Study of Incremental Updating Algorithm for Association Rules

An Hongmei, Chen Ping
Technology Department Maanshan Teachers' School
Maanshan, 243000, China

Huang Lijing
School of Information Science & Engineering
Hebei University of Science and Technology
Shijiazhuang, 050018, China

***Abstract*—Discover frequent itemsets is the key problem of mining association rules, and the expenditure of this process is high. So, the research of incremental updating algorithms is particular important. This paper presents a comprehensive study on the performance of incremental updating algorithms that currently available, such as IUA [1] which makes full use of frequent itemsets already got. Based on the shortcomings of IUA, presents a new algorithm AIUA as an improved algorithm of IUA.**

Keywords-Data mining, Association rules, Incremental updating

I. INTRODUCTION

Data mining, also known as knowledge discovery from database （KDD） [2], is an intelligent data analysis technique in the 1980's then gradually developed. It is a procedure to obtain valid, new, potentially useful and ultimately comprehensible information and knowledge from a large number of data that stored in a database, data warehouse or other repository.

As one of the many ways that data mining uses, association rules mining is first proposed by Agrawal in 1993. Its purpose is to find valuable knowledge that describes the affiliation between data items from a large number of data and its successful application in business and so on makes it the most mature, important, active research of data mining. But, a lot of work has focused on mining algorithms until the end of last century, and the database that association rules used is extremely large, so not only designing of efficient algorithms for mining association rules but also the update, maintenance and management of the association rules which have been found are both important. Thus, since then many scholars have done a lot in the update of association rules.

This paper considered the update when the minimum_support(min_sup) and minimum confidence(min_conf) changed, pointed out the shortcomings of IUA, and upon the shortcomings of IUA, presented AIUA. AIUA improves the efficiency of mining in ensuring the validity.

II. ASSOCIATION RULES MINING

Formal description of association rules is as follows:

Let $I = \{i_1, i_2, ..., i_m\}$, this is a data items set, D is a data set that associated with task, namely a transaction database, each transaction T is a subset of data items set, namely $T \subseteq I$; each transaction contains an identification code TID. Let A is a data items set, if and only if $A \subseteq T$, we said transaction T contains A. An association rule is an implication that in the form of "$A \Rightarrow B$", where $A \subset I$, $B \subset I$ and $A \cap B = \Phi$. The rule $A \Rightarrow B$ is true in the transaction database D, and with s support and c confidence, means that in the transaction database D has s ratio of the transaction T contains $A \cup B$ data items, and has c ratio of the transaction T to meet "if contains A then contains B". Concrete description is as follows:

$$support(A \Rightarrow B) = P(A \cup B)$$
$$confidence(A \Rightarrow B) = P(B|A)$$

Rules that meet the given min_sup and min_conf are called strong association rules. Mining association rules is to find out all strong rules from transaction database.

The two steps of mining association rules are as follows:

Step 1: Screen out all itemsets (nonempty subset of data itemsets I) with user-specified min_sup from transaction database D. The itemsets with the min_sup are called frequent itemsets, otherwise known as non-frequent itemsets.

Step 2: Using frequent itemsets to generate all association rules. For each frequent itemset A, find out all nonempty subset of A as a, if the ratio of support (A) / support (a)> min_conf, then generate association rule $a \Rightarrow$ (A-a),in which min_conf is the user-specified minimum confidence.

III. INCREMENTAL UPDATING ALGORITHM FOR ASSOCIATION RULES

The problem of updating association rules can be divided into two categories: first, given a transaction database D, keepping the min_sup unchanged, when a new transaction database d added to D, such as literature [3], [4], [5] and [6]; second, the transaction database D remains unchanged, while the min_sup and min_con changed, such as literature [1]. At present, there are many mature algorithms available, these algorithms take advantage of all the original information to efficiently find out all the new association rules or frequent itemsets, and effectively solve the above problems.

In the process of mining association rules, in order to discover the unknown and of potentially useful association rules in advance, users need to constantly adjust the min_sup and min_conf to gradually focus to those association rules that they really interested, this will be a dynamic, interactive

Published by Atlantis Press, Paris, France.

process. Therefore, we urgent need an efficient updating algorithm to meet user's demand of faster response time.

This paper and literature [1] consider the second. Due to limited space, this paper is no longer pay attention to the detailed description of IUA, please see [1], and the symbols used in this paper are in the same meaning except where extremely noted.

A. The shortcomings of IUA

In the join phase of generating candidate itemsets, IUA presents a new candidate k-itemsets generation function named iua-gen(L^1_j) to generate C^3_k.

Throw careful study you will find that some large itemsets will be deleted by mistake, because the subsets of k-1 length that included in L^3_k must be large itemsets, but not necessarily include in L^3_{k-1}, they may also included in L^1_{k-1}, or L^2_{k-1}. For example: a case in point is that in L^3_k, only one item is included in L_1, the remaining k-1 items are included in L''_1, its k-1 subsets include one kind that all the items are included in L''_1, means that they are included in L^2_{k-1}. Similarly, there is also one kind that all the items are included in L_1. To solve this problem, this paper presents improved algorithm AIUA.

B. AIUA

For C^1_k and C^2_k, from the beginning loop k = 2, that is k ≥ 2, we directly use function apriori_gen of Apriori, C^1_k generated by self-join of L^1_{k-1}; C^2_k generated by self-join of L^2_{k-1}.

C^1_k = apriori_ gen (L^1_{k-1}) -L_k;

C^2_k = apriori_ gen (L^2_{k-1}) ;

Before generating C^3_k, we divide it into three categories: C^{31}_k, C^{32}_k and C^{33}_k. C^{31}_k contains all k-itemsets that consists of one item included in L_1 and k-1 items included in L''_1; C^{33}_k contains all k-itemsets that consists of k-1 items included in L_1 and one item included in L''_1; C^{32}_k contains all other candidate k-itemsets. According to the definition above we know that C^{31}_k generated by connection of L^2_{k-1} and L^{31}_{k-1}; C^{33}_k generated by connection of L^1_{k-1} and L^{33}_{k-1}; C^{32}_k generated by self-connection of L^3_{k-1}. Accordingly L^3_k is also divided into three categories, L^{31}_k, L^{32}_k and L^{33}_k.

For generation of C^3_k, when k=2 and k=3, it does not have the classification as described above, so raise them alone, the beginning loop from k = 4.

(1) k=2 C^3_2 generated by join of L1 and L''_1, via function aiua_gen2(L_1,L''_1):

①join:

insert into C^3_2
select p.item1, q.item1
from L1p, L''_1q

②pruning:

for all itemsets c∈C^3_2 do
  for all 1-items of c do
    if (s∉ L'_1) then
      delete c from C^3_2;

(2) k=3 divides C^3_3 into two categories: C^{31}_3 — generated by join of L^2_2 and L^2_3 (get L^{31}_3 by pruning), C^{33}_3 — generated by join of L^1_2 and L^2_3 (get L^{33}_3 by pruning), via function aiua_gen3(L^1_2,L^2_2,L^3_2):

①join:

insert into C^{31}_3
select p.item1,p.item2,q.item2
from L^2_2 p, L^3_2 q
where p.item1=q.item1

insert into C^{33}_3
select p.item1,p.item2,q.item2
from L^1_2 p, L^3_2 q
where p.item1=q.item1

②pruning:

for all itemsets c∈C^3_3 do
  for all 2-items of c do
    if (s∉ L'_2) then
      delete c from C^3_3;

(3) k=4 loop start

This paper presents the improved function aiua_gen(L^1_{k-1}, L^2_{k-1},L^3_{k-1},L^{31}_{k-1},L^{32}_{k-1},L^{33}_{k-1}) of iua_gen(L^1_j), including the following two steps:

①join:

insert into C^{31}_k

select p. item1, p. item2,. . . , p. item(k-2), p. item(k-1), q. item(k-1)

from L^{2}_{k-1} p, L^{31}_{k-1} q where p. item1 = q. item1, p. item2 = q. item2, . . . , p.item(k-2)= q.item(k-2)

insert into C^{33}_{k}

select p. item1, p. item2,. . . , p. item(k-2), p. item(k-1), q. item(k-1)

from L^{1}_{k-1} p, L^{33}_{k-1} q where p. item1 = q. item1, p. item2 = q. item2,. . . , p.item(k-2)= q.item(k-2)

insert into C^{32}_{k}

select p. item1, p. item2,. . . , p. item(k-2), p. item(k-1), q. item(k-1)

from L^{3}_{k-1} p, L^{3}_{k-1} q where p. item1 = q. item1, p. item2 = q. item2,. . . , p.item(k-2)= q.item(k-2)

②pruning:

for all itemsets $c \in C^{3}_{k}$ do

for all (k-1) -subsets s of c do

if ($s \notin L'_{k-1}$) then

delete c from C^{3}_{k};

The correctness of AIUA can be proved by the previous definition and discussion, its analysis process is the same as the proof of Apriori, no further explanation.

Description of AIUA:

Input:

(1)transaction database D;

(2)old support s, new support s′;

(3)candidate itemsets L k with s.

Output: candidate itemsets with s′.

Begin:

1) $L^{1} = \{\text{new frequent 1-itemsets}\}$; $L''^{1} = L'^{1} \cup L1$;

2) for (k= 2;$L'_{k-1} \neq \Phi$; k++) do begin

3) $C^{1}_{k} = \text{apriori\_ gen } (L^{1}_{k-1})$ -Lk;

4) $C^{2}_{k} = \text{apriori \_gen } (L^{2}_{k-1})$;

5) $C^{3}_{k} = \Phi$;

6) if (k==2) then

7) $C^{3}_{k} = \text{aiua \_ gen2 } (L1, L''^{1})$;

8) else if (k==3) then

9) $C^{3}_{k} = \text{aiua \_ gen3 } (L^{1}_{2}, L^{2}_{2}, L^{3}_{2})$;

10) else $C^{3}_{k} = \text{aiua\_ gen } (L^{1}_{k-1}, L^{2}_{k-1}, L^{3}_{k-1}, L^{31}_{k-1}, L^{32}_{k-1}, L^{33}_{k-1})$;

11) for all transactions t∈D do begin

12) Ct1= subset (C^{1}_{k} , t) ;

13) for all candidates c∈Ct1 do

14) c.count++ ;

15) Ct2= subset (C^{2}_{k} , t) ;

16) for all candidates c∈Ct2 do

17) c.count++ ;

18) Ct3= subset (C^{3}_{k} , t) ;

19) for all candidates c∈Ct3 do

20) c.count++ ;

21) end

22) $L^{0}_{k} = \{c \in C^{1}_{k} | c.count \geqslant s'\}$; $L^{1}_{k} = L^{0}_{k} \cup Lk$;

23) $L^{2}_{k} = \{c \in C^{2}_{k} | c.count \geqslant s'\}$;

24) $L^{3}_{k} = \{c \in C^{3}_{k} | c.count \geqslant s'\}$;

25) $L'_{k} = L^{1}_{k} \cup L^{2}_{k} \cup L^{3}_{k}$;

26) end

27) Answer= $\cup_{k} L'_{k}$.

IV. PERFORMANCE ANALYSIS

A. Compare to Apriori

The simplest and intuitionistic approach for updating association rules is to run Apriori once again in the original database with new support, the following will compare AIUA to rerunning Apriori once again. Apriori first scans the database once to count all the candidate 1-itemsets, but all of the frequent 1-itemsets with the original support are removed from AIUA, and the amount of itemsets that included in L_1 is large, so this will greatly reduce the number of itemsets which need to calculate the support. In addition, during the subsequent course of scanning the database, divide C_k into three categories, and remove L_k from C^{1}_{k}, these also reduce the number of candidate items. Reducing the number of candidates can save computing time and storage space that required.

B. Compare to IUA

Although IUA reduced the generation of candidate itemsets to a great extent, but when it generates C^{3}_{k}, join a frequent j-itemset ($1 \leq j \leq k-1$) of ① to a frequent (k-j)-itemset of ②, this also generates a large number of useless candidates, then AIUA does well, it not only reduces the candidate itemsets, but also makes connections only when the conditions meet the connection condition, so it greatly reduces the number of itemsets that need test. The efficiency will be greatly improved when the number of frequent k-itemsets is large.

In addition, in the case that the change of min_sup is smaller, the number of L''_{1} is small than the number of L_1, so the efficiency of AIUA will be more significant than the original algorithm.

V. SUMMARIES

This paper studyed the second type of association rules updating problem, presented an improved efficient algorithm AIUA. Moreover, since AIUA divided C_k^3 into three categories, multiprocessor can achieve parallel computing. This is very useful to reduce the running time of large-scale program. But AIUA also has its own drawbacks, such as explicitness decreases.

In this paper, just considered the min_sup changes of the association rules updating problem, how to implement the update when the transaction database changes (including the amount of data increases and decreases) will be the future work.

ACKNOWLEDGEMENT

This work is supported by the 2011 Anhui Provincial Natural Science Research Project of University named "Study of Contract Programming Development Environment Based on AOP" under contract number KJ2011B171.

REFERENCES

[1] FENG Yu-cai; FENG Jian-lin. Incremental Updating Algorithms for Mining Association Rules. Journal Software, 1998,9(4), PP:301-306

[2] Tan Pang-Ning; Steinbach.M.; Kumar.V. Introduction to Data Mining. China Machine Press. 2010

[3] CAI Jin; XUE Yong-Sheng; LIN Li; ZHANG Dong-Zhan. Updating Algorithm for Association Rules Based on Fully Mining Incremental Transactions. Computer Science. Vol.34 No.2 2007, PP:220-233

[4] TIE Zhi-xin; YU Rui-zhao. An Efficient Algorithm for Incremental Updating Association Rules. ournal of Zhejiang Sci-Tech University. Vol.25 No.2 2008(3), PP:169-173

[5] ZHENG Wen-zheng. Research on Increment Association Rules Mining Algorithm. Computer and Modernization, 2009(2), PP:92-97

[6] ZHANG Yong-tao; ZHANG Gang-hua. Improved Updating Algorithm Assciation Rules. Computer and Information Technology, 2010(1), PP:4-7