http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 12 (11): 2246-2250, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.2246.2250

Improved Mining Frequent Itemsets Algorithm Based on Sim

[1]Jiang Yongchun, [1]Li Xiaona, [1]Cui Hairong, [2]Xun Jiao and [1]Wang Yingchun
[1]Qingdao University, China
[2]Shandong Normal University, China

Abstract: In order to solve the problem that apriori algorithm generates candidate itemsets, this study presents an improved mining frequent itemsets algorithm based on Sorting Index Matrix (SIM). The algorithm directly generates frequent 2-itemset through 1-itemset vector and the corresponding matrix multiplication sequentially. From the frequent 3-itemset, it establishes simple SIM for the frequent k-itemsets to realize itemsets leap-step search and connection with the SIM. The whole process just scans the database once, it does not produce candidate itemsets. Experimental result shows that the algorithm improves the efficiency of mining frequent itemsets.

Key words: Sorting index matrix, candidate itemsets, leap-step search, data mining algorithm

INTRODUCTION

Currently, information has become an important strategic resource, the large amount of data led to explosion of information, So, it becomes a very difficult task for people to obtain valuable information from these massive datas. Data mining is a core of knowledge discovery in databases, it is a process that people extract potential useful and ultimately understandable knowledge from the database. It is a very important research subject that mining association rules from transaction datas in the field of data mining research. The target of association rules mining is to mine association relationship either subjects or properties from large database and data warehouse, which is used to guide people to make correct decisions (Caiyan and Ruqian, 2006). Faced by the area of association rule mining to design an efficient mining algorithms of association rules, it is an important challenge. The apriori algorithm was proposed in Agrawal *et al*. (1993), it was a classic algorithms of frequent itemsets mining, its shortcomings was that it produced a large number of candidate itemsets and took up much memory space CPU processing time, so it was difficult to adapt to the mass data mining. To solve these problems, Yixia *et al*. (2006) constructed a matrix combined with the ordered characteristic structure of itemsets, it reduces the number of candidate itemsets. Mingxuan *et al*. (2009) used four pruning strategies to improve the efficiency of data mining. Although, these algorithms have been able to reduce the number of candidate itemsets or improve mining efficiency, but they still did not completely solve the candidate no longer to have this problem. This study presents association rule mining algorithm based on Sorting Index Matrix (SIM) , which search the itemsets by the form of the index. So, this algorithm can avoid generation of candidate itemsets and repeatedly database scan of Apriori algorithm.

FEASIBILITY ANALYSIS

Association rules: Let i is the set of all items, it is called itemset, if the itemset contains k items, it is called k-itemset. Bo *et al*. (2008) let T is the transaction table, each transaction T_i is a itemset, T_i I. Let A and B are the itemsets, transaction T_i contains A, only if A T_i. An association rule can be represented as R: A B, of which A I, B I and A B = φ.

Frequent itemsets: If the support degree of an itemset is greater than or equal to the pre-defined minimum support threshold, the itemset is called frequent itemsets (Ying *et al*., 2011).

Property 1: If the support count of the k-itemsets $\{I_{i1}, I_{i2}, ..., I_{ik}\}$ is not less than minimum support count, then, k-itemsets $\{I_{i1}, I_{i2}, ..., I_{ik}\}$ is called k-frequent itemsets (Jiali and Jia, 2010).

Maximal frequent itemset: In set of k-frequent itemsets, if the itemsets number in set is less than, then the set of k-frequent is called the maximal frequent itemset.

Property 2: If frequent (k-1)-itemsets still can generate frequent k-itemsets, then the itemsets number of frequent (k-1)- itemsets must be greater than or equal to k.

Corresponding Author: Jiang Yongchun, Qingdao University, China

Property 3: If the length of transaction T database within a database is less than (k+1), then transaction T needn't be scanned when generating (k-1)-frequent itemsets from k-requent itemset.

Apriori algorithm: Apriori algorithm uses an iterative method, which is called layered research, k-itemsets are used to explore (k+1) itemsets. First, when this algorithm scans the database, it collects items of meeting the minimum support count, so as to denote the collection of frequent 1-itemsets as, Then, L_1 is used to find frequent collection of 2-itemsets L_2 and L_2 is used to find frequent collection of 2-itemsets L_3. Continue to do so until you can no longer find frequent k-itemsets.

The process of getting L_k from L_{k-1} need two steps:connecting steps and pruning steps:

- **Connecting steps:** Connecting L_{k-1} and itself to generate the set of k-candidate itemsets, marked it as C_k. Let l_1 and l_2 are itemsets of L_{k-1}, $l_1[j]$ is marked as the jth of l_1. For convenience, it is assumed that the transactions or items of itemsets are sorted by dictionary order. For (k-1) itemsets l_1, it means that the items are sorted in sequence, so $l_1[1]<l_1[2]<....<l_1[k-1]$. Performing the connection of $L_{k-1} \bowtie L_{k-1}$ among them, l_1 and l_2 are connectable. If $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge ... \wedge (l_1[k-2] = l_2[k-2] = l_2)$ $[k-1])$, $l_1[k-1] = l_2[k-1]$ only ensures not to generate repeating. The result itemsets, which connect the $l_1$and l_2, is $l_1[1], l_2[1],...,l_1[k-1], l_2[k-1]$
- **Pruning steps:** C_k is superset of L_k, that means all the frequent k-itemsets are included in C_k. Scaning the database to determine the count of each candidate in C_k, thereby to determine L_k. But, maybe C_k is very large, which leads to large calculation amount. In order to compress C_k, Apriori property can be used by these methods mentioned later. whichever, non-frequent (k-1)-itemset is all not subset of frequent k-itemset. Therefore, if the subsets of (k-1) items among the candidate k-itemsets are not in the L_{k-1}, then the candidate k-itemsets can not be frequent, which can be deleted from C_k. Let the length of transaction database is n, in order to generate frequent itemsets which the maximum length is m, Apriori algorithm will scan the database m times, its complexity degree is Ω $(n\times m)$. And Apriori algorithm will produce a large number of candidate itemsets. Based on these two shortcomings, this study presents a new algorithm to generate frequent itemsets, which overcomes the two problems that Apriori algorithm exists

FREQUENT ITEMSETS MINING BASED 0N SORTING INDEX MATRIX

Related definitions

1-Itemsets matrix: Let 1-itemsets matrix $V_j = (d_{1j}, d_{2j},..., d_{nj})$, among them:

$$d_{ij} = \begin{cases} 0 & ij \notin Ti \\ 1 & ij \in Ti \end{cases}, \ 1 \le i \le n, 1 \le j \le m,$$

$$V = (V_1, V_2, ...V_j, ...V_m)$$

is called 1- Itemsets Matrix.

Sort index matrix: When the frequent (k-1)-itemsets produce k-itemsets ($k\le 3$), it is needed to construct sort index matrix of frequent 1-itemsets, it will achieve the leap-step search and connection, Its structure is defined as follows:

Columns of the sort index matrix are composed of serial number, descending 1-itemsets and the itemsets's support counts (sup).

Rows of sort index matrix are produced by descending order frequent (k-1)-itemsets ($k\ge 3$).

Serial number, namely the index number, is obtained row by row, in alphabetical order such as a,b,c,ets, it is used to record the next position of an itemset's occurrence. The itemset support count sup expresses the support count of frequent (k-1)-itemset .

If a line of frequent (k-1)-itemsets is l_1l_2, Firstly, the columns l_i and columns l_2 all set to 1 in some row, the other columns set to 0, each row is operated in such a way, a matrix containing 0 and 1 is obtained. And then 0-1 matrix is indexed.

Arithmetical description: Let the number of transactions as n, the number of frequent 1-itemsets as m, the minimum sup_counts is 2. Combined with Classic Apriori algorithm, association rules mining algorithm based on sort index matrix is described as:

Step 1: Using the formula sup_count:

$$(l_1) = \sum_{i=1}^{n} d_{ij}$$

it can calculate the sup_counts of each itemsets and delete the itemsets which support counts is less than 2 and get frequent 1-itemsets L_1, it can also form descending 1-itemsets and 1-itemsets matrix $|A| = (V_1, V_2,... V_j, ... V_m)$ by sorting the size of sup_counts of L_1.

Table 1: Sort index

Serial number	L_2	I_2	I_1	I_3	I_5	Sup
a	$I_2\ I_1$	B	c	0	0	4
b	I_2, I_3	D	0	c	0	4
c	I_1, I_3	0	e	1	0	4
d	I_2, I_5	1	0	0	e	2
e	I_1, I_5	0	1	0	1	2

Table 2: Sort index matrix

Serial number	L_2	I_2	I_1	I_3	I_5	Sup
a	$\{I_2\ I_1\ I_3\}$	b	B	1	0	4
b	$\{I_2I_1,I_5\}$	1	1	0	1	2

$$
\begin{array}{c}
\\
T_1\\ T_2\\ T_3\\ T_4\\ T_5\\ T_6\\ T_7\\ T_8\\ T_9
\end{array}
\begin{array}{c}
\begin{array}{ccccc} I_2 & I_1 & I_3 & I_5 & I_4 \end{array}\\
\begin{bmatrix}
1 & 1 & 0 & 1 & 0\\
1 & 0 & 0 & 0 & 1\\
1 & 0 & 1 & 0 & 0\\
1 & 1 & 0 & 0 & 1\\
0 & 1 & 1 & 0 & 0\\
1 & 0 & 1 & 0 & 0\\
0 & 1 & 1 & 0 & 0\\
1 & 1 & 1 & 1 & 0\\
1 & 1 & 1 & 0 & 0
\end{bmatrix}
\end{array}
$$

Fig. 1: 1-itemsets matrix

Step 2: Multiplying column vector v^T_j $(1 \le j \le m)$ by matrix $|A-V_j|$ to get L_2.

Step 3: Constructing the sort index matrix, leap-step search k-itemsets by traversing the index numbers.

Case analysis: The transaction database is set up as shown in Table 1. It is assumed that the min-support degree is of 20%, the transaction the number is 10, the min-support count of this itemset is 2.

Formatting frequent 1-itemsets: Traversaling transactions database, calculating support count of each itemset in Table 1, deleting the non-frequent itemsets I_6, getting the descending 1-itemsets (I_2,I_1,I_3,I_5,I_4) by sorting their support counts 1-itemsets, 1-itemsets matrix, as shown in Fig. 1.

Multiplicates the vector and matrix to get frequent 2-itemsets: For matrix |A|, each itemset corresponds to a vector, 5 itemsets corresponds to 5 vectors:

$$V_1 = (1,1,1,1,0,1,0,1,1)^T$$

$$V_2 = (1,0,0,1,1,0,1,1,1)^T$$

$$V_3 = (0,0,1,0,1,1,1,1,1)^T$$

$$V_4 = (1,0,0,0,0,0,0,1,0)^T$$

$$V_5 = (0,1,0,1,0,0,0,0,0)^T$$

Multiplicates the vector V^T_J and matrix $(V_{i+1},V_{i+2} \ldots V_{i+5})$, figure out to the frequent 2-itemsets that begins with the column corresponding to the V_i.

$V_1{}^T = (V_2,V_3,V_4,V_5) = (4,4,2,2)$, we can get 4 2-itemsets beginning with as follows: I_2I_1 (4), I_2I_3(4), I_2I_5 (2), I_2I_4 (2).

$V_2{}^T = (V_3,V_4,V_5) = (4,2,1)$, we can get two 2-itemsets begining with I_1 as follows: I_1I_3 (4), I_1I_5 (2), inside, 2-itemset of I_1I_4 is discarded, because its support count is less than the minimum support count.

$V_3{}^T = (V_4,V_5) = (1,0)$, no frequent 2-itemsets whose support count is less than the minimum support count are produced.

$V_4{}^T$ (V_5) = (0), it does not produce 2-itemsets.

After being arranged, six frequent 2-itemsets are got,t hat is $L_2=\{I_2I_1(4),I_2I_3(4),I_2I_5(2),I_2I_4(2),I_1I_3(4),I_1I_5(2)\}$.

Forming 3-itemsets: Six frequent 2-itemsets are sorted, there is only one 1 corresponding to the column of I_4, It means that itemset I_2I_4 can not be connected with the other 2-itemsets, so delete the rows and the columns corresponding I_4. As shown in Table 1.

The serial numbers are alphabetic list of lowercase letters of a,b,c,d and e, which are the row numbers of the itemset, they just play a role of the row index:

- Firstly scan the 1th row of I_2I_1, leap-step search the column where I_1 is located , go to the third row by the index number c, where I_1I_3 is located row, connect row 1 and row 3 to get frequent 3-itemsets $I_2I_1I_3$. then connect the first row and the last row to get frequent 3-itemsets $I_2I_1I_5$. The support count of $I_2I_1I_5$ is 4, while I_2I_1 and I_1I_5 are respective 4 and 2, so the support of $I_2I_1I_5$ is 2
- Secondly continue to scan from row 2 to row 3, it can not stop until the last row. In this step, Apriori algorithm is to form candidate 3-itemsets and it produce 6 itemsets as well, it is not conducive to the improvement of the efficiency of the algorithm
- Thirdly after arranged, two 3-itemsets are got, it expressed as $L_3= \{I_2I_1I_3\ (4), I_2I_1I_5\ (2)\}$. Then, construct sort index matrix of L_3, as known from table 2, no index number can meet the conditions of downward connection, therefore, no frequent 4-itemsets will be produced, algorithm ends

The algorithm result as follows:

Frequent 1-itemsets is $L_1 = \{I_2(7), I_1(6), I_3(6), I_4(2), I_5(2)\}$
Frequent 2-itemsets is $L_2 = \{I_2I_1\ (4), I_2I_3\ (4), I_2I_5\ (2), I_2I_4\ (2), I_1I_3\ (4), I_1I_5(2)\}$
Frequent 3-itemsets is $L_3 = \{I_2I_1I_3\ (4), I_2I_1,I_5\ (2)\}$
Frequent 2-itemsets is $L_4 = \varphi$, the maximum frequent itemsets is frequent 3-itemsets.

Efficiency analysis: Compared with Apriori algorithm in time: Suppose the length of transaction database is n, we want to produce maximum length m of frequent itemsets.

- The time complexity of Apriori algorithm to scan the database is Ω (n×m) and the algorithm to scan the database when asked complexity Ω (m)
- In the entire process of generating frequent itemsets, Apriori algorithm generates candidate itemsets C_k and frequent itemsets L_k. However, in this study, the algorithm directly generates the frequent itemsets L_k
- Generate frequent 2-itemsets L2, Apriori algorithm first connects frequent 1-itemsets L_1 to generate candidate 2-itemsets, then, carry on pruning operations on itemsets whose support count is less than minimum support threshold by scanning database. The algorithm of this study generates frequent 2-itemsets by multiplying the vectors and matrix to improve the efficiency of operations without pruning operations
- Begining with k≥3, in order to generate frequent k-itemsets, frequent (k-1)-itemsets need to create a simple sort index matrix, as the most important feature of the index is to accelerate the speed of information retrieval. It accomplishes the leap-step search with index number, the rows which can't meet the conditions of downward connection don't need to scan, so it improve the mining speed. While the total operation time of Apriori algorithm is time of scanning the database and generating candidate itemsets and pruning, Therefore,the running time of this algorithm is far shorter than Apriori algorithm

EXPERIMENT ANALYSIS

Experiment environment: This algorithm is accomplished by language C# in the condition of VS2008, Inter (R) Core (TM) i3 2.27 GHz/1.92 GB (OS is Windows XP).

Experiments automatically generate a data set by using the random function Random (), The database will produce a new data set as soon as input transaction |T| and itemset |I|, This kind of data set fully embodies the efficient performance of the proposed algorithm without any artificial interference.

Contrast of the experiment results: Analyzing the experiment results from two expects:

- The comparison of running time with different support degrees makes shown in Fig. 2. It shows that the running time of this algorithm is always shorter than that of Apriori algorithm in the same data set (|T| =1000, |I| = 30)

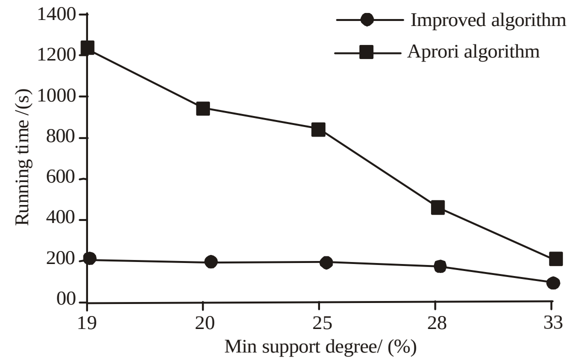

Fig. 2: Comparison of running time with different support degrees

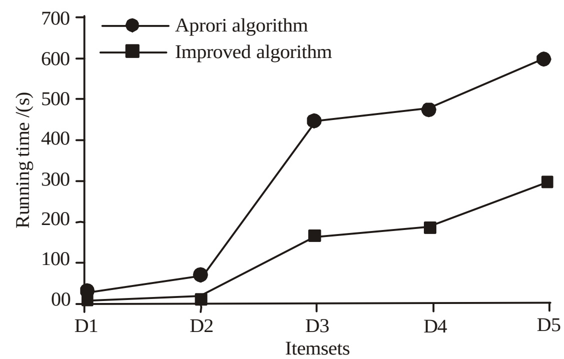

Fig. 3: Comparison of running time with different data sets

- The comparison of running time of different datasets with the same support degrees makes shown in Fig. 3. It generates 5 datasets of different average length of transactions and itemsets, they are D1|T| = (100, |I| = 15), D2|T| = (200, |I| = 30), D3 |T| = (1000, |I| = 30), D4|T| = (500, |I| = 30), D5|T| = (2000, |I|=30). It shows that the running time of this algorithm is always shorter than that of Apriori algorithm in each data set when the minimum support count is less than 28

CONCLUSION

With the help of the sort index matrix of this data structure, the structure achieves leap-step search and connection of the itemsets, then frequent itemsets are produced. It needs to scan the database only once during the whole process. Experiment results show that the algorithm is better than Apriori algorithm in time efficiency. Lower minimum support degree, the superiority of performance is more obvious. Our further research will introduce this algorithm into cloud computing and study the algorithm in the field of computer forensics under the condition of cloud computing applications.

REFERENCES

Agrawal, R., T. Imielinski and A. Swami, 1993. Mining association rules between sets of items in large databases. Proceedings of the ACM SIGMOD International Conference on Management of Data, May 25-28, 1993, Washington, DC., USA., pp: 207-216.

Bo, X., X. Qianfang, L. Zhiqing, 2008. The trusted association rules and based on maximal clique mining algorithm. J. Software, 19: 2597-2610.

Caiyan, J. and L. Ruqian, 2006. Association rule mining quantitative models of sampling error and quickly estimate calculation method. J. Computers, 29: 625-634.

Jiali, X. and C. Jia, 2010. A matrix based frequent itemsets updating algorithm calculation. Application Stud. Comput., 27: 837-840.

Mingxuan, H., Y. Xiaowei and Z. Shichao, 2009. Based on the pseudo-phase matrix weighted association rule mining. J. Software, 20: 1854-1865.

Ying, X., Z. Jun and W. Guoyin, 2011. The mining algorithm of temporal association rule and its application in ITS. Computer Sci., 38: 173-176.

Yixia, Z., Y. Liwen and H. Shuiyuan, 2006. Based on sort matrix and tree association rules Mining algorithms. Computer Scie., 33: 196-198.