An Improved Mining Algorithm of Maximal Frequent Itemsets
Liu Xiao-bing, Zhai Kun

Abstract

Mining maximal frequent itemsets is very important in many data mining applications. How to improve the efficiency and effectiveness of mining algorithm has become an interesting issue in the world. In this paper, we introduce a new method to solve this problem, which is based on graph theory. Firstly, the concept of directed itemsets graph and the trifurcate linked list storage structure are proposed. Secondly, the mining algorithm of maximal frequent itemsets based on this kind of structure is presented and developed. Finally, through some experimental analyses on the standard data sets, the results have demonstrated that the proposed mining algorithm can realize one time database scanning and improve time-storage efficiency a lot.

Keywords: Improved Algorithm, Maximal Frequent Itemsets, Graph Theory

1. Introduction

Association rule mining is proposed by a database expert named R. Agrawal in 1993, mainly based on “support and confidence” framework system [1]. The problem with mining association rules can be distilled into two steps. The first step involves finding all frequent itemsets in databases. Once the frequent itemsets are found, generating association rules is straightforward and can be accomplished in linear time [2].

Mining frequent itemsets in association rules mining plays a crucial role [3]. Due to the closure principle of frequent itemsets, the maximal frequent itemsets (MFI, for short) can include all information of frequent itemsets [4]. In other words, the question of mining frequent itemsets can be translated to mining maximal frequent itemsets. So in recent years, much research has focused on mining maximal frequent itemsets of association rules.

Nowadays, most mining algorithms of maximal frequent itemsets are improved or derivative algorithms based on Apriori or FP-tree [5, 6].

The underlying idea of the Apriori-based algorithm is to generate candidate itemsets by joining frequent itemsets that have been found, and then check their frequencies. Max-Miner [7], Pincer-Search [8], Depth-Project [9], SmartMiner [10], GenMax [11], Mafia [12], are based on the generic concept of the Apriori algorithm. These algorithms need to scan databases many times, which increase the consumption of I/O resources [13].

The basic idea of the FP-tree-based algorithm is to partition the original database to smaller sub-databases by some partition cells, and then to mine itemsets in these sub-databases. Unless no new itemsets can be found, the partition is recursively performed with the growth of partition cells. DMFIA [14], FPmax [15], SFP-Max [16], DFMFI [17], MFIM_P [18], FPmax* [19], exploit the concept of the FP-tree. These algorithms need to scan databases two times, and are highly demanding when it comes to data structures and storage resources [20].

In order to improve the efficiency and effectiveness of mining algorithm of maximal frequent itemsets very much, we will put forward a new method, which is based on graph theory. The method can realize a single scanning of the databases, and also significantly decrease time-storage consumption.

2. MFI mining algorithm based on graph theory

2.1. Directed itemsets graph

The directed itemsets graph denoted as \( G = (V, E) \), is defined as follows:
(1) The node set $V$ of the directed itemsets graph is defined as the set $F$ of 1-frequent item of databases, viz. $F = \{ f_1, f_2, \ldots, f_n \}$. Each node has three attributes: a name of frequent item, a transaction list (denoted as tidlist) of frequent item, and a supporting number of the frequent item.

(2) The directed edge $<f_i, f_j>$ of the edge set $E$ of directed itemsets graph is defined as a 2-frequent itemset.

2.2. Trifurcate linked list storage structure

The trifurcate linked list storage structure of directed graph is composed of index linked list, node linked list and connection linked list. The detailed information of the structure is shown in the Figure 1.

![Figure 1. Detailed structure of trifurcate linked list storage structure](image)

Index linked list is composed of index units. Index units are arranged sequentially. Every unit includes two linked list pointers and two integer variables. Two pointers are pointed to a node linked list and a connection linked list, respectively. Two integer variables are applied to store the length information of node linked list and the connection linked list separately.

Node linked list is used to store the node information of directed itemsets graph. Its units are nodes, which are arranged sequentially. Every unit includes data domain and connection domain. In the view of directed itemsets graph, the nodes are described as the 1-frequent itemsets. Therefore, the information of data domain includes name of frequent item, tidlist of frequent item, supporting number of frequent item. The information of connection domain includes a pointer and an integer variable. The out pointer points to the relevant connection unit of connection linked list.

The unit of connection linked list is applied to store relative address of the adjacent node. The relative address is composed of two offset variables that are applied to store the offset of node linked list in index linked list and the offset of node in node linked list. Therefore, the node of node linked list can find a section of sequential units in the connection linked list through out pointer and out degree. The node address of connection units is the relative address of adjacent node. Through the relative address, the adjacent node can be found and connected, which represents the edge information.

2.3. Construction algorithm of the directed itemsets graph

The “traditional” algorithms of association rules mining adopt a notion of horizontal databases, viz. $T = (Tid, Itemset)$. For the sake of decreasing the number of times of scanning databases, we transform horizontal databases to their vertical counterparts in this study, viz. $(Item, Tidlist)$ [21].

The binary coding technique is used [22, 23]. The Tidlist length of item is defined as being equal to the total number $L$ of transactions in the databases. The Tidlist of item is represented through $L$ binary bits. Every bit is set to “0” or “1” to denote non-support or support of relevant transaction of item in the databases. The supporting number of itemsets is just computed through executing logical operation of
binary bits of Tidlist instead of set operation. Therefore, this method improves the computational and storage efficiency.

Sort ascending of 1-frequent itemsets can improve search efficiency [24]. Therefore, we sort the 1-frequent itemsets in the ascending order of the supporting number. If two items have the same supporting number, they will be sorted according to the lexicographic order [25]. Then the nodes take logical operation of the Tidlist with each other in turn. If the supporting number is no lower than the minimum supporting number after running the “and” operation of two nodes, these two nodes are connected by the edge that forms 2-frequent itemsets. After repeating this operation, the directed itemsets graph is constructed. The construction algorithm of the directed itemsets graph is concisely described in Figure 2.

**Figure 2.** Construction algorithm of the directed itemsets graph

**2.4. MFI mining algorithm based on directed itemsets graph**

After constructing the directed itemsets graph, the mining process of maximal frequent itemsets is entirely transformed to the traversal process of the directed itemsets graph. Initially, the first node is selected as a starting node of the overall traversing process, where we visit its adjacent node. Afterwards, starting from this adjacent node, we visit its adjacent node. This visiting process continues until the last adjacent node or the supporting number being lower than minimum support degree. In this way, a maximal frequent itemset has been mined and stored into the set of maximal frequent itemset. If the following mined frequent itemsets are a subset of maximal frequent itemsets that already have mined, the frequent itemsets are not stored to the set of maximal frequent itemset. By repeating this operation, the
overall traversing process does not finish until every node of directed itemsets graph has been selected as a starting node once. MFI mining algorithm based on directed itemsets graph is shown in Figure 3.

\[
\begin{align*}
S & \text{ is the stack storage} \\
\mathcal{FI} & \text{ is the set of frequent item} \\
\mathcal{MFI} & \text{ is the set of maximal frequent item} \\
\cup \text{ and } - & \text{ are union and difference operations.}
\end{align*}
\]

**Input:** directed itemsets graph and minimum support degree  
**Output:** maximal frequent itemsets

**Method:**
1. \(\mathcal{Nlist} = \text{List node pointer and } \mathcal{Clist} = \text{List connection pointer} \) /* the physical address of directed itemsets graph is located */
2. \(\mathcal{MFI} = \emptyset \) /* MFI is initialized to null */
3. For each node \( N \) in the \( \mathcal{Nlist} \) do
4. Push node \( \mathcal{Nlist} \) into stack \( S \)
5. \( \mathcal{FI} = \mathcal{Nlist} \) /* \( \mathcal{FI} \) is initialized to an 1-frequent itemset */
6. While the stack \( S \) is not empty
7. Select unraveled \( \mathcal{Clist} (S, \text{top}) \) to \( \text{New} \) /* \( \text{New} \) is a variable to store the node that is visited just now */
8. If \( \text{New} \) is not null Then
9. If the support degree of itemsets \( \{\mathcal{FI}, \text{New}\} \) is no lower than minimum support degree Then
10. \( \mathcal{FI} = \mathcal{FI} \cup \text{New} \)
11. Push the \( \text{New} \) into stack \( S \)
12. Else continue
13. Else If \( \mathcal{FI} \) does not have a superset in \( \mathcal{MFI} \) Then
14. Add \( \mathcal{FI} \) to \( \mathcal{MFI} \)
15. Else continue
16. \( \text{Old} = \text{Pop} \) (\( S \)) /* the node is popped for backtracking visit */
17. \( \mathcal{FI} = \mathcal{FI} - \text{Old} \) /* \( \text{Old} \) is a variable to store the node that is returned just now */
18. Loop
19. Output \( \mathcal{MFI} \)

**Figure 3.** MFI mining algorithm based on directed itemsets graph

### 3. Experimental analysis

The improved mining algorithm proposed by this study will be referred to as TDM-MFI. To evaluate the performance of the algorithm, we contrast the performance of the TDM-MFI algorithm with some known mining algorithms of the maximal frequent itemsets, such as GenMax, MAFIA, MFM_P and FPmax*. All experiments are run on a PC with P4-3GHz and 1GB main memory. The programs are written by C++.

#### 3.1. Data sets

Two real-world data sets are used in this experiment, namely Congressional Voting Records data, and Adult data, which come from the UC Irvine Machine Learning Database Repository [26]. Congressional Voting Records data set has 435 transactions and 17 attributes, which compose of a sparse database. Adult data set has 48,842 transactions and 15 attributes. It forms a dense database. Before mining algorithms are applied to these data, several pre-processing methods have been realized in the databases, such as intervention method [27, 28].
3.2. Performance of execution time

The experimental results concerning execution time for the five algorithms for each database are shown in Figure 4 and figure 5. We can note that Mafia is faster than Fpmax* in Congressional Voting Records database in Figure 4, because Mafia does not exhibit much workload in the small and sparse databases. So only few maximal frequent itemsets and candidate maximal frequent itemsets are generated. Fpmax* has to construct FP-tree from database and use much pruning algorithms, which needs to take much time. However, TDM-MFI constructs directed itemsets graph directly, which has included the information of 1-frequent itemsets and 2-frequent itemsets. So TDM-MFI is the fastest algorithm for the sparse databases.

Figure 4. Performance comparison - execution time (Congressional Voting Records data)

In Figure 5, FPmax* is better than Genmax, Mafia and MFIM_P, but TDM-MFI outperforms FPmax*, which means that TDM-MFI also has the better time performance for dense databases, especially under condition of high minimum support degree, because the traversing time of the directed itemsets graph decreases quickly when the maximal frequent itemsets have been generated fewer and the length of maximal frequent itemsets is shortened.

3.3. Performance of memory usage

The experimental results concerning memory usage are reported in Figure 6 and Figure 7. From the figures, we can conclude that the TDM-MFI uses less main memory than the other algorithms, and this observation applies not only to sparse databases but also the dense ones. As Figure 6 illustrated, Mafia exhibits better performance of memory usage than Genmax, MFIM_P, and Fpmax* when few maximal frequent itemsets and candidate maximal frequent itemsets are generated, which do not occupy much memory. The Fpmax*, in which one constructs FP-trees there is also some memory requirement to construct FP-arrays. However, TDM-MFI only requires three linked list to store directed itemsets graph. Many variables in the trifurcate linked list storage structure are just to store physical addresses or offset addresses, which can save memory a lot.
In Figure 7, Mafia and Genmax must store lots of candidate maximal frequent itemsets and intermediate sets in main memory. MFIM_P constructs and stores FP-tree after certain pruning. FPmax* stores a FP-arrays and a variation on the FP-tree (MFI-tree). TDM-MFI just stores the information of 1-frequent itemsets and 2-frequent itemsets, which is more efficient and compact. If the number of 1-frequent itemsets does not change much when minimum support degree becomes lower, the size of directed itemsets graph base on trifurcate linked list does not change much either.

Figure 6. Performance comparison - memory usage (Congressional Voting Records data)

Figure 7. Performance comparison - memory usage (Adult data)

4. Conclusions

Mining maximal frequent itemsets is a fundamental issue in many types of data mining applications. In this paper, we introduce a directed itemsets graph to store the information of frequent itemsets of transaction databases. Next we create the trifurcate linked list storage structure of directed itemsets graph, and finally develop the mining algorithm of maximal frequent itemsets based on directed itemsets graph. Through running some well-known algorithms and carrying out the thorough comparative analysis on two real-world standard data sets, the experimental results have demonstrated that the improved MFI mining algorithm provided by this paper can not only realize a single scanning of the database, but it also significantly decreases time-storage consumption.

5. Acknowledgements

Support from the National Natural Science Foundation, China (Grant No. 61034003) and Canada Research Chair Program (W. Pedrycz) is gratefully acknowledged.
6. References