An Improved Apriori-based Algorithm for Association Rules Mining

Huan Wu, Zhigang Lu, Lin Pan, Rongsheng Xu
Computing Center
Institute of High Energy Physics, Chinese Academy of Sciences
Beijing 100049, China
e-mail: {wu-h, luzhigang, panl, xurs}@ihep.ac.cn

Wenbao Jiang
School of Information Management
Beijing Information Science & Technology University
Beijing 100101, China
e-mail: jiangwenbao@tsinghua.org.cn

Abstract—Because of the rapid growth in worldwide information, efficiency of association rules mining (ARM) has been concerned for several years. In this paper, based on the original Apriori algorithm, an improved algorithm IAA is proposed. IAA adopts a new count-based method to prune candidate itemsets and uses generation record to reduce total data scan amount. Experiments demonstrate that our algorithm outperforms the original Apriori and some other existing ARM methods.

Keywords—association rules mining; Apriori

I. INTRODUCTION

Nowadays, with the rapid development of information technology, especially the web service-based application, service-oriented architecture and cloud-computing, continually expanding data are integrated to generate useful information. Many techniques have been used for data mining. Association rules mining (ARM) is one of the most useful techniques. The challenges associated with ARM, especially for parallel and distributed data mining, include minimizing I/O, increasing processing speed and reducing communication cost [3]. A major concern in ARM today is to continue to improve algorithm performance.

The Apriori-based algorithms find frequent itemsets based upon an iterative bottom-up approach to generate candidate itemsets. Since the first proposal of association rules mining by R. Agrawal [1, 2], many researches have been done to make frequent itemsets mining scalable and efficient. But there are still some deficiencies that Apriori-based algorithms suffered from, which include: too many scans of the transaction database when seeking frequent itemsets, large amount of candidate itemsets generated unnecessarily and so on.

In this paper, on the basis of analysis and study of existing research efforts, an improved Apriori-based algorithm (IAA) for ARM is proposed. Experimental results demonstrate that this algorithm will significantly improve the performance of the original algorithm by decreasing the time of prune operation and candidate itemsets verification operation.

The paper is organized as follows: Section 2 introduces related work on ARM. Our algorithm is proposed in section 3. Section 4 gives the experimental results. Finally the conclusion is given.
such as HPA [3], WDPA [5], Apriori-T [9], etc. HDSS [10] implements ARM on grid environments, and some works have been done in peer-to-peer system.

FIMI’03 has tested some existing frequent itemsets mining algorithms, and performed timing and memory usage experiments for given datasets. According to the experiment report, there are no clear winners for all kinds of datasets [11]. A generating artificial data model has been proposed in [12] to produce realistic synthetic data for testing ARM algorithms. Algorithms of ARM for specific dataset are still needed.

III. THE IAA ALGORITHM

After generating the 1-dimensional frequent itemsets, every round of the original Apriori algorithm can be separated into 2 steps, pruning candidates and counting candidate itemsets occurrence. Our algorithm improves prune operation by using a count-based method; the count occurrence operation is improved by decreasing the mount of scan data using generation record.

Details of IAA algorithm are given below.

A. Count-based Candidate Prune Method

Let \( L_k \) denotes the set of k-dimensional frequent itemsets. \( C_k \) denotes the set of k-dimensional candidate itemsets. \( \overline{L_k} \) denotes the complement of set \( L_k \) where the universal set is \( C_k \). And let \( s \) denotes an (k-1)-dimensional subset of \( C_k \).

[Theorem 1] If the itemset \( Y \) is frequent, then every subset of \( Y \) will also be frequent. \([1]\)

[Deduction 1] If the itemset \( Y \) is infrequent, then every superset of \( Y \) will also be infrequent.

To determine if \( c \) is a frequent itemset, the original Apriori algorithm use \( L_{k-1} \) , if \( \exists s \notin L_{k-1} \) then \( c \) is not a frequent itemset. The HDO Apriori algorithm proposed in [4] uses \( L_{k-1} \) to delete infrequent \( c \in C_k \), if \( \exists s \notin \overline{L_{k-1}} \) then \( c \) is not a frequent itemset. The prune runtime of both operations depend on the length of the reference collection \( L_{k-1} \) or \( \overline{L_{k-1}} \).

In our algorithm, we use a count-based candidate prune operation. The procedure of Apriori-gen function is modified as Fig.1. Note that we also record the shortest two (k-1)-dimensional subsets \( c_1 \) and \( c_2 \) of a k-dimensional candidate itemsets with only one different item, which will be used in the next step.

To prove the validity of count-based method, we use a deduction of theorem 1.

[Deduction 2] All k-dimensional frequent candidate itemsets can be generate from (k-1)-dimensional frequent itemsets with only one different item.

Proof: Let \( \{L,a,b\} \) and \( \{L,c,d\} \) are (k-1)-dimensional frequent itemsets with two different items, if k-dimensional itemsets \( \{L,a,b,c\} \) is frequent, due to theorem 1, \( \{L,a,b\} \) must also be a frequent itemset. So \( \{L,a,b,c\} \) can be generated from \( \{L,a,b\} \) and \( \{L,a,b\} \), the generate operation of \( \{L,a,b\} \) and \( \{L,c,d\} \) is redundant.

Hence, if k-dimensional candidate itemsets \( c \) is frequent, all its k (k-1)-dimensional subsets are also frequent, each two of these subsets will generate \( c \) once, and the total time will be the combination number \( C_k^2 \). If the count can’t reach \( C_k^2 \), the candidate itemset must have infrequent subset(s). Due to deduction 1, this candidate itemset must be infrequent.

B. Candidate Generation Record for Count Occurrence

One deficiency of Apriori is the multiple time data scans. The Apriori-Tid [2] algorithm uses a <TID, itemsets> structure to store data, and only rescans the transactions that support (k-1)-dimensional frequent itemsets each round. Let \( N_{\overline{L_{k-1}}} \) denotes the number of transactions used in k-th round, \( N_{c} \) denotes the number of k-dimensional candidate itemsets, \( N_t \) denotes the number of items in transaction \( t \). \( N_c \) denotes the number of items in candidate itemset \( c \). Equation (3.1) represents the total count time of k-th round, where \( \sum(N_t,N_c) \) gives the count time of determining whether \( t \) supports \( c \) or not.

\[
T_{\text{Apriori-Tid}}^k = \sum_{t \in T} \sum_{c \in C_k} \text{fun}(N_t,N_c) \quad (3.1)
\]

Count-based Apriori-gen function:
Input: (k-1)-dimensional frequent itemsets \( L_{k-1} \).
Output: k-dimensional frequent itemsets \( C_k \).

begin
\( C_k = \emptyset \)
foreach \( l_1 \in L_{k-1} \), \( l_2 \in L_{k-1} \), \( l_1 \neq l_2 \) {
if \( l_1[1]=l_2[1]=\ldots=l_1[k-2]=l_2[k-2]=l_1[k-1] \neq l_2[k-1] \)
\( c=l_1 \cup l_2 \)
if (\( c \in C_k \)) \( c \).count++
else \( c \).count=1, insert \( c \) into \( C_k \)
endif
if \( l_1 \).length=\( l_2 \).length<\( \text{MinGenSubsetsLength} \)
update \( \text{MinGenSubsetsLength} \)
record \( c \) is generate from \( l_1 \) and \( l_2 \)
endif
}
foreach \( c \in C_k \)
if \( c \).count<\( C_k^2 \), remove \( c \) from \( C_k \)
end
Because all the transactions that support $c \in C_k$ should also support the (k-1)-dimensional subsets $c_1 \text{or}/\text{and} c_2$. In the count occurrence step, to count the number of transactions that support a k-dimensional candidate itemsets, we only use the transactions that support $c_1 \text{or}/\text{and} c_2$ based on the generate record produced in prune operation.

Fig.2 illustrates an example of <itemset, TIDs> structure. Assuming $\{ab, ac, bc, cd, ef\}$ is the set of 2-dimensional frequent itemsets $L_2$. In WDPA, data of $\{ab, ac\}$, $\{ab, bc\}$, $\{ab, cd\}$, $\{ab, ef\}$, $\{ac, bc\}$, $\{ac, cd\}$, $\{ac, ef\}$, $\{bc, cd\}$, $\{bc, ef\}$ and $\{dc, ef\}$ will be counted. In our algorithm, $\{ab, dc\}$, $\{ab, ef\}$, $\{ac, ef\}$ and $\{bc, ef\}$ have more than one different item, they will not be counted, since $ad$ (subset of $acd$) and $bd$ (subset of $bed$) are not frequent, $\{ac, cd\}$ and $\{bc, ed\}$ will not be counted, for the last three pairs, they will all generate itemset $abc$, but due to the generation record, only $\{ac, bc\}$ will be counted. Fig.3 illustrates this difference, all 10 white blocks represent the generation works in WDPA, and only one block with $\checkmark$ mark will be done in IAA, the generation record of $abc$ is $\{ac, bc\}$.

Thus, the total count time of k-th round in IAA wound be (3.2), where $c_1$ and $c_2$ are the shortest two (k-1)-dimensional subsets of $c$. Experiment in later section will show that this method will reduce the count times and saves plenty of time.

$$T_{\text{IAA}}^k = \sum_{c \in \mathcal{C}} \min \{N_{c_1}, N_{c_2}\} \quad \text{(3.2)}$$

C. The Whole Algorithm

The ARM problem has first arisen in basket market analysis, where every relation between items or itemsets will be considered. Whereas in some particular situation, we are only interest in some kind of relations, For instance, when data mining in network intrusion log file, we want to know what combination of attributions is the signal of certain type of action. The relations between these attributions are useless. Let $C$ denotes the condition items set, $R$ denotes the result items set, and $C \cap R = \emptyset, I = \emptyset \cup R$. The objective of this problem is to find all $c \Rightarrow r$ association rules where $c \in C, r \in R$. Apparently, the result of this condition-result problem (C-R problem) $S_c$ is a subset of $S$.

In C-R problem, only ‘ccrr’ style itemsets need to be calculated. IAA algorithm generates frequent itemsets and association rules synchronously. Assuming $c_1c_2\eta_2$ is a 4-dimensional frequent itemset, due to theorem 1, $c_1c_2\eta_1$, $c_1c_2\eta_1$ and $c_2\eta_2$ are all 3-dimensional frequent itemsets. Since the support and confidence of $c_1c_2\eta_1$, $c_1c_2\eta_1$ and $c_2\eta_2$ have already been calculated.

![Figure 2. <itemset, TIDs> data structure example](image2)

![Figure 3. WDPA and IAA work amount comparison](image3)

**Improved Apriori-based Algorithm (IAA):**

Input: $DB_T$, $\text{Sup}_{\min}$ and $\text{Conf}_{\min}$.

Output: association rules.

begin

scan database get $L_4$ using (2.1)
build <itemsets, TIDs> of $L_4$
k=2;
while($L_k-1 \neq \emptyset$) {
    //use count-based gen function
    $C_k$ = Count-based Apriori-gen ($L_{k-1}$)
    foreach $c \in C_k$ {
        // build <c, TIDs>
        foreach tid $\in L_k$ TIDs {
            if (tid $\in L_{k-1}$ TIDs)
                c.TIDcount++;
            c.TIDs = c.TIDs $\cup$ tid
        }
        $L_k$ = {c$\in C_k, \text{Sup}_c > \text{Sup}_{\min}$}
        foreach le $\in L_k$
            use (2.2)(2.3) to generate $c \Rightarrow r$ rules
        k++
    }
}
end

![Figure 4. Improved Apriori-based Algorithm (IAA)](image4)
in the 3rd round, and the 2-dimensional frequent itemsets are analogous, only \( c_1 \subseteq c_2 \Rightarrow n_2 \) is need to be considered from \( c_1 \subseteq c_2 \). Thus, in synchronization generation operation, one frequent itemset will generate only one association rule. After each calculate round, all of frequent ‘cccc’ style itemsets are reserved in memory for further calculating work; the ‘ccrr’ and ‘rrrr’ style itemsets and infrequent ‘cccc’ style itemsets can be removed from memory.

The benefits of synchronization generation are that we can stop operation before all large frequent itemset has been computed if the existing results don’t match our expectant due to inappropriate pre-defined \( \text{Sup}_{\min} \) or if we only want certain number of association rules. Besides, in parallel and distributed data mining, this work can be done in master node or idle nodes, which will save the whole operation time of ARM.

The whole algorithm is as Fig.4.

IV. EXPERIMENT

All the experiments are performed on a 2.2GHz Intel(R) Core(TM) 2 Duo PC with 2048MB memory, running on the Windows XP Professional OS. Programs are coded in C# on the platform of Microsoft.Net Framework 3.5.

The first experiment compares the prune efficiency of original Apriori, HDO and IAA. Synthesized dataset T10I5D10KN50 generated by IBM's Quest Synthetic Data Generator [6] was used. The result is shown in Fig.5, the count-based prune operation used in IAA is much efficient than the others.

The second experiment compares the whole runtime of original Apriori-Tid, Modified WDPA (using single process and only considering (k-1)-dimensional frequent itemsets with one different item to generate k-dimensional candidate itemsets) and IAA with different minimum support use dataset T10I4D10KN100.

WDPA and IAA will spend some time in building <itemsets, TIDs> structure items which may never be used, especially in the last round. When minimum support is greater than 3%, the whole ARM operation finishes in only 2 or 3 rounds, and a large number of frequent itemsets are generated in the last round. As illustrated in Fig.6, both the two algorithms are slower than Apriori-Tid. With minimum support decreased, while operation round increased, the time saved from data scanning in WDPA and IAA becomes prominent.

With the help of count-based prune operation and candidate generation record, IAA performs better than the other algorithms. Meanwhile, IAA records the TIDs length of each (k-1)-dimensional frequent itemsets, so the weighted load-balance strategy proposed in WDPA can be adopted when parallel or distributed data mining using IAA.

V. CONCLUSION

In this paper, an improved Apriori-based algorithm IAA is proposed. Through pruning candidate itemsets by a new count-based method and decreasing the amount of scan data by candidate generation record, this algorithm can reduce the redundant operation while generating frequent itemsets and association rules in the database. Validated by the experiments, the improvement is notable.

This work is part of our Distributed Network Behavior Analysis System, though we have considered C-R problem in our algorithm, for specific dataset, more work is still needed. We also need further research to implement this algorithm in our distributed system.

ACKNOWLEDGMENT

This research was supported by the Natural Science Foundation of Beijing under Grant No. 4072010 and the Natural Science Foundation of Hainan Province under Grant No. 80639.
REFERENCES


