A Novel Algorithm for Association Rule Mining Without Candidate

Huanyin Zhou
East China Institute of Technology
State Key laboratory of Robotics
Shenyang Institute of Automation
Fuzhou, China
z_huanyin3@163.com

Jinsheng Liu
School of Environment Engineer
East China Institute of Technology
Fuzhou, China
Jinsheng22@sina.com

Abstract—A-Priori is an influence algorithm for finding frequent itemsets from association rules. But there are two hard questions may be involved for average users during finding frequent candidates. One question is massive amounts of candidates and the other is that set support count threshold for every level candidate generations. This paper discusses one algorithm called And, which is usually used in logical algorithms, And Code (AC) algorithm can discover frequent itemsets without producing candidates and setting thresholds for candidates. Frequent itemsets can be fast discovered by corresponding codes which are cited by this paper to describe different itemsets for AC algorithm. The support count of frequent itemsets can be computed before the process of scanning during processes of AC algorithm. The codes of itemsets are defined and detailed in section two. Lastly, an example is presented to detail processes of AC algorithm and test that AC algorithm can more efficiently find frequent without candidates than A-Priori algorithm which may produce a large of candidates during scan process.

Keywords- association rule mining; And Code algorithm; frequent itemsets; A-Priori algorithm

I. INTRODUCTION

With huge amount of data being continuously collected and stored, many merchants are becoming more and more interested in mining association rules from their trade databases. Generally, some algorithms of association rule mining have two processes: one is to find all frequent itemsets, the other is generate strong association rules from frequent itemsets[1][2]. Association rules are easily produced once the sub-problem of finding frequent sets in the database is solved. Generally, the first step is the heaviest sub-task which seriously affects the efficiency of some algorithm because of producing a great deal of useless candidates. Some ones advised pruning function optimization [3] to reduce the number of candidates. Reducing times of scanning is also considered in [8].

There have existed many ways to discover association rules [5] [6] [7]. Among them, A-Priori algorithm is one of the most influenced for its finding association rule by candidates with its A-Priori property. This algorithm contains two actions, namely, join and prune. During join step, k-itemset candidates are generated by linking k-1-itemsets with each other. The prune step is to find frequent k-itemsets by removing all k-itemsets candidates whose support counts are less than their corresponding threshold.

From the above two steps, users have to suffer two hard struggles: one is to deal with huge amounts of candidates which are produced during join steps, the other is to set support thresholds for candidates embedded in prune step. The less value of threshold of k-itemsets is the more number of candidates will be produced. Contrarily, if the thresholds set too big some interesting association rules may be lost. Every threshold should be considerate. This paper discusses the And Code (AC) algorithm that does not involve candidates and thresholds.

The paper is organized as the following: AC algorithm and its characters are detailed in section 2. Section 3 describes processes of AC algorithm. An example is presented in section 4. Then the last section concludes this algorithm.

II. AND CODE ALGORITHM AND ITS PROPERTIES

Boolean description is used to judge the presence or the absence of a single item in itemsets. Here the number ‘1’ delegates the presence of an item in one itemset while ‘0’ means an item is the absence of the itemsets. Then every itemset can be represented by one code. Logical algorithm of And is detailed in some math books so this paper is negative its operation.

A. Itemsets Code

Itemset coding appoints ‘1’ as the existence of one item in one itemset, while the number of ‘0’ is considered as the nonexistence of one item in the itemset. The number of 1-items in database decides the itemset code length at the same time its order is determined by user’s needing. Now it is detailed by one example: in one database there are seven 1-items (I1, I2, I3, I4, I5, I6, I7) and there is one itemset (I1I3I4I5I7), so the length of code is 7 and we order the items by I1, I2, I3, I4, I5, I6, I7. the itemset (I1I3I4I5I7) code is 101101. The length of code is reduced during the steps of algorithm.

B. Transaction Reduction

One transaction that does not contain any k-itemsets cannot contain any frequent (k+1)-itemsets. This character can directly prune some itemsets out from the database, so save a lot of compute quantity for these itemsets in coding and in And algorithm. For example, given the least item number of frequent itemsets is j, so these k-itemsets, k<j, will be removed from the transaction database.
C. And Code Algorithm

The symbol of set A And set B is AB at logical algorithm of And. If every itemset code runs And algorithm with other codes it will waste a lot of compute quantity. According to the goal of the algorithm for finding frequent itemsets, only the ‘1’ and ‘1’ is significant to discover frequent itemsets. Some measures have been taken to improve the efficiency of AC algorithm. Firstly, the support count order of single item is arranged from small to large by scanning database. Secondly, the code is produced according to the 1-item existed or did not. The last step is to operate And algorithm all these codes with each other. AC algorithm can reduce a lot of And computation and avoid copying the same common itemsets.

D. Support Count

Support count, which is a usual measure to seek for one frequent itemset support count by AC algorithm, is the number of one itemsets in a transaction database. The following are measures to compute support count by itemsets code. If one itemset code runs And algorithm with another itemset code to achieve one common itemset, then the initial support count is 2. If other two itemsets get the same common code as the above two itemsets, the support count of the common itemset is four. These methods can be clearly understood by some examples. Supposed that I1I2I3I4, I1I2I4I6, I1I2I4I7, and I1I2I4 are four itemsets in one transaction database. Their corresponding codes are 1111000, 1101010, 1101000, and 1101001. Run AC algorithm, 1111000 1101010=1101000, 1111000 1101000=1101000, so the support count of I111I2I4 (1111000) is 4. However, 1111000 And 1101010=1101000; 1111000 And 1101000=1101000; 1111000 And 1101000=1101000, according to the above equations the support count of I11I2I4 (1111000) is 2+1+1=4 for the code of 1111000 is repeatedly applied by And algorithm.

III. PROCESSES OF AC ALGORITHM

The steps of AC algorithm are detailed as the following. Firstly the database should take measure of transaction reduction [1] to lessen the number of itemsets. Then scan database to compute support count of 1-items and prune out these 1-items whose support counts don’t exceed or not equal to the least threshold. The following introduces steps of AC algorithm.

The steps of AC algorithm:

a) Set the least length of item is k. Scan transaction database to decide the number of 1-item and arrange them from small to large. Set the least threshold $\varepsilon_{\text{min}}$ and one other parameter i which is to record the left number of 1-items in database. So $i \geq k$.

b) Run transaction reduction program and prune out these itemsets whose item number is less than k.

c) According to the above introduced coding rules make codes for the left itemsets in database.

d) Scan database to count the support count of every 1-item then prune out these 1-items whose support counts less than $\varepsilon_{\text{min}}$. Set the number of pruned item is n, so $i=i-n$.

e) Order the rest of 1-items by their support count from the least. And record these itemset codes which contain the first 1-item.

f) Run And algorithm for the itemset codes from the step 5). Then according to the support count rules compute every itemset codes whose item numbers are not less than k. Directly prune out all these results with transaction reduction. Last prune the 1-item so i--.

g) Check whether parameter i is larger than or equal to k. If $i=k$, then compute these itemset codes which only contain ‘1’ not ‘0’, then return to 8), or return 5).

h) Output all frequent itemsets and their corresponding support counts.

The following section is about one example for AC algorithm.

IV. AN EXAMPLE FOR AC ALGORITHM

Now an example is presented to detail AC algorithm again. Give a small transaction database to describe AC algorithm procedures. Table 1 is the transaction database and it contains 9 itemsets. The needed itemsets are 3-itemsets. Delete all 2-itemsets, such as I2I4, I2I3, and I1I3. Table 2 shows the left itemset codes. Table 3 counts the support of every 1-items. These describe the detail of figure 1. Then following we will portray the AC algorithm steps in figure 2 and depict the figure 2.

Then following depicts the steps by the example in figure 2.

The steps of AC algorithm:

a) Initial the least item number of frequent itemsets is 3 so $k=3$, its support threshold is 2; there are 9 transaction itemsets in the above transaction database.

b) After transaction reduction 9 transaction itemsets are decreased into 6 transaction itemsets. And there are 6 1-items in the databases so $i=6$.

c) Achieve every itemsets corresponding codes according to rules of itemsets coding. Every itemset coding can be seen in table 2 in figure 1.

d) Compute every 1-item support count and put them in table 3 of figure 1. Every 1-item support count is: S(I1)=S(I2)=S(I6)=5, S(I3)=S(I4)=3, S(I5)=2. So among these 1-items, I5 is the least support count. i is still equal to 6 for every 1-item support count is not less than 2. The support count of I5 is the least.

e) Record the itemset codes that contain I5: 110011, 110110.

f) Run And algorithm for the above two codes: 110011 And 110110=110010. Transaction the code into itemset is (110100) is 2+1+1=4 for the code of 1111000 is repeatedly applied by And algorithm.
I11215. Remove I5 from all the itemset codes, then i--., so i=5.

g) i>3 so return step5, now the support count of I4 or I3 is the least. In AC algorithm every 1-item is right. Then firstly collect all itemset codes that contain I4: 11011, 11010, and 10111. Run And algorithm: 11011 And 11010=11010, 11011 And 10111=10011, 11010 And 10111=10010 because of transaction reduction after this transit 11010 into itemset is I11214 and its support count is 2.

Prune out I4 code from all itemset codes. So i=4. Then it turns to I3. Record all itemset codes that contain I3: 0111, 1111, and 1011. Perform And algorithm for each code. 0111 And 1111=0111, 0111 And 1011=0011, 1111 And 1011=1011, then achieve 0111 and 1011 but prune out 0011 because the length of 0011 is 2. Transit 0111 and 1011 into itemsets I21316 and I11316 respectively. Their support counts are all 2.

Prune out I3 from all itemset codes. So i=3 and parameter i equals to k. then record all itemset codes who have all ‘1’ and from the figure 2 we can tell that there are 3 itemset codes. Then the support count of I11216 is 3. Return 8.

h) Output all frequent itemsets and their corresponding support counts: I11215, I11214, I11416, I12136, I11316 and I11216. Their support counts are 2, 2, 2, 2, 2 and 3 respectively.

So far AC algorithm has been successfully discover all frequent itemsets based on users need. Compared to A-Priori algorithm, AC algorithm has two prominent merits: one is that users can suffer from setting support threshold of candidates; the last one is that there is no candidate in AC algorithm. All these perfectly meet some hard questions in A-Priori algorithm. For example if user sets the support threshold of 1-item is bigger than 2, I11215 is 3 and more than 80 3-itemset candidates are produced during the processes of A-Priori algorithm so there are a great deal of candidates and candidate generations support count, because it only cares for frequent itemsets. So with AC algorithm any user can engage all their attention to frequent itemsets and set the least item number frequent itemsets and their corresponding support thresholds according to users needed.

AC algorithm adopts transaction reduction so as to decrease the itemset codes and save And algorithm computation. In a word all these are served to improve the efficiency of association rule mining. One example is also test the efficiency of this algorithm.

ACKNOWLEDGMENT

The authors thank Yong Zhang professor for his instruction during preparation of this paper.

REFERENCES

[1] Jiawei Han, Micheline Kamber. Data Mining: Concepts and Techniques, 2001.
Figure 2: AC algorithm procedures