PROCEEDINGS OF SCIENCE

Improved Frequent Pattern Mining Algorithm Based on FP-Tree

Fei Wei[1]
School of Management Science and Engineering, Shandong Normal University
Jinan, 250014, China
E-mail: weifei1027@163.com

Laisheng Xiang[2]
School of Management Science and Engineering, Shandong Normal University
Jinan, 250014, China
E-mail: 459132653@qq.com

Xiyu Liu[3]
School of Management Science and Engineering, Shandong Normal University
Jinan, 250014, China
E-mail:xyliu@sdnu.edu.cn

The FP-Growth algorithm is currently one of the most efficient frequent pattern mining algorithms, which does not produce a large number of candidate item sets in the process of data mining. However, the traditional FP-Growth algorithm needs to scan twice conditional pattern bases to construct the FP-Tree and conditional FP-Tree, which runs slowly and consumes a lot of time. To optimize this algorithm, an improved FP-Growth algorithm is put forward, which is carried out by using a two-dimensional table to record support counts between items, therefore the second traversal of conditional pattern bases can be avoided. In this paper, some instances and experiments are implemented. Experimental results demonstrate that this improved FP-Growth algorithm based on a two-dimensional table is an efficient frequent pattern mining algorithm of higher performance than the traditional one.

ISCC 2015
18-19, December, 2015
Guangzhou, China

[1]Speaker
[2]Corresponding Author
[3]This research is supported by the Natural Science Foundation of China (No.61170038, 61472231), the Humanities and Social Science Project of the Ministry of Education of China (No.12YJA630152), the Social Science Foundation of Shandong Province (No.11CGLJ22), the Natural Science Foundation of Shandong Province (No.ZR2011FM001) and the University Independent Innovation Project (No.201401202).

PoS(ISCC2015)037

1. Introduction

The association rule miningis an important research topic in the field of data mining[1] , and also a technology of discovering useful information hidden in large databases or data warehouses. In terms of researches on the association rule mining, the frequent pattern mining is the key technique and basic step.

To solve the problem underlying in mining frequent pattern, Apriori algorithmwas firstly presented by R.A grawal et al. in 1993[2]. It is a breadth-first algorithm and uses the strategy of generating candidate sets step by step, which can produce a large number of candidate sets in the mining process and result in the repeatedly scanning of the database. When the database grows large to some extent, its efficiency can be reduced. As to defects of Apriori algorithm, Han Jiawei et al. proposed an frequent pattern mining algorithm based on the FP-Tree (Frequent Pattern Tree) [3,4]. Compared with the Apriori algorithm, it does not generate candidate sets and only needs to scan the database twice, which greatly improves the efficiency of mining frequent item set. However, the FP-Growth (Frequent Pattern Growth) algorithm[5] is required to create the complex data structure and most of the time is mainly consumed in the traversal of conditional pattern bases and the construction of the FP-tree and conditional FP-tree. In this paper, an improved FP-Growth algorithm is put forward. This algorithm uses a two-dimensional table to record the support counts between items, omitting the second traversal of conditional pattern bases when constructing the corresponding conditional FP-Tree, which is time-saving. [6]

2. Basic Concepts

2.1 Frequent Item Set

Let I={I1, I2, I3...In} be a collection of items. D is the set of transactions in the database, where each transaction T is a set of items, so I includes T. For any transaction A which is included in I, if and only if the transaction T concludes A, A can be called the item set. The support count of item set A is the number of the transaction which includes A in the database D. If the support count of item set A is greater than or equal to a given support count, A can be called the frequent item set and the given support count is the minimum support count[7].

2.2 FP-Tree and FP-Growth Algorithm

In the FP- tree structure, each node consists of three elements: item name, support count of node and node-link. To facilitate the traversal, the head table is created, which consists of the item name and head of node-link. Meanwhile, the head of node-link is pointed to the first node with the same same as it.

The FP-Growth algorithm is mainly the process of constructing the FP-Tree and conditional FP-Tree[8]. Specific steps are as follows.

Step1: scan the database D firstly and export a collection of frequent items (1 item set) and their corresponding support counts. Delete the item whose support does not meet the minimum support. At the same time, create the head table.

Step2: construct the root node of the original FP-Tree and scan the database D twice. Items of each transaction are processed according to their order (i.e.,support count in a descending order). Afterwards, create a branch for each transaction.

Next, the frequent pattern mining process is converted into an FP-Tree mining process.

(1) For each frequent item, construct its corresponding conditional pattern base and FP-tree.

(2) Repeat this process for each new constructed FP-tree, until it is empty or contains only one path.

(3) When the constructed FP-tree is empty, the prefix is the frequent pattern. When it contains only one path, all frequent patterns can be acquired by connecting their prefixes with all possible combinations enumerated from the path.

Among all of frequent pattern mining algorithms, the FP-Growth algorithm opens a new way of effectively mining frequent patterns, but its temporal and spatial efficiency is not high enough. As can be seen from the above process, when constructing a new conditional FP tree, its corresponding conditional pattern base will be traversed twice. If the database is very large, two traversals of conditional pattern bases will consume a lot of time.

3. FPGrowth Algorithm Based on Two-dimensional Table

3.1 Impvoved FPGrowth Algorithm

Traditional FP-Growth algorithms ignore the relationship between the new support count list and original support count list, and needs to traverse conditional pattern bases two times. Considering the shortcoming of the traditional algorithm, an improved algorithm is presented. Its main core is to use a two-dimensional table to record the support counts between any two items, so that when a new conditional FP-Tree should be constructed, existing support counts can be given by original ones, which avoids the first traversal of conditional pattern bases.

The detailed description of the improved FP-Growth algorithm is as follows.

(1) Create a two-dimensional table while traversing the database firstly. The table records support counts between any two items in each transaction and all initial values are set at 0. Then, update the data of this two-dimensional table according to a certain rule. For example, if there is [I1, I4] in a transaction, the corresponding support count increases by 1,where[I1, I4] and [I4, I1] are two different items.

(2) When a FP-Tree mining goes on, only traverse its corresponding conditional pattern base once. The new support count can be obtained from the original two-dimensional table. For instance, if the current item is I2, extract columns and rows related with the item I2 from this two-dimensional table. Next, get support counts between I2 and other items by adding corresponding row and column values. Then just traverse its corresponding conditional pattern base one time to construct the conditional FP-Tree. At the same time, create a new support count with a two-dimensional table.

Overall, in addition to using a two-dimensional table to record items' support counts, remaining steps of the improved FP-Growth algorithm are same as the traditional one.

3.2 Algorithm Example

In the table below, an example is given to describe clearly the implementation of the improved FP-Growth algorithm based on a two-dimensional table. In this case, the database D contains 12 transactions, as shown in Table 1. Set the minimum support count at 2.

Transaction	Item	Transaction	Item
T1	I1	T7	I1 I2 I3 I5
T2	I1 I2 I5	T8	I1 I2 I3
T3	I2 I4	T9	I1 I4 I5
T4	I2 I3	T10	I1 I6
T5	I1 I3	T11	I2 I3 I7
T6	I1 I2 I4	T12	I3 I5 I6

Table 1: Database D

First of all, scan the database D and obtain the head table as shown in Table 2. Then create an original support count two-dimensional table and set all initial support counts at 0. The transaction T1 only has one item I1, so the two-dimensional table is not updated. For the transaction T2, combinations between any two terms have [I1, I2], [I1, I5] and [I2, I5]. Therefore, its corresponding value in the two-dimensional table adds 1. Similarly, handle other transactions in the database D to update this two-dimensional table. At last, the resulting original two-dimensional table records support counts between any two items in the database D, as shown in Table 3.

Item	Frequent Degree	Pointer
I1	8	0
I2	7	0
I3	6	0
I5	4	0
I4	4	0

Table 2: the Head Table

Item	I1	I2	I3	I4	I5	I6	I7
I1	0	4	3	2	3	1	0
I2	0	0	4	2	2	0	0
I3	0	0	0	0	2	0	1
I4	0	0	0	0	1	0	0
I5	0	0	0	0	0	0	0
I6	0	0	0	0	0	0	0
I7	0	0	0	0	0	0	0

Table 3: Original Support Count Two-dimensional Table

Since frequent degrees of the item I6 and I7 do not meet the minimum support count, delete rows and columns related with them from the original support count two-dimensional table to get a new one, as shown in Table 4. Then, scan the database D twice and construct the initial FP-Tree.

Item	I1	I2	I3	I4	I5
I1	0	4	3	2	3
I2	0	0	4	2	2
I3	0	0	0	0	2
I4	0	0	0	0	1
I5	0	0	0	0	0

Table 4: New Support Count Two-dimensional Table

As the support count of the item I4 is the smallest, construct the conditional pattern base starting from I4 and then its corresponding FP-Tree. Extract rows and columns associated with I4 from this new support count two-dimensional table to give support counts between the item I4 and other items: sup[I4, I1]=4+0=4, sup[I4, I2]=2+0=2, sup[I4, I3]=0+0=0, sup[I4, I5]=0+1=1. From the above process, the support count list [I1:4, I2:2, I5:1, I3:0] can be obtained when the current item is I4. Afterwards, traverse the conditional pattern base once to construct its conditional FP-tree.

Similarly, when the current item is I5, the support count list is [I1:3, I2:2, I3:2,I4:1]。

Anyhow, in addition to constantly creating the new support count two-dimensional table and omiting the first traversal of conditional pattern bases, other frequent pattern mining steps are taken according to the traditional FP-Growth algorithm.

3.3 Algorithm Analysis

When the database is quite large and sparse, the improved FP-Growth algorithm based on the two-dimensional table can operate more efficiently, which avoids the first traversal of conditional pattern bases and can construct the FP-tree directly. As the FP-tree and conditional FP-tree will be great in the sparse database, the time saved by omitting traversals is much more than the time to construct a two-dimensional table under this circumstance.

When the database is very dense, the FP-tree will become very close. For the densely compressed FP-tree, each traversal can be quite fast. Calculations of support counts between any two items is very high. In this situation, effect of the improved FP-Growth algorithm is not very remarkable.

Therefore, in practical applications, if the database is sparse, the improved FP-Growth algorithm is better with respect to its effectiveness.

4. Experiments and Results

In order to demonstrate the effectiveness of the improved FP-Growth algorithm, the following experiments have been carried out. VC++6.0 is used on P4-2.4/512 M computer to achieve this improved FP-Growth algorithm provided the minimum support is 1%, 2%, 3%, 4%, 5%, 6% or 7%, compared with the traditional FP-Growth algorithm. In this experiment, the

database contains 100,000 transactions. Experimental results of the running time of these two algorithms are shown in Table 5 and Fig. 1.

Support count(%)	Traditional algorithm time(s)	Improved Algorithm time(s)
1.00%	35	20
2.00%	41	32
3.00%	50	39
4.00%	56	44
5.00%	60	54
6.00%	69	56
7.00%	73	64

Table 5: the Running Time of These Two Algorithms Between Different Supports

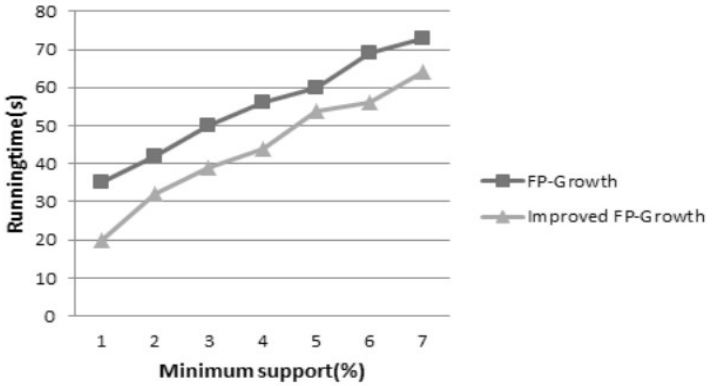

Figure 1: the Running Time of these Two Algorithms

As we can see from Table 5 and Fig. 1, the running time of the improved FP-Growth algorithm is always less than the traditional one with different minimum supports. Furthermore, by comparing the frequent pattern mining results of these two algorithms, the improved FP-Growth algorithm has no effect on the operating results of the traditional algorithm, which confirms its validity and effectiveness.

5. Conclusion

The frequent item set mining is the major concern in the association rule mining. In this paper, an improved FP-Growth algorithm based on the FP-Tree and combined with a two-dimensional table is put forward. By using a two-dimensional table to record support counts between any two items, the improved FP-Growth algorithm reduces the scanning number of condition pattern bases when constructing the conditional FP-tree, as a result, its efficiency is enhanced.

References

[1] R. Agrawal, Mielnskit, Swamia. *Ming association rules between sets of items in large database*. Proceedings of the ACM SIGMOD International Conference on Management of Data New York ACM, 207-216(1993)

[2] R.Agrawal, R.Srikant. *Fast algorithm for mining association rules*. Proceedings of the International Conference on Very Large Database. Morgan Kaufmann. 471-499(1994)

[3] J. w. Han et al. *Data mining concepts and techniques*. High Education Press. Beijing.157-159(2001)

[4] L. S. Ma, Huiwen Deng. *The minimum frequent item set mining algorithm based on FP-Tree*. Computer engineering and design. 29(2): 385-388(2008)(In Chinese)

[5] J. w. Han, J. Pei. *Mining frequent patterns without candidate generation*. ACM SIGMOD Record. 29(2): 1-12(2000)

[6] W.Chen. Mining *Algorithm for Weighted Frequnet Pattern Based on FP Tree*. Cnmputer Engineering.38(6): 63-65(2012)(In Chinese)

[7] Y.Yang, Y.X. Luo. *Improved algorithm based on FP-Growth*. Computer Engineering and Design.31(7): 1506-1509(2013)

[8] X. J. Lv. *Research on Improved FP-Growth Algorithm with Mapreduce*. Computer technology and development.22(11): 123-130(2012)