Third 2008 International Conference on Convergence and Hybrid Information Technology

An Efficient Frequent Patterns Mining Algorithm based on Apriori Algorithm and the FP-tree Structure

Bo Wu, Defu Zhang, Qihua Lan, Jiemin Zheng
Department of Computer Science, Xiamen University, Xiamen 361005, China
Longtop Group Post-doctoral Research Center, Xiamen, 361005, China
dfzhang@xmu.edu.cn

Abstract

Association rule mining is to find association relationships among large data sets. Mining frequent patterns is an important aspect in association rule mining. In this paper, an efficient algorithm named Apriori-Growth based on Apriori algorithm and the FP-tree structure is presented to mine frequent patterns. The advantage of the Apriori-Growth algorithm is that it doesn't need to generate conditional pattern bases and sub- conditional pattern tree recursively. Computational results show the Apriori-Growth algorithm performs faster than Apriori algorithm, and it is almost as fast as FP-Growth, but it needs smaller memory.

1. Introduction

Data mining has recently attracted considerable attention from database practitioners and researchers because it has been applied to many fields such as market strategy, financial forecasts and decision support [1]. Many algorithms have been proposed to obtain useful and invaluable information from huge databases [2]. One of the most important algorithms is mining association rules, which was first introduced in [3, 4].

Association rule mining has many important applications in our life. An association rule is of the form X => Y. And each rule has two measurements: support and confidence. The association rule mining problem is to find rules that satisfy user-specified minimum support and minimum confidence. It mainly includes two steps: first, find all frequent patterns; second, generate association rules through frequent patterns.

Many algorithms for mining association rules from transactions database have been proposed [5, 6, 7] since Apriori algorithm was first presented. However, most algorithms were based on Apriori algorithm which generated and tested candidate itemsets iteratively. This may scan database many times, so the computational cost is high.

In order to overcome the disadvantages of Apriori algorithm and efficiently mine association rules without generating candidate itemsets, a frequent-pattern-tree (FP-Growth) structure is proposed in [9]. The FP-Growth was used to compress a database into a tree structure which shows a better performance than Apriori. However, FP-Growth consumes more memory and performs badly with long pattern data sets. In order to further improve FP-Growth algorithm, many authors developed some improved algorithms and obtained some promising results [10, 11, 12, 13].

Due to Apriori algorithm and FP-Growth algorithm belong to batch mining. What is more, their minimum support is often predefined, it is very difficult to meet the applications of the real-world. Recently, there are some growing interests in developing techniques for mining association patterns without a support constraint or with variable supports [14, 15, 16]. Association rule mining among rare items is also discussed in [17,18]. So far, there are very few papers that discuss how to combine Apriori algorithm and FP-Growth to mine association rules. In this paper, an efficient algorithm named Apriori-Growth based on Apriori algorithm and FP-Growth algorithm is proposed, this algorithm can efficiently combine the advantages of Apriori algorithm and FP-Growth algorithm. Computational results verify the good performance of the Apriori-Growth algorithm.

The organization of this paper is as follows. In Section 2, we will briefly review the Apriori method and FP-Growth method. Section 3 proposes an efficient Apriori-Growth algorithm that based on Apriori and the FP-tree structure. Experimental results will be presented in Section 4. Section 5 gives out the conclusions.

978-0-7695-3407-7/08 $25.00 © 2008 IEEE
DOI 10.1109/ICCIT.2008.109

2. Two Classcical Mining Algorithms

2.1 Apriori Algorithm

In [4], Agrawal proposed an algorithm called Apriori to the problem of mining association rules first. Apriori algorithm is a bottm-up, breadth-first approach. The frequent itemsets are extended one item at a time. Its main idea is to generate *k*-th candidate itemsets from the (*k*-1)-th frequent itemsets and to find the *k*-th frequent itemsets from the *k*-th candidate itemsets. The algorithm terminates when frequent itemsets can not be extended any more. But it has to generate a large amount of candidate itemsets and scans the data set as many times as the length of the longest frequent itemsets. Apriori algorithm can be written by pseudocode as follows.

Procedure Apriori

Input: data set D, minimum support minsup

Output: frequent itemsets *L*

(1) L_1 = find_frequent_1_itemsets(D);

(2) for ($k = 2$; $L_{k-1} \neq \phi$; k++)

(3) {

(4) C_k = Apriori_gen(L_{k-1}, minsup);

(5) for each transactions $t \in D$

(6) {

(7) C_t = subset(C_k, t);

(8) for each candidate $c \in C_t$

(9) c.count++;

(10) }

(11) $L_k = \{c \in C_k \mid c.\text{count} > \text{minsup}\}$;

(12) }

(13) return $L = \{L_1 \cup L_2 \cup ... \cup L_n\}$;

In the above pseudocode, C_k means *k*-th candidate itemsets and L_k means *k*-th frequent itemsets.

2.2 FP-Growth Algorithm

In [9], Han, Pei et al. proprosed a data structure called FP-tree (frequent pattern tree). FP-tree is a highly compact representation of all relevant frequency information in the data set. Every path of FP-tree represents a frequent itemset and the nodes in the path are stored in decreasing order of the frequency of the corresponding items. A great advantage of FP-tree is that overlapping itemsets share the same prefix path. So the information of the data set is greatly compressed. It only needs to scan the data set twice and no candidate itemsets are required.

An FP-tree has a header table. The nodes in the header table link to the same nodes in its FP-tree. Single items and their counts are stored in the header table by decreasing order of their counts. Fig.1a shows an example of a data set while Fig.1b shows the FP-tree constructed by that data set with minsup = 30%.

Transaction
a b c
a c e f
d f
a b c
a c e g
b c

Fig.1a. A data set

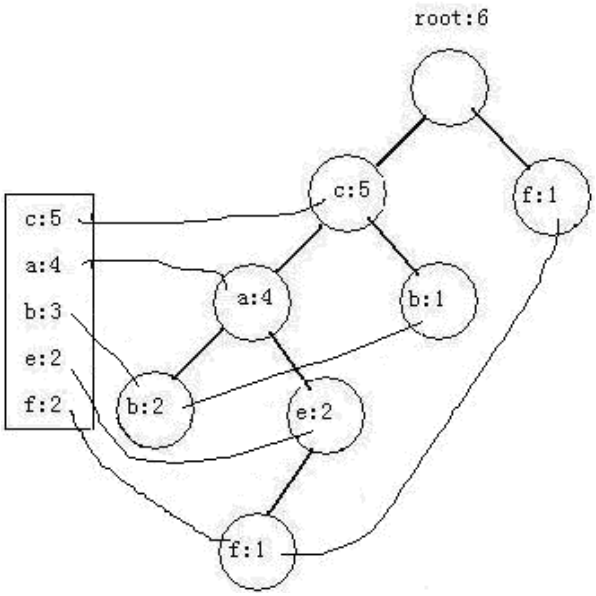

Fig.1b. FP-tree Constructed by the above data set

The disadvantage of FP-Growth is that it needs to work out conditional pattern bases and build conditional FP-tree recursively. It performs badly in data sets of long patterns.

3. Apriori-Growth Algorithm

In this Section, a new algorithm based on Apriori and the FP-tree structure is presented, which is called Apriori-Growth.

Fig.2a shows the data structure of the node of header table. Its tablelink points to the first node in FP-tree which has the same name with it. And Fig.2b shows the data structure of the node of FP-tree. Its tablelink points to the next node in FP-tree which has the same name with it.

Name
*tablelink
Count

Fig.2a the data structure of the node of header table

Name
*tablelink
*parent
*child
Count

Fig.2b the data structure of the node of FP-tree

The Apriori-Growth mainly includes two steps.

First, the data set is scanned one time to find out the frequent 1 itemsets, and then the data set is scanned again to build an FP-tree as [9] do.

At last, the built FP-tree is mined by Apriori-Growth instead of FP-Growth. The detailed Apriori-Growth algorithm is as follows.

Procedure Apriori-Growth
Input: data set D, minimum support minsup
Output: frequent itemsets L

(1) L_1 = frequent 1 itemsets;
(2) for($k = 2$; $L_{k-1} \neq \phi$; k++)
(3) {
(4) C_k = Apriori_gen(L_{k-1}, minsup);
(5) for each candidate $c \in C_t$
(6) {
(7) sup = FP-treeCalculate(c);
(8) if (sup > minsup)
(9) $L_k = L_k \cup c$;
(10) }
(11) }
(12) return $L = \{ L_1 \cup L_2 \cup ... \cup L_n \}$;

Procedure FP-treeCalculate
Input: candidate itemset c
Output: the support of candidate itemset c

(1) sort the items of c by the decreasing order of header table;
(2) find the node p in the header table which has the same name with the first item of c;
(3) $q = p$.tablelink;
(4) count = 0;
(5) while q is not null
(6) {
(7) if the items of the itemset c except first item all appear in the prefix path of q
(8) count += q.count;
(9) $q = q$.tablelink;
(10) }
(11) return count / totalrecord;

As we know, the item of bigger count must be the ancestor of the smaller one if several items appear together in a path. So the count of the path is equal to the count of the node which is closest to leaf. So to find the count of candidate itemsets is to find the sum of those nodes.

In this case, we can work out the support of candidate itemsets by traversing related nodes in FP-tree and their prefix paths.

4. Experimental Results

The content of our test data set are bank card transactions seized from bank of China. There are 27 different items and 50905 records in that data set.

In order to verify the performance of the Apriori-Growth algorithm, we compare Apriori-Growth with Apriori and FP-Growth. Three algorithms are performed on a computer with a 1.41GHz processor and 512MB memory. The program is developed by Visual C++ 6.0. The computational results of three algorithms are reported in Table 1.

The clearer comparison of three algorithms is given in Fig.3.

Table 1. The running time of three algorithms

Min_sup	Apriori	Apriori-Growth	FP-Growth
10%	766ms	391ms	407ms
5%	1515ms	406ms	437ms
1%	6953ms	438ms	561ms
0.5%	13125ms	484ms	688ms
0.2%	31672ms	687ms	874ms
0.1%	60547ms	1093ms	1078ms

From Fig.3, we can make the following two statements. First, Apriori-Growth works much faster than Apriori. It uses a different method FP-treeCalcualte to calculate the support of candidate itemsets. Second, Apriori-Growth works almost as fast

as FP-Growth. But it consumes less memory than FP-Growth because it doesn't need to generate conditional pattern bases and build sub-conditional pattern tree recursively.

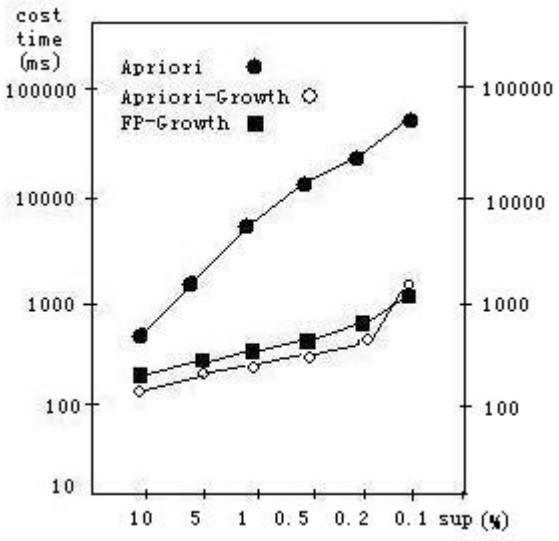

Fig.3

5. Conclusions and future work

In this paper, we have proposed the Apriori-Growth algorithm. This method only scans the data set twice and builds FP-tree once while it still needs to generate candidate itemsets.

The future work is to further improve the Apriori-Growth and test more and larger datasets.

Acknowledgments

This work was supported by the National Nature Science Foundation of China (Grant no. 60773126) and the Province Nature Science Foundation of Fujian (Grant no. A0710023) and academician start-up fund (Grant No. X01109) and 985 information technology fund (Grant No. 0000-X07204) in Xiamen University.

Reference

[1] M.S. Chen, J. Han, P.S. Yu, "Data mining: an overview from a database perspective", *IEEE Transactions on Knowledge and Data Engineering*, 1996, 8, pp. 866-883.

[2] J. Han, M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publisher, San Francisco, CA, USA, 2001.

[3]R.Agrawal, T.Imielinski and A.Swami, "Mining association rules between sets of items in large databases," in: *Proceedings of the Association for Computing Machinery, ACM-SIGMOD*, 1993, 5, pp.207-216.

[4] R. Agrawal, R. Srikant, "Fast algorithms for mining association rules", *Proceedings of the 20th Very Large DataBases Conference (VLDB'94)*, Santiago de Chile, Chile, 1994, pp. 487-499.

[5]Agrawal, R., Srikant, R., & Vu, Q, "Mining association rules with item constraints", *In The third international conference on knowledge discovery in databases and data mining*, Newport Beach, California, 1997, pp. 67-73.

[6]J.Han, Y. Fu, "Discovery of multiple-level association rules from large database", In *The twenty-first international conference on very large data bases*, Zurich, Switzerland, 1995, pp. 420-431.

[7]Fukuda, T., Morimoto, Y., Morishita, S., & Tokuyama, T., "Mining optimized association rules for numeric attributes", In *The ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems*, 1996, pp. 182-191.

[8]Park, J. S., Chen, M. S., & Yu, P. S., "Using a hash-based method with transaction trimming for mining association rules", *IEEE Transactions on Knowledge and Data Engineering*, 1997, 9(5), pp. 812-825.

[9]J.Han, J.Pei and Y.Yin., "Mining frequent patterns without candidate Generation", in: *Proceeding of ACM SIGMOD International Conference Management of Data*, 2000, pp. 1-12.

[10] J.Han, J.Wang, Y.Lu and P.Tzvetkov, "Mining top-k frequent closed patterns without minimum support", in: *Preceeding of International Conference Data Mining*, 2002,12, pp. 211-218.

[11]G.Liu, H.Lu, J.X.Yu, W.Wei and X.Xiao, "AFOPT: An efficient implementation of pattern growth approach", in: *IEEE ICDM Workshop Frequent Itemset Mining Implementations*, CEUR Workshop Proc., 2003, 80.

[12] J.Wang, J.Han, and J.Pei, "CLOSET+: searching for the best strategies for mining frequent closed Itemsets", in: *Preceeding of International Conference, Knowledge Discovery and Data Mining*, 2003, 8, pp. 236-245.

[13]Tzung-Pei Hong, Chun-Wei Lin, Yu-Lung Wu, "Incrementally fast updated frequent pattern trees", *Expert Systems with Applications*, 2008, 34, pp. 2424-2435.

[14] K. Wang, Y. He, D. Cheung, Y. Chin, "Mining confident rules without support requirement", in: *Proceedings of ACM International Conference on Information and Knowledge Management*, CIKM, 2001, pp. 89-96.

[15] H. Xiong, P. Tan, V. Kumar, "Mining strong affinity association patterns in data sets with skewed support distribution", in: *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM, 2003, pp. 387-394.

[16]Ya-Han Hu, Yen-Liang Chen, "Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism", *Decision Support Systems*, 2006, 42, pp. 1-24.

[17] J. Ding, "Efficient association rule mining among infrequent items", *Ph.D. Thesis*, University of Illinois at Chicago, 2005.

[18] Ling Zhou, Stephen Yau, "Efficient association rule mining among both frequent and infrequent items", *Computers and Mathematics with Applications*, 2007, 54, pp. 737-749.