ATLANTIS PRESS
Advances in Computer Science Research, volume 83
8th International Conference on Social Network, Communication and Education (SNCE 2018)

Design and Implementation of an Improved Apriori Data Mining Algorithm

Meilin Zeng[1,a *],Qiangqiang Xiong[2,b] and Ke Li[3,c]
[1]Jiangxi Vocational Technical College of Industry Trade Nanchang 330038
[2]Nanchang institute of technology Nanchang 330013
* The Corresponding Author

Keyword: Data Mining; Association Rules; Frequent Itemsets; Apriori Algorithm

Abstract. The need to scan the database D many times when the Apriori algorithm is applied to a large database causes the I/O load overhead of the disk to increase. An improved Apriori algorithm is designed. After scanning the original database D for the first time, it generates a candidate transaction database A_k. In the process of generating frequent itemsets, the candidate transaction database A_k is scanned each time. Experiments show that if the K value is very large, the number of A_k will be much less than that of the original database, which will solve the problem of I/O overload and reduce operation time, so as to achieve the purpose of optimizing Apriori algorithm.

Introduction

Association rules mining is an important research area in the field of data mining, and aims to mine interesting associations in transactional databases. Apriori algorithm is proposed by Agrawal et al. However, Apriori algorithm has two bottlenecks: a). Scanning the transaction database multiple times requires a lot of I/O load; b). Generating a large set of candidates, this is a challenge to run time and main memory space.

Aiming at the disadvantages of Apriori algorithm, such as low efficiency of generating candidate itemsets and frequent scanning of database, some improved algorithms are proposed. Some papers propose to reduce the number of scan candidate sets by first finding the maximal frequent K-sets and then finding the remaining frequent-sets. However, these methods are relatively complicated. To this end, this paper presents a simple and easy method. After the first scan of the transaction database D, the algorithm not only generates a database of candidate transactions, but only needs to scan candidate transactions in the process of generating frequent itemsets Database, instead of repeatedly scanning the original database D, so as to better solve the first bottleneck, to improve the problem of excessive I/O load, reduce the computing time and achieve the goal of optimizing the Apriori algorithm.

Analysis of Apriori Algorithm

Implementation Principle of Apriori Algorithm. As a classic data mining algorithm, Apriori algorithm occupies an important position. The basic idea is as follows:

a). By scanning all the data itemsets in the transaction database, a candidate 1-itemset is obtained, denoted C_1, and the frequency of occurrence of the corresponding data item is counted. According to the pre-defined minimum support, selected frequent 1-itemsets from C_1 are screened as L_1.

b). By frequent self-connection of 1-itemset L_1, a candidate 2-itemset C_2 is obtained, and then the transaction database is scanned again to calculate the corresponding frequency so as to be compared with the minimum support to obtain L_2.

c). Repeat the above steps until there are no longer frequent K-itemsets that meet the requirements.

It can be seen from the above that Apriori algorithm uses the iterative method of layer-by-layer search to generate high-dimensional frequent items through low-dimensional frequent itemsets. Each mining a frequent item set L_k, you need to scan the entire transaction database. In the process, we also generate a large number of candidate sets, making Apriori algorithm less efficient.

Key Steps of Apriori. The Apriori algorithm is an iterative way to find the L_k by the L_{k-1}, so the key steps are: connection, pruning.

a). Connection Step: In order to obtain the L_k, L_{k-1} needs to be connected with itself to generate a candidate -K item set, and the premise of performing is that the elements of th e L_{k-1} are connectable.

b). Prune Steps: C_k members may or may not be frequent, but all frequent K-itemsets ar e included in the C_k. The C_k is determined by scanning the database to determine the freq uency of each candidate in the L_k.

Inefficiency reasons of Apriori algorithm. Apriori algorithm can get frequent itemsets more efficiently, but there are some shortcomings.

a). Whenever mining K frequent itemsets, the transaction database must be scanned multi ple times to get the corresponding support count, which will inevitably lead to long time c onsumption.

b). A lot of candidate sets will be generated. When the L_{k-1} gets a L_k by self-connectio n, a large number of candidate sets may be generated, especially C_2.

The Thought and Realization of Improved Apriori Algorithm

The above analysis shows that the main reason for the inefficiency of Apriori algorithm is that the transaction database D needs to be repeatedly scanned. If the size of the transaction database D is small, the problem is not obvious. But obviously the vast majority of databases that need data mining are data warehouses that have massive amounts of data. Therefore, this problem will become cannot be ignored. The Apriori algorithm is also optimized for the sake of streamlining the transaction database D being scanned every time a frequent item set is generated through a candidate set.

The basic idea of the optimization algorithm is to generate a candidate transaction database A_k after the first scan of the original database D. In the process of generating frequent itemsets, each time the original transaction database is no longer scanned, the candidate transaction database A_k is scanned, so as to achieve the purpose of improving the efficiency of the algorithm. The elements in candidate transaction database A_k are denoted as $<TID,\{X_k\}>$, where X_k represents a frequent set of X-items that may be present. A_k specifies the following: A_k = D when k = 1. When K> 1, the set A_k = transaction S, $<s.TID,\{C \in A_k \mid C \subset s\}>$.

The pseudo-code for generating frequent itemsets L_k in the improved Apriori algorithm is as follows:

Enter: Database D, the minimum support min-sup
Output: frequent itemset L in database D.

A_1:=Database D;

$L_1 := \{C \in A_1 \mid C.count \geq \min-\sup\}$;// min-sup stands for the minimum support

K:=2;

while (L_k -1) do

{ C_k =Apriori_hx (L_{k-1}) ; // Apriori_hx () is a candidate project set generation function C_k =F;

for all $S \in A_{k-1}$ do

$\{C_s = \{C \in C_k \mid c-c[k] \in s.set-itemsets \wedge c-c[k-1] \in s.set-itemsets\}$ if($C_s \neq$F) then $A_k += < s.TID, C_s >$;
/}
$L_k := \{C \in C_k \mid C.count \geq \min-\sup\}$;
/}
D_items:= $U_k L_k$;// D_items represents all the big item sets

Application of Improved Apriori Algorithm

There is a database D, things |D|=4, the minimum support count min-sup=2, the Apriori algorithm and the Apriori Apriori algorithm Apriori algorithm described in detail below, of which Fig. 1 is Apriori algorithm operation graph, Fig. 2 is Apriori algorithm operation process diagram.

Database D			A_1			L_1	
Transaction	Items		item-sets	Support		item-sets	Support
K_1	B_1, B_2, B_3, B_4	ScanD	$\{B_1\}$	3	Crude	$\{B_1\}$	3
K_2	B_1, B_3		$\{B_2\}$	2		$\{B_2\}$	2
K_3	B_1, B_3, B_4		$\{B_3\}$	4		$\{B_3\}$	4
K_4	B_2, B_3, B_5		$\{B_4\}$	2		$\{B_4\}$	2
			$\{B_5\}$	4			

Figure 1. Finite Frequent itemsets L_1 is generated by candidate frequent itemsets C_1

C_2		A_2			L_2	
item-sets		item-sets	Support		item-sets	Support
$\{B_1, B_2\}$	Scan D	$\{B_1, B_2\}$	1	Crude	$\{B_1, B_3\}$	3
$\{B_1, B_3\}$		$\{B_1, B_3\}$	3		$\{B_1, B_4\}$	2
$\{B_1, B_4\}$		$\{B_1, B_4\}$	2		$\{B_2, B_3\}$	2
$\{B_2, B_3\}$		$\{B_2, B_3\}$	2		$\{B_3, B_4\}$	2
$\{B_2, B_4\}$		$\{B_2, B_4\}$	1			
$\{B_3, B_4\}$		$\{B_3, B_4\}$	1			

Figure 2. Finite Frequent itemsets C_2 generate frequent itemsets L_2

C_3	ScanD	A_3		Crude	L_3	
item-sets		Transaction	item-sets gather		item-sets	Support
$\{B_1, B_3, B_4\}$		$\{B_1, B_3, B_4\}$	2		$\{B_1, B_3, B_4\}$	2

Figure 3. Finite The frequent itemsets L_3 is generated by the candidate frequent itemsets C_3

Database D		Scan D	A_1		Scan A_1	L_1	
Transaction	Items		Transaction	item-sets gather		item-sets	Support
K_1	B_1, B_2, B_3, B_4		$\{K_1\}$	$\{B_1\}, \{B_2\}, \{B_3\}, \{B_4\}$		$\{B_1\}$	3
K_2	B_1, B_3		$\{K_2\}$	$\{B_1\}, \{B_3\}$		$\{B_2\}$	2
K_3	B_1, B_3, B_4		$\{K_3\}$	$\{B_1\}, \{B_3\}, \{B_4\}$		$\{B_3\}$	4
K_4	B_2, B_3, B_5		$\{K_4\}$	$\{B_2\}, \{B_3\}, \{B_4\}$		$\{B_4\}$	2

Figure 4. Finite The frequent item set L_1 is generated by the candidate transaction database A_1

C_2	Scan D	A_2		Scan A_2	L_2	
item-sets		Transaction	item-sets gather		item-sets	Support
$\{B_1, B_2\}$		$\{K_1\}$	$\left\{\begin{matrix}\{B_1, B_2\}, \{B_1, B_3\}, \{B_1, B_4\}, \\ \{B_2, B_3\}, \{B_2, B_4\}, \{B_3, B_4\}\end{matrix}\right\}$		$\{B_1, B_3\}$	3
$\{B_1, B_3\}$		$\{K_2\}$	$\{B_1, B_3\}$		$\{B_1, B_4\}$	2
$\{B_1, B_4\}$		$\{K_3\}$	$\{B_1, B_3\}, \{B_1, B_4\}, \{B_3, B_4\}$		$\{B_2, B_3\}$	2
$\{B_2, B_3\}$		$\{K_4\}$	$\{B_2, B_3\}$		$\{B_3, B_4\}$	2
$\{B_2, B_4\}$						
$\{B_3, B_4\}$						

Figure 5. Finite The frequent itemsets L_2 is generated by the candidate transaction database A_2

C_3	Scan D	A_3		Scan A_3	L_3	
Transaction		Transaction	item-sets gather		item-sets	Support
$\{B_1, B_2, B_3\}$		K_1	$\{B_1, B_2, B_3\}$		$\{B_1, B_2, B_3\}$	2
		K_3	$\{B_1, B_2, B_3\}$			

Figure 6. Finite Frequent itemsets L_3 is generated by the candidate transaction database A_3

Although both the Apriori algorithm and the improved Apriori algorithm generate the frequent itemsets in three steps, the Apriori algorithm scans the original database three times, whereas the improved Apriori algorithm only scans the original database in the first step, The second and third step scan are the introduction of the A_k that is, the candidate transaction database, we can see that if the value of K is very large, the number of A_k will be much smaller than the original database, this will solve the I/O load is too large Reduce the computation time and achieve the goal of optimizing Apriori algorithm.

Experimental Results

In order to more intuitively verify that the improved algorithm priori algorithm compares the superiority of Apriori in performance, performance testing experiments are conducted on the same computer. The software and hardware configuration of the computer is as follows: Hardware Configuration: Processor: Intel (R) Core (TM) i3-3110M CPU @ 2.40GHz, memory: 2.0GB.Software Platform: Windows 7 Ultimate, Microsoft SQL Server 2005, Microsoft Visual Basic 6.0

The database used in this experiment is a teacher evaluation database of Jiangxi Institute of Industry and Trade. Random sampling of teachers of different grades and different professions in the whole hospital is carried out and contains 600 sets of data. The performance test of the two algorithms adopts fixed minimum support and Transform the method of minimum support for testing.

Fixed Minimum Support. Set the minimum support min-sup = 20%, candidates frequent k-item set respectively C_1, $C_2 \ldots C_k, C_{k+1}$ ..., Apriori, improved Apriori two algorithms to generate candidate frequent k- Shown:

Table 1 The time required for both algorithms to generate candidate frequent k-itemsets (ms)

Candidateitem-sets	C_1	C_2	C_3	C_4	C_5
Apriori Algorithm	163	176	245	192	141
Improved Apriori Algorithm	164	92	129	87	0

It can be seen from the figure above that in the case of fixed minimum support, except for the time required to generate candidate frequent K-sets, the two algorithms are basically the same, and the time-modified Apriori algorithm for generating candidate frequent itemsets is obviously lower than Apriori algorithm, which shows that the improved algorithm significantly reduces the computing time, performance is higher than the Apriori algorithm.

Transform the Minimum Support. The minimal support min-sup is set to be 10%, 15%, 20%, 25%, 30% respectively. Apriori and Apriori are used to generate candidate frequent k-itemsets. To make the results more accurate, The algorithms are run three times to take the average time, using different support each operation time required as shown in Table 2 below:

Table 2. Transformation support two algorithms to generate frequent candidate K-term sets the timetable (ms)

Support	Algorithm	First operation time	Second operation time	the third operation time	Average operation time
10%	Apriori	3085	3127	3096	3098
	Improved Apriori	1172	1198	1205	1193
15%	Apriori	1639	1708	1729	1696
	Improved Apriori	862	862	856	848
20%	Apriori	918	923	939	928
	Improved Apriori	519	509	524	517
25%	Apriori	759	747	765	759
	Improved Apriori	308	315	318	317
30%	Apriori	428	437	419	428
	Improved Apriori	219	209	226	217

It can be seen from the above figure that the algorithm of transformation support is used for testing. The improved Apriori algorithm has obvious advantages over Apriori algorithm. When the support degree is smaller, the performance of the improved algorithm is more prominent, and the advantage is more obvious.

Summary

Based on the deep research of Apriori algorithm, this paper points out the limitations of Apriori algorithm and proposes a new improved algorithm. The improved algorithm only needs to scan the database once to achieve the same operation effect as the Apriori algorithm, and avoids the repeated judgment between frequent itemsets and directly obtains a higher candidate frequent itemset, thereby greatly improving the algorithm efficiency. Experimental results also show that the algorithm does improve the efficiency of Apriori algorithm. It is believed that this is of some significance to the problems of lack of information for solving the data explosion facing the present era.

References

[1] Ho Yue Shun. Research and Application of Association Rules Mining Technology [D]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2010.
[2] Jiawei Han, Micheline Kamber. Data Mining Concepts and Techniques [M]. Beijing: Mechanical Industry Press, 2006.
[3] Lin Jiaxiong, yellow war. Improved Apriori Algorithm Based on Array Vector [J]. Computer Applications and Software, 2011,28 (5): 268-271.
[4] Peng L, Shizhao N, Zheng W, Ziwei J, Jianwu Y, Zhongxiang Q, Wangmo P. Predicting durations of online collective actions based on Peaks' heights [J]. Communications in Nonlinear Science and Numerical Simulation. 2018, 55: 338-354.
[5] Liu Yuwen. Research and improvement of Apriori algorithm based on cross linked list [J]. Computer Applications and Software, 2012,29 (5): 267-269.
[6] Zhang Chunsheng, Song Linlin. Segmentation support Apriori algorithm and its application [J]. Computer Engineering and Applications, 2010,46 (16): 157-159.
[7] Tan Pangning, Steinbach M, Kumar V. Introduction to Data Mining [M]. Beijing: Mechanical Industry Press, 2011.