, Apriori rule takes 238 seconds but B-Compress takes only 5 seconds on processing time. That it mean amount of records will be slightly affected B-Compress but Apriori rule performance takes longer.

Figure 7. Data from Phranakorn Yontrakarn Co., Ltd., Increase itemsets

Figure 8. Sampling Data, Increase Itemsets, Number of records = 50,000 records

Figure 9. Increase number of records and itemsets
Experiment 4: Increase of minimum support from 10 percent to 60 percent. The step to change minimum support, Apriori rule low minimum support takes time to process but B-Compress slightly affects the performance because B-Compress compresses data, process only the actual data and reduce to each candidate.

![Figure 10. Performance of Apriori Rule and B-Compress](image)

<table>
<thead>
<tr>
<th>Minimum Support (%)</th>
<th>Number of Candidates</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1022</td>
</tr>
<tr>
<td>20</td>
<td>916</td>
</tr>
<tr>
<td>30</td>
<td>615</td>
</tr>
<tr>
<td>40</td>
<td>301</td>
</tr>
<tr>
<td>50</td>
<td>141</td>
</tr>
<tr>
<td>60</td>
<td>55</td>
</tr>
</tbody>
</table>

The result of experiments, Boolean Algebra Compress Technique for Association Rule Mining discovers an association is faster than Apriori rule. If increasing the number of records, Apriori rule will take time to read the whole data. If increasing the number of itemset, Apriori rule will create more candidates depending on the number of items set but Boolean Algebra Compress Technique for Association Rule Mining takes shorter time because it will compress data and create candidates in existing data only and delete candidates’ data that are lower than minimum support for each candidate.

7. Conclusion

The paper proposes a new association rule mining theoretic models and designs a new algorithm base on established theories. B-Compress compresses data, processes only the actual data and reduces the data of each candidate. The Boolean Algebra Compress Technique for Association Rule Mining is able to discover data more than ten times faster than Apriori rule.

From the experiments we found that the number of records will not affect the performance of Boolean Algebra Compress Technique for the Association Rule Mining but the number of Itemsets will affect the performance slightly. The results are especially amplified in itemsets experiment 3 where we used data from 10,000 to 190,000 records. Boolean Algebra Compress Technique for Association Rule Mining has proven to have the best performance. We believe that B-Compress is an important way to find Association data or Data mining.

However, in the experiment, Boolean Algebra Compress Technique for Association Rule Mining required hard disk space to generate candidate data. In the future, performance could be improved
further by reducing the number of candidates as well as generate the candidates in RAM instead of hard disks

8. References