Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 15 (2011) 1827 – 1831

Procedia Engineering

www.elsevier.com/locate/procedia

Advanced in Control Engineeringand Information Science

An Improved Apriori Algorithm Based On the Boolean Matrix and Hadoop

Honglie Yu[*a], Jun Wen[b], Hongmei Wang[c] ,Li Jun [de]

[a] *honglie58@163.com,School of Computer Science and Technology, University of Electronic Science and Technology of China, Chengdu, China*

[b]*wenjun@uestc.edu.cnSchool of Computer Science and Technology, University of Electronic Science and Technology of China, Chengdu, China*

[c]*redfoxlover09@163.com,School of Computer Science and Technology, University of Electronic Science and Technology of China, Chengdu,China*

[d]*605246435@qq.com,School of Software, University of Electronic Science and Technology of China,Chengdu, China*

[e]*Liupanshui Tobacco Corp., LiuPanshui, Guizhou, China*

Abstract

The association rule mining plays an important part in the data mining. Association rule mining aims to find rules in the transaction database with the minimum support and minimum confidence which are the user given. In order to find all the frequent item sets from the transaction database efficiently and quickly, an improved *Apriori* algorithm of mining the association rules in this paper is put forward to solve the bottleneck problems of the traditional *Apriori* algorithm. First the Boolean matrix array is used to replace the transaction database. Then the "AND" operation and random access characteristics of array are used. Next the mining algorithm is carried out on the Hadoop Platform. According to the number of the Data Nodes of the Hadoop, the matrix is divided into several parts. Each part is operated separately on one Data Node. It can improve the efficiency of the algorithm.

Selection and/or peer-review under responsibility of [CEIS 2011]

*Keywords:*Apriori algorithm; Boolean Marix; frequent item sets; Hadoop

1. Introduction

A branch of computer science, data mining is the process of extracting patterns from large data sets by combining methods from statistics and artificial intelligence with database management [1]. And

* Corresponding author. Tel.:+8618985902633

E-mail address: 605246435@qq.com

doi:10.1016/j.proeng.2011.08.340

association rule mining is the commonest method in Association Knowledge Discovery. The associations between datasets in the transaction database are complicated and most are implicit. Mining Association rules a component of data mining, is a research field proposed earlier. *Apriori* algorithm is the most famous algorithm, which has been brought in 1993 by Agrawal. *Apriori* follows a generate-and-test methodology for finding frequent item sets, generating successively longer candidate item sets from shorter ones that are known to be frequent. Each size of candidate item set requires a scan through the dataset to determine whether its frequency exceeds the minimum support threshold. With the minimum support and minimum confidence of the user given the *Apriori* algorithm can find rules in the transaction database. And this can be seen as two processes: The first one is to find all frequent item sets, each of the frequent item sets will be at least as frequently as a predetermined minimum support count, *min_sup.* The second is to generate interesting association rules from the frequent item sets. The overall performance of mining association rules is determined by the first step. Mining association rules is usually converted to find the frequent item sets. But traditional *Apriori* algorithm has two deadly bottlenecks [2]:

(1) A great many I/O operations are needed when the database is being scanned frequently. Each item in the candidates of frequent item sets must scan database one time to decide whether it can be joined to the L_k(frequent item sets). So it needs to scan the transaction database as the same number as the elements of the frequent item sets.

(2)A lot of candidates of frequent item sets may be produced. The growth of the candidates increases in a very high speed, which needs a lot of time and main Memory.

The rest of the paper is organized as follows: section 2, give the brief overview of the related works; in section 3, the ideology and description of the improved algorithm is introduced; in section 4, an analysis of the performance of the improved algorithm is clarified; in section 5, the algorithm is analyzed through a case study; in section 6, a conclusion is made.

2. Related work

In this paper, the new improved *Apriori* algorithm [3] makes good use of the Boolean matrix and the parallelism of the Hadoop, to improve its own efficiency. Several algorithms have been proposed in the literature to address the problem of mining association rule [4]. One of the key algorithms, which are the most popular in many applications for enumerating frequent item sets, is the *Apriori* algorithm [5]. The Traditional *Apriori* algorithm needs to scan the transaction database every time when it wants to know the k-frequent item sets. And the candidates are very large. It will waste a lot of time and memory [6]. In this paper the improved algorithm just scans the transaction database one time. Through this scanning it converts the transaction database into Boolean Matrix. And in the first time scanning it can find all 1-frequent item sets. The following operations are carried on the matrix. The “AND” operation of vector is used to operate on the Boolean matrix to find the frequent item sets. Following the Boolean matrix is divided into several blocks, and each block is located on one Data Node of the Hadoop. All the blocks can be operated at the same time; therefore it can reduce the memory and time used for finding the frequent item sets.

Apache Hadoop is a framework for running applications on large cluster built on commodity hardware. The Hadoop framework transparently provides with both reliability and data motion. Hadoop implements a computational paradigm named Map/Reduce, where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster. In addition, it provides a distributed file system (HDFS) that stores data on the compute nodes, providing very high aggregate bandwidth across the cluster [7].

3. The Improved Apriori Algorithm

3.1. The notion of the improved algorithm.

The Boolean matrix is used to describe the transaction database. And the number '1' and '0' are used to replace the corresponding items. After this the Boolean matrix is used to count the support of the item sets in the database and need not scan the transaction database any more. According to the operation of vector, the Boolean matrix's components can be seen as row vectors and such improved algorithm just uses the "and" operation to count the support quickly on the Boolean matrix.

(1) First the transaction database needs to be converted into Boolean matrix. If the transaction database contained *m* items and *n* transactions the Boolean matrix will have *m+1* row and *n+2* columns. The first column registers "items" and the first row records "TID" of the transactions. The last column is used to record the support of the item sets. Second, the min-support is compared with the support of the item sets, if the support of the item sets is smaller than the min-support the row of the item sets will be deleted. By doing this the new Boolean matrix just contains the one-frequent item sets. And if it wants to know the k-frequent item sets, the "AND" operation will just be carried out on the k rows. Finally all the frequent item sets can be found out.

(2) Usually there are a large number of transactions in the transaction database, so the Boolean matrix is very large. And the algorithm is carried out on the Hadoop platform. According to the number of the Data Nodes of the Hadoop, the Boolean matrix is divided into several parts based on the columns. And each part is located on each Hadoop Data Node. By doing this the algorithm can be executed parallel.

3.2. The description of the algorithm.

First input: the transaction database D and the min-support *minsup*. The result output: all the frequent item sets in the database D. The following is the description of the algorithm:

(1)[Convert the transaction database D to Boolean matrix DB]
  DB = convert (database D, *minsup*);
  If DB = NULL
    End there is no frequent item set
  Else
    The 1-frequent item sets are the items which are the rows of the Boolean matrix
(2)[Divide the Boolean matrix into blocks]
  If n is the number of the Data Nodes of Hadoop Platform
  Then the Boolean matrix DB is divided into n blocks, each having several columns of the matrix
(3)[Find all other frequent item sets]
  L_1 = the collection of items in the matrix DB is 1-frequent item sets
  For k from 2 to the Max which is count of the rows of the matrix DB, execute
    $C_k = GETC_k(DB,L_{k-1})$ [use the AND operation of matrix on any k rows of the DB]
    $L_k = GETL_k(DB,C_k\text{-}1,minsup)$ [get the k-frequent item sets]
  Then $L = L_1 \cup L_2 \cup ... \cup L_{k-1} \cup L_k \cup ... \cup L_{Max}$
(4)[Compute the confidence of the item sets to find the association]
(5)[END]

The first step of the algorithm is to convert the transaction database into Boolean matrix. The algorithm needs to scan the database once only. And in this step it can find the 1-frequent item sets and delete the non-frequent item sets. Next, the Boolean matrix is divided into several parts and each part is

located on one Data Nodes of Hadoop. The "AND" operation is carried out on the Boolean matrix to find the other frequent item sets.

4. The Performance Analysis of the Improved Algorithm

The traditional *Apriori* algorithm needs to scan the transaction database when it wants to know the candidates of the item sets. If n is the number of the transactions of database and m is the average length of item sets. So the traditional *Apriori* algorithm needs to scan O(*n***m*) time to find the first 1-frequent item sets. And the time $O(L_{k-1}*L_{k-1})$ to find the candidates Ck. To count the support needs time of $O(n*C_k)$.

The improved *Apriori* algorithm in this paper needs to scan the transaction database one time. And the first time to scan the database can find the 1-frequent item sets and can exclude the non-useful item sets. In the following steps we just use the "AND" operation of matrix to find other frequent item sets. And it need not scan the original database. Because of the high degree parallelism of the Hadoop, the time consumed in counting the support should be cut down. The time is only one in n(there are n Data Nodes of Hadoop).

5. Case Analyze

Assume that D be a transaction database and $D = \{T_1, T_2, T_3, \ldots, T_n\}$. There n is the number of the transaction. And I is a set of items and $I = \{I_0, I_1, I_2, \ldots, I_m\}$, There m is the number of the items.

According to the property the Boolean matrix is used to describe the transaction database. First the transaction database's data format is vertical data format as table 1.

Second, the transaction database is scanned for one time and is converted to Boolean matrix. If the transaction in the database has the items, the Boolean value in the Boolean matrix is noted '1'; otherwise the value is '0'. Table 2 is the Boolean expression of the transaction database.

Table 1 vertical data format of transaction database

Item	TID
I_0	T_1,T_2,T_4
I_1	T_1,T_2,T_4
I_2	T_1,T_2,T_3
I_3	T_3
I_4	T_1,T_4

Table 2 Boolean matrix of the transaction database

Item	T_1	T_2	T_3	T_4	support
I_0	1	1	0	1	3
I_1	1	1	0	1	3
I_2	1	1	1	0	3
I_3	0	0	1	0	1
I_4	1	0	0	1	2

Assume the *minsup* = 3. It deletes the items whose support is less than the *minsup* in the Boolean matrix. Table 3 is the new Boolean matrix.

Table 3 the new Boolean matrix

Item	T_1	T_2	T_3	T_4	support
I_0	1	1	0	1	3
I_1	1	1	0	1	3
I_2	1	1	1	0	3

In table 3, there are three frequent 1-items I_0, I_1, and I_2. And the frequent item sets are made up of the three items. So the largest frequent items are composed up to three items. Next we divide table 3 into blocks according to the number of the Data Nodes of Hadoop. Suppose that there are two Data Nodes, it is divided into two blocks by column. For example, the blocks are as in table 4. DataNode1 holds the block1 and the DataNode2 holds the block2. In each Data Node the Boolean matrix can be seen as the row-vectors as I_0 = {1,1,0,1}， I_1= {1,1,0,1}， I_2= {1,1,1,0}. If it wants to know the frequent 2-items just get them by phase and get the sum.

Table 4 the two blocks_block1 and blocks_block2

Item	T_1	T_2	support
I_0	1	1	3
I_1	1	1	3
I_2	1	1	3
I_3	0	0	1
I_4	1	0	2

Item	T_3	T_4	support
I_0	0	1	3
I_1	0	1	3
I_2	1	0	3
I_3	1	0	1
I_4	0	1	2

Since there are three frequent 1-items I_0, I_1, I_2, the frequent 2-items can be $\{I_0, I_1\}$, $\{I_0, I_2\}$ and $\{I_1, I_2\}$. Therefore the support of $\{I_0, I_1\}$ is $I_0 \odot I_1$ = {1, 1}⊙{1, 1}+{0, 1}⊙{0, 1}=3. So $\{I_1, I_2\}$ is frequent 2-items. As the same the supports of $\{I_0, I_2\}$ and $\{I_1, I_2\}$ are 2, they are not the frequent 2_items. The support of 3-items $\{I_0, I_1, I_2\}$ is $I_0 \odot = 1 \odot I_2$ = {1, 1}⊙{1, 1}⊙{1, 1}+{0, 1}⊙{0, 1}⊙{1, 0} = 2. Thence it is not the frequent 3-items. Finally it can get all the frequent item sets {{I_0}, {I_1}, {I_2}, {I_0, I_1}}. The meaning of the symbol ⊙ is to compute the support of the item sets. It counts the support by adding the summation of the vector by bitwise. In each Data Node of Hadoop the operations is run at the same time.

6. Conclusion

Boolean matrix is used to replace the transaction database; therefore those non-frequent item sets can be removed from the matrix. And it does not need to scan the original database; it just need to operate on the Boolean matrix using the vector operation “AND” and the random access characteristics of array so that it can directly generate the k-frequent item sets. The algorithm is carried out on the Hadoop Platform, thus it can exponentially increase the efficiency of the algorithm.

7. References

[1] M.S.Chen, J.Han, and P.S. Yu. Data Mining: An Overview from Database Perspective. IEEE Yrans, on Knowledge and Data Eng, 1996, Vol.8, Vol.8(No.6): 866-833.

[2] Chen Wenwei. Data warehouse and data mining tutorial [M]. Beijing: Tsinghua University Press. 2006

[3] Hand David, Mannila Heikki,Smyth Padhraic. Principles of Data Mining[M].Beijing:China Machine Press,2002.

[4] J.Hipp, U.Guntzer, and G.Nakaeizadeh. Algorithms for association rule mining – a general survey and comparison. ACM SIGKDD Explorations, 2(1):58-64, June 2000.

[5] R.Agrawal and R.Srikant. Fast algorithms for mining association rules. In Proc. 1994 Int. Conf. Very Large Data Bases, pages 487-499, Santiago, Chile, September 1994.

[6] Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases[C]. In: Proc. of the l993ACM on Management of Data, Washington, D.C, May 1993. 207-216

[7] Tom White: Hadoop: The Definitive Guide.2009.