An improved Apriori algorithm for mining association rules

Cite as: AIP Conference Proceedings **1820**, 080005 (2017); https://doi.org/10.1063/1.4977361
Published Online: 13 March 2017

Xiuli Yuan

An Improved Apriori Algorithm for Mining Association Rules

Xiuli Yuan

School of Shanghai University, Shanghai 200444, China.

rachelyxl@shu.edu.cn

Abstract. Among mining algorithms based on association rules, Apriori technique, mining frequent itermsets and interesting associations in transaction database, is not only the first used association rule mining technique but also the most popular one. After studying, it is found out that the traditional Apriori algorithms have two major bottlenecks: scanning the database frequently; generating a large number of candidate sets. Based on the inherent defects of Apriori algorithm, some related improvements are carried out: 1) using new database mapping way to avoid scanning the database repeatedly; 2) further pruning frequent itemsets and candidate itemsets in order to improve joining efficiency; 3) using overlap strategy to count support to achieve high efficiency. Under the same conditions, the results illustrate that the proposed improved Apriori algorithm improves the operating efficiency compared with other improved algorithms.

INTRODUCTION

Data mining is the discovery of hidden information and interesting patterns in databases. As one of the most important branch of data mining, association rule mining identifies the associations and frequent patterns among a set of items in a given databases. It is composed of two sub-problems: 1) Discover frequent itemset according to some pre-defined threshold; 2) Generate association rules satisfying the confidential constraint.

The first application of association rule mining is buying performance analysis in a supermarket with Apriori algorithm developed by Agrawal and Srikan in 1994 [1]. Since then on, association rule mining not only plays an important role in commercial data analysis but also has a success in finding interesting patterns and associations in many other fields, such as new service opportunity identification [2] and medical data analysis [3].

It worth mentioning that although more and more managers realized that data analysis based on associate rules mining techniques can help them in developing more attracting products or services for clients so as to improve their service quality and get more advantages in competition, the association rule mining algorithms developed to the present suffers either from low efficiency or complication. This study aims at developing an association-rules-mining algorithm that should be both easy to implemented but also high efficient.

The rest of this paper is organized as follows: the related work is presented in section 2. The section 3 is dedicated to introduction of the main contributes of the study about how to improve the efficiency of the algorithm. In section 4, the improved Apriori algorithm called T_Apriori is developed. The experimental results are presented in section 5. The paper ends up with conclusions and perspectives.

RELATED WORK

The basic search strategies of the association-rules-mining can mainly be classified into two categories: Candidate generation and pattern growth [4].

Candidate-Generation-Based (CGB) strategy. It is a Breath-First search strategy. Algorithms with CGB strategy generates candidate (k+1)-itemsets based on frequent k-itemsets. In each transaction, the occurrence of each itemset is counted and recorded as its frequency, which should be compared with a pre-given threshold to decide if this itemset will be kept or not. At the end of the algorithm, frequency of any possible itemset has been determined and all frequent

Advances in Materials, Machinery, Electronics I
AIP Conf. Proc. 1820, 080005-1–080005-6; doi: 10.1063/1.4977361
Published by AIP Publishing. 978-0-7354-1488-4/$30.00

itemsets can be selected. The first algorithm based on CGB strategies is Apriori algorithm proposed by Agrawal in 1994[1].

Pattern-Growth-Based(PGB) strategy. It is a Depth-Frist search strategy using a tree-like data structure, called FP-tree. Algorithms with PGB strategy detects the frequent itemsets by recursively finding all frequent 1-itemset in the conditional pattern base that is efficiently constructed with node link represented by Frequent-Pattern-tree (FP-tree). The first algorithm based on PGB strategy is FP-Growth algorithm developed by Han et al. in 2000[5].

Although the performance of classical Apriori algorithm cannot compete with that of state-of-the-art depth-first approaches[4-7], Apriori algorithm is always regarded as an important association rule mining algorithm because its basic idea of finding all frequent itemsets in a given database is universal and easy to implement for any association rules mining problems though the depth-first approaches suffers not only from the complexity of constructing FP-tree but also from the consumption of storage for recording nodes as well.

According to the literature, besides Agarwal and Srikant themselves [1], many researchers endeavored in improving Apriori algorithm [9-13]. For instance, Park et al. proposed Direct Hashing and Pruning algorithm that utilizes a hash approach to geneate candidate itemsets [9]. Zaki adapted parallel systems to the association rule discovery aimed at taking its advantage of the higher speed and greater storage capacity. The partitioning of the database is used for the transition to a distributed memory system among the processors [10]. Toivonen uses sampling to generate candidate itemsets but still requires a full database scan. Sampling is a powerful data reduction technique that has been applied to a variety of problems in database systems [11]. Dong and Han proposed an algorithm named as BitTableFI, which compresses the database into BitTable, and candidate itemsets generation and support count can be performed quickly with the special data structure [12]. The I-Apriori algorithm, proposed by Bhandari et al. in 2015, is a combination of FP-tree and Apriori algorithm. Zhao The main idea of this study is to reduce the number of transactions to save the time consumed in transactions scanning for candidate itemsets [13]. Zhao proposed frequent itemsets mining algorithm BITXOR based on the efficient bittable, which uses the bit table and represents item sets with binary sequence [14].

This study aims to inherit the advantages of Apriori algorithm, and improve the efficiency of the mining algorithm from following aspects: 1) avoid rescanning the database and 2) reduce the size of candidate itemsets and 3) accelerate both joining and the pruning process.

BASIC CONCEPT

Association rule mining identifies interesting associations, frequent patterns among a set of items in large databases [4]. Formally, Association-Rule-Mining algorithms should fulfill the following task: Let an item be either a product or an event. $X = \{i_1, i_2, \dots, i_n\}$ represents the set of all products/events concerned in this problem. Let $T = \{t_1, t_2, \dots, t_m\}$ be a set of transactions, where each transaction is a set of items. Let itemset refer to any subset of the item base X, The association rule, noted as $X => Y$, indicates a certain relation between two itemsets X and Y. An association rule $X => Y$ is supported if the percentage of transactions that contain both itemset X and Y in T exceeds a certain threshold, called support threshold. Furthermore, the confidence for the association rule $X => Y$ is defined by the percentage of transactions that contain itemset Y among transactions containing itemset X [15].

Apriori algorithm explores candidates in two steps: (a) Generate all frequent itemstets. An itemset is frequent when its occurrence exceeds the minimum support, a pre-given threshold; (b) Suppose all frequent k-itemset have been explored, create (k+1)-itemset based on k-itemset and keep just frequent (k+1)-itemset, in otherwords, i.e. a priori pruning operation is taken for excluding all infrequent (k+1)-itemsets.

The main contributions of this study are: 1) proposing a new searching strategy for accelerating the searching process; 2) Reducing the storage cost by using compacted vector structure.

IMPROVED APRIORI ALGORITHM

Description of the Improved Apriori Algorithm

In order to avoid scanning the database repeatedly, the paper proposed a new method depicted as below:

1. Only scan transaction database once, get the transaction identifier (TID) set for each item;
2. Before the candidate itemsets C_k come out, further prune L_{k-1}: count the times of all items occurred in L_{k-1}, delete item sets with this number less than $k-1$ [16].

3. Using overlap strategy to count the support of candidate itemsets C_k based on the TID set of L_{k-1} and TID set of L_1;
4. Terminate the algorithm, if the number of $|L_k| \leq k$[17].

It is obvious that with the proposed algorithm described above, not only the number of candidate items will decline. but also the time used to count the support will be greatly reduced.

Pseudo-code of T_Apriori Algorithm

T_Apriori algorithm explores frequent itemset as follows:
//Generate items, items support, transaction IDs
1) L_1=find_frequent_1_itemsets(T);
//Construct $C_2$by self-join
2) $C_2 = L_1 \infty L_1$
3) $L_2 = items\ in\ C_2 \geq \min\_sup$;
4) For (k=3; $L_{k-1} \neq \emptyset; K++$){
//further prune L_{k-1}
5) $Prunel(L_{k-1})$;
6) $Lx \in L_k, Ly \in L_k$;
7) $if\ (Lx[1] = Ly[1]\ \wedge\ Lx[2] = Ly[2]\ \wedge \ldots Lx[k-2] = Ly[k-2] \wedge\ Lx[k-1] < Ly[k-1])$
8) $c = Lx \cup Ly$;
9) $if (k-1) - subsets\ s\ of\ c \notin L_{k-1}$
10) $then\ delete\ c\ from\ Ck$;
11) $Ck = c \cup Ck$;
//Intersection between the target transaction TID set to calculate the support
12) $Lk = New\_quick\_support\_count(Ck, TID\_Set)$;
13) $Answer = \cup_k L_k$;
$New\_quick\_support\_count(Ck, TID\_Set)$;
$\{forall\ itemsets\ C \in C_k$
$C.TID\_Set = L_{k-1}.TID\_Set \cap L_1.TID\_Set$
$C.Sup = Length(Ck.TID\_Set)$;
$if\ C.Sup < \min\_sup$
$delete\ C\ from\ Ck$;
$Lk = \{C \in C_k | C.Sup \geq \min\_sup\}\}$
$Prunel(L_k)$
for all itemsets $L_1 \in L_k$;
$if\ count(L_1)\ in\ L_k \leq k$;
$then\ delete\ all\ L_j\ from\ L_k$;
$return\ L'_k$;

Illustration of T_Apriori Algorithm with A Simple Example

For a better understand, in this section, a simple example, with 10 transactions and 6 items as shown in Table 1, is used for illustrating the procedure of T_Apriori algorithm. Minimum support level is set as 2.

Step 1: Browse through all transaction to identify for each 1-itemset the corresponding list of transactions, i.e. transactions containing the corresponding item. Calculate the number of TIDs for each item and eliminate all 1-itemsets whose support level is less than the minimum, and $L_1 = \{\{I_1\}, \{I_2\}, \{I_3\}, \{I_4\}, \{I_5\}\}$.

Step 2: join L_1 with L_1 to generate candidate 2-itemsets, using overlap strategy to count the support of candidate itemsets and eliminate all 2-itemsets whose support level is less than the minimum, and $L_2 = \{\{I_1I_2\}, \{I_1I_2\}, \{I_2I_3\}, \{I_3I_4\}, \{I_3I_5\}\}$.

Step 3: Generate frequent k-item sets (here is 3-item sets).

1.1. Before the candidate itemsets C_k (here is C_3) come out, further prune L_{k-1} (here is L_2), count the times of all items occurred in L_{k-1} (here is L_2), delete item sets with this number less than $k-1$ (here is 2). In L_2, the times of I_4 and I_5 is 1 that is less than 2. So we delete the 2-itemsets containing I_4 and I_5, and get

$L'_2 = \{\{I_1I_2\}, \{I_1I_2\}, \{I_2I_3\}\}$.

1.2. Join $L_{k-1}(L'_2)$ with L_{k-1} L'_2 when their prior $k-2$ elements are same. Next, delete all itemsets if their (k-1)-subset of c are not in $L_{k-1}(L_2)$. Then get the candidate 3-itemsets $C_3 = \{I_1I_2I_3\}$.

1.3. Using overlap strategy to count the support of k-candidate itemsets (here C_3) based on L_k and L_1 (L_2 and L_1), the TIDs of I_1I_2 is {T1,T3,T4,T5,T6} and TIDs of L_3 is {T2,T4,T5,T6,T7,T8,T10} , so after intersection, the result is { T4,T5,T6}, so the support is 3 for candidate 3-itemset $I_1I_2I_3$, and delete candidate itemsets if support level less than the minimum and finally get frequent k-candidate itemsets (here is $L_3 = I_1I_2I_3$).

Step 4: Calculate the number of itemsets in $L_3(L_k)$, if the number is less than or equal to 3 ($|L_k| \leq k$), the algorithm terminates and returns all frequent items. If the number of $L_3(L_k)$ is larger than 3 ($|L_k| > k$), repeat from step 3 and step 4. Here the number of item sets in L_3 is only 1, so the algorithm terminates.

TABLE 1. The transactions

TID	Itemsets	TID	Itemsets
T1	I1,I2	T6	I1,I3,I5
T2	I2,I3	T7	I2,I3
T3	I1,I2	T8	I3,I4
T4	I1,I2,I3,I4	T9	I4,I6
T5	I1,I2,I3	T10	I3,I5

EXPERIMENTAL RESULTS

In order to evaluate the performance of T_Apriori algorithm, numerical experiments are executed and results are compared with not only the original Apriori Algorithm, noted as Apriori, proposed by Agrawal and Srikant[1] but also the mining algorithm proposed by Bhandari[13], noted as I_Apriori as well as the BITXOR algorithm proposed by Zhao [14]. BITXOR algorithm is selected as a benchmark in this study because it represents the most recent achievement in comparison with FP-Growth and Apriori. All experimental data are extracted from the standard data warehouse et set a standard experimental environment.

Experiments were conducted on a MacBook Pro, OS X Yosemite system equipped with processor of 2.2GHz Intel Core i7 and memory of 16GB 1600 MHz DDR3.

In order to test the Evaluation of Efficiency of T_Apriori algorithm, Mushroom datasets, extracted from the UCI repository, are tested with various minimum support thresholds. As shown in Fig.1, T_Apriori algorithm needs much less execution time to get the final solution than the other two algorithms in all experiments. Moreover, BITXOR algorithm performs better than FP-growth algorithm [14] (see Fig.2), so T_Apriori algorithm is better than FP-growth algorithm. In one word, the experimental results are quite encouraging because T_Apriori demonstrates the best performance among three algorithms different values of minimum support.

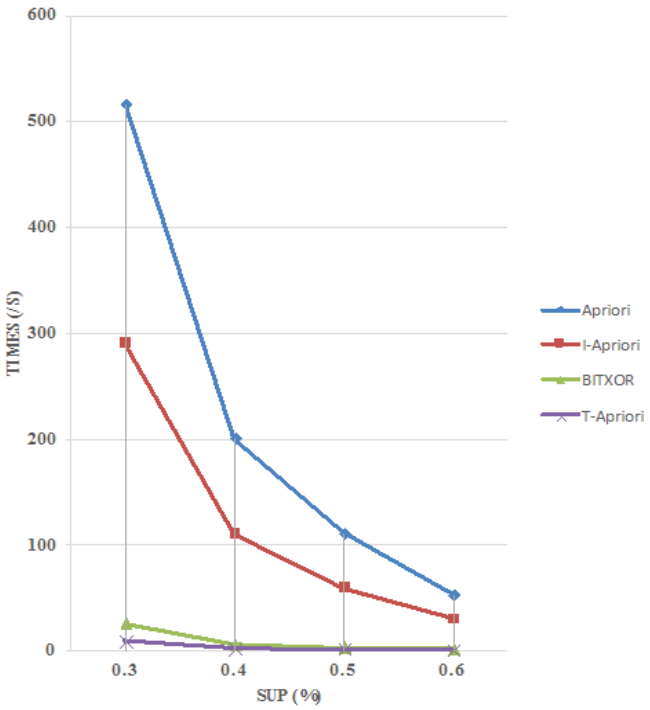

Figure 1. Time consuming comparison for different values of minimum support in the mushroom database

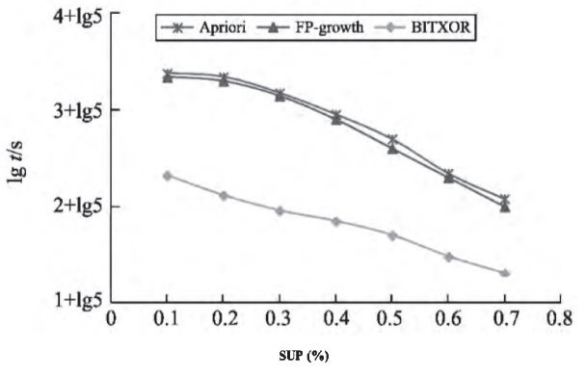

FIGURE 2. The comparison of performance of three algorithms in the Mushroom database

CONCLUSIONS AND PERSPECTIVES

The improved Apriori algorithm achieves excellent performance by reducing the time consumed in transaction scanning for the generation of candidate itemsets and by reducing the number of transaction to be scanned. Generally, from view of time consumed, the T_Apriori performs much better as the group of transaction number increases and the value of minimum support increases. It reduced the time consumption by over 98% in average. Therefore, our improved Apriori algorithm is far more efficient than the original Apriori algorithm. In the future, this improved algorithm can be further improved from the perspective of space complexity. In application, it can be applied in healthcare processes to analyze relevant medical processes and identify the effective ways to improve the efficiency of medical procedures and services, in order to save medical costs and alleviate the hospital pressure for patients overcrowding, as well as improve patients' satisfaction.

ACKNOWLEDGMENTS

This work was financially supported by Shanghai Pujiang Program, China (No. 13PJC061).

REFERENCES

1. Agrawal R, Srikant R. *Fast Algorithms for Mining Association Rules in Large Databases* (International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc, 1994), pp.487-499.
2. Karimi-Majd A M, Mahootchi M. *A new data mining methodology for generating new service ideas* (Information Systems and e-Business Management, 2015, 13(3)), pp.421-443.
3. Wang J, Li H, Huang J, et al. *Association rules mining based analysis of consequential alarm sequences in chemical processes* (Journal of Loss Prevention in the Process Industries, 2016(41)),pp.178-185.
4. Borgelt C. *Frequent item set mining* (Wiley Interdisciplinary Reviews Data Mining & Knowledge Discovery, 2012, 2(6)), pp.437–456.
5. Han J, Pei J, Yin Y. *Mining frequent patterns without candidate generation* (Acm Sigmod Record, 2000, 29(2)), pp.1-12.
6. Bhaskar R, Laxman S, Smith A, et al. *Discovering frequent patterns in sensitive data* (ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, Dc, USA, July 2010), pp.503-512.
7. Vo B, Chi M, Minh H C. *Fast Algorithm for Mining Generalized Association Rules* (International Journal of Database Theory & Application, 1994, 2(12)), pp.161–180.
8. Rao S., Gupta R., *Implementing Improved Algorithm Over APRIORI Data Mining Association Rule Algorithm* (International Journal of Computer Science and Technology, 2012), pp. 489-493.
9. Park J.S., Chen M.S., Yu P.S. *Using a Hash-Based Method with Transaction Trimming and Database Scan Reduction for Mining Association Rules* (IEEE Transactions on Knowledge & Data Engineering, 1997,9(5)), pp.813-825.
10. Zaki M.J., *Parallel and distributed association mining: A survey* (IEEE Concurrency, Special Issue on Parallel Mechanisms for Data Mining, 1999, 7(4)), pp.14-25.
11. Toivonen H., *Sampling Large Databases for Association Rules* (Proc Vldb, 2000), pp.134-145.
12. Dong J., Han M., *BitTableFI: An efficient mining frequent itemsets algorithm* (Knowledge-Based Systems, 2007, 20(4)), pp.329-335.
13. Bhandari A., Gupta A., Das D., *Improvised Apriori Algorithm using frequent pattern tree for real time applications in data mining* (Procedia Computer Science,2015(46)), pp.644–651.
14. Zhao BG., Liu Y., *An Efficient Bittable Based Frequent Itemsets Mining Algorithm* (Journal of Shandong University, 2015(5)), pp.23-29.
15. Lazcorreta E., Botella F., Fernández-Caballero A. *Towards personalized recommendation by two-step modified Apriori data mining algorithm* (Expert Systems with Applications, 2008, 35(3)), pp.1422-1429.
16. Jiao Y. *Research of an Improved Apriori Algorithm in Data Mining Association Rules* (International Journal of Computer & Communication Engineering, 2013, 2(1)), pp.25-27.
17. Fu S., Zhou HJ., *The Research and Improvement of Apriori Algorithm for Mining Association Rules* (Microelectronics & Computer, 2013(9)), pp.110-114.