Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 2, No. 3, 2022

Improvement of Association Algorithm Based on Matrix Optimization

Yang Chen

School of Information, Yunnan University of Finance and Economics, Kunming, Yunnan 650221, China
511926706@qq.com

Abstract: In order to solve the problems of high memory usage and low efficiency of frequent itemsets generation in the process of data mining, the existing association algorithm is improved. The data information is expressed in the form of a Boolean matrix, and the matrix is continuously simplified in the operation to optimize the item set connection method, thereby reducing memory consumption and improving algorithm efficiency.

Keywords: Frequent itemsets; Boolean matrix; Simplified matrix; Optimal connection.

1. Introduction

The earliest concepts related to data mining were put forward by R. Agrawal, Swami and Imielinskih [1] in 1993. They analogized the connection rules between users in different transactions and generated consumer behavior patterns. Then Apriori algorithm is proposed, which is the most classical algorithm in the field of association rules. Toivonen [2] and other researchers optimized the generation process of frequent itemsets in the classic Apriori algorithm through tests. YJHAN [3] [4] and other researchers improved the traditional algorithm and proposed the FP- growth algorithm, which puts the data in the database into the nodes of the tree in turn according to rules, avoiding repeated scanning of the database during the mining process.

2. Basic concepts and principles of association algorithms

2.1. Basic concepts

The main purpose of association algorithm is to find out the connection between data in a given data set, that is, strong association rules [5]. At the beginning of the algorithm, the user defines the minimum support and minimum confidence according to the actual situation.

Support (count): The number of times customers purchase some products at the same time is called support count, and the ratio of the count to the number of times of all products in database D is called support. The threshold for measuring whether the customer purchases the product frequently is called the minimum support (count), which is recorded as min_support.

Confidence degree: Under the condition that customers purchase some commodities A at the same time, the probability of purchasing other commodities B at the same time, that is, the conditional probability P(B|A) of the rule (or implication) A⇒B, is recorded as confidence(A= >B), the threshold for measuring the frequent purchase of some products by customers and the frequent purchase of other products is called the minimum confidence, which is recorded as min_confidence.

2.2. Algorithm principle

Scan the database D, extract and count each product, and generate C1 item set . C1 is a special candidate item set, which can directly calculate the support count of each item in the item set , and delete the item set whose support count is less than the minimum support count . Generate frequent itemsets L1 .

Connection step: In order to find the frequent itemset Lk+1, connect Lk with itself to generate a set of candidate k+1 itemsets (Lk▷Lk⇒(generate)Ck+1), set Lk={l1,l2,.. .,lm} l1={a1,a2,...,ak} l2={b1,b2,...,bk} only guaranteed condition a1=b1,a2=b2,...,ak-1=bk -1,ak<bk. The resulting item set produced by joining l1 and l2 is {a1,a2,...,ak-1,ak,bk} or {b1,b2,...,bk-1,ak,bk}.

Pruning step: divided into two steps (1) Find all the largest non-empty proper subsets, scan Lk, because a new element is added, a new combination is generated, scan Lk to know whether it is frequent, according to the prior nature , any infrequent k-itemset is not a subset of the frequent k+1-itemset, therefore, if the k-item subset of a candidate k+1-itemset is not in Lk, the candidate cannot be frequent, so Can be deleted from Ck+1. (2) Scan the database D again to obtain the count, delete the itemsets whose support count is less than the minimum support count, and obtain the frequent itemset Lk+1.

The connection step and the pruning step are cyclically scanned until Lk cannot self-connect to generate a set of candidate k+1 itemsets or an infrequent itemset, the loop iteration stops, and all frequent itemsets are found.

3. Improvement of association algorithm

3.1. Algorithm improvement ideas

Existing association algorithms have certain limitations [6]. On the one hand, the association algorithm will continue to read the database information when obtaining frequent itemsets, which makes the algorithm very inefficient; on the other hand, the association algorithm can generate new candidate itemsets through the connection step, but it will generate too many duplicate itemsets. It is necessary to compare the generated itemsets with the database to obtain frequent itemsets, which consumes a lot of time and space. Therefore, the algorithm makes the following improvements from these two points: (1) Record data in the form of a matrix and combine the idea of the connection step of the classic

Apriori algorithm in the association algorithm to group frequent itemsets so that the connection generates N+1-dimensional candidates Ensure that no duplicates are generated during the set, improving the efficiency of the algorithm. (2) Constantly pruning and optimizing the itemset matrix during the algorithm operation, thereby reducing the consumption of memory and operation time.

3.2. Algorithm process

enter:

D: transactional database

min_sup: minimum support threshold

Output: L: frequent itemsets in D

Algorithm flow:

(1) Read the transaction information sequentially from the database, and perform certain processing on it, and convert it into a Boolean value, 1 if it exists, and 0 if it does not exist, and generate a Boolean matrix D1 corresponding to the database D. Calculate the support count corresponding to each item, if it is less than min_sup, delete that column, and the remaining item set is the frequent item set, and get the frequent item set matrix D1 (at this time, the matrix is deleting non-frequent items set while rotating it into a row representing the item set, which is convenient for subsequent code processing).

(2) Perform an AND operation on the matrix D1 with the subsequent rows starting from the first row, and obtain the combined support count at the same time as the operation. If it is less than min_sup, it will be discarded directly. If it is greater than min_sup, it will be retained and used to generate new matrix D2. In this process, the newly generated vectors with a support count greater than min_sup should be grouped into new groups. For example, the first row and the subsequent row are grouped together, the second row and the subsequent row are grouped together, and so on. .

(3) Find frequent k-itemsets (k>2). Starting from the frequent binomial set, put the frequent binomial set matrix into a queue in order according to the divided groups. If the number of vectors in the group is less than 2, just give up (because the AND operation cannot be performed to generate a new vector), and then Carry out each group in turn according to step 2, and then continue to put it at the end of the queue after a new group is generated. When the queue is empty, it means that no new frequent itemsets can be generated, exit the loop, and find all frequent itemsets.

4. Algorithm Analysis and Experimental Comparison

4.1. Algorithm analysis

Compared with the Apriori algorithm and existing improved methods, this algorithm has the following advantages:

This algorithm records the data read from the database in the form of a matrix. Support counting only needs to perform an AND operation on the matrix vector without scanning the database multiple times.

In the process of operation, a new frequent item set, that is, a new matrix vector, will be generated to replace the previous vector, thereby continuously reducing and compressing the matrix and improving memory utilization.

Group the newly generated frequent itemsets starting from frequent binomial sets, such as {I1, I2, I3} {I1, I2, I4} into a group {I1, I3, I5}, {I1, I3, I6} Divided into a group, this is done by referring to the connection step of the original Apriori algorithm, which can make the newly generated itemset unique and orderly, avoid generating duplicate item sets, thus saving a lot of time.

When putting the newly generated item set group into the queue, if the number of items sets in the item set group is less than 2, it will be discarded, which has a certain pruning effect and saves a certain amount of time.

4.2. Example analysis

The following is an example to analyze the data mining process of the improved algorithm. The data set selected in this research comes from the default of credit card clients Data Set data set in the UCI data set. Here, 9 of the data are intercepted for instance analysis, and the minimum support is set to 3. The data are shown in the table below.

Table 1. Example analysis data set

TID (transaction)	item set
T1	I1,I2,I3
T2	I2,I4,I5
T3	I1,I2,I3,I6
T4	I2,I4,I6
T5	I2,I3,I4
T6	I1,I3,I4
T7	I1,I2,I3,I6
T8	I3,I4
T9	I1,I3,I4,I5

(1) Read the database in turn, set the items that exist in the transaction to 0, and set the items that do not exist to 1. For example, the items read from transaction T1 are I1, I2, and I3, and the matrix vector obtained is 111000. Read All things generate the following Boolean matrix:

$$D=\begin{bmatrix}1&1&1&0&0&0\\0&1&0&1&1&0\\1&1&1&0&0&1\\0&1&0&1&0&1\\0&1&1&1&0&0\\1&0&1&1&0&0\\1&1&1&0&0&1\\0&0&1&1&0&0\\1&0&1&1&1&0\end{bmatrix}$$

Calculate the support count of the grid column as shown in the following table:

Table 2. Support count

item set	{I1}	{I2}	{I3}	{I4}	{I5}	{I6}
support count	5	6	7	6	2	3

It can be seen that the support count of the itemset {I5} is 2, which is less than the minimum support count of 3, so the itemset {I5} is an infrequent itemset, if it is deleted, the frequent item set is {{I1},{ I2}, {I3}, {I4}, {I6}}, generate a new frequent item set matrix as shown in the figure below, where the rows and columns of the matrix have been converted, which is a matrix converted while pruning, without Extra time consuming.

$$D=\begin{bmatrix}1&0&1&0&0&1&1&0&1\\1&1&1&1&1&0&1&0&0\\1&0&1&0&1&1&1&1&1\\0&1&0&1&1&1&0&1&1\\0&1&0&0&0&0&0&0&1\\0&0&1&1&0&0&1&0&0\end{bmatrix}\begin{matrix}I1\\I2\\I3\\I4\\I5\\I6\end{matrix}$$

(2)The candidate binomial set vector is obtained by performing AND operation on the frequent item set matrix from the first row vector, such as the first row {I1} and the second row {I2} to generate {I1,I2}= {1,0,1,0,0,0,1,0,0}, and its support count is calculated as 3, which meets the requirements of the minimum support count, and {I1,I2} is a frequent binomial set. The following steps can be followed to obtain the frequent binomial set matrix:

$$D=\begin{bmatrix}1&0&1&0&0&0&1&0&0\\1&0&1&0&0&1&1&0&1\\1&0&1&0&1&0&1&0&0\\0&1&0&1&1&0&0&0&0\\0&0&0&0&1&1&0&1&1\end{bmatrix}\begin{matrix}\{I1,I2\}\\\{I1,I3\}\\\{I2,I3\}\\\{I2,I4\}\\\{I3,I4\}\end{matrix}$$

(3)While obtaining the frequent binomial set matrix, group the vectors of the matrix, {I1,I2}{I1,I3} into a group, {I2,I3}{I2,I4} into a group, {I3, I4} Divide into one group, record the number of vectors in each group, create a vector queue queue, and put the divided groups with the number of vectors greater than or equal to 2 into the queue in turn, as shown in the following table:

Table 3. Vector grouping

0	1
{I1, I2} {I1, I3}	{I2, I3} {I2, I4}

(4) Enter the loop when the queue is not empty, take out the vector group at the top of the queue, and execute step (2). Get a new itemset {I1, I2, I3}, and calculate the support count of the second frequent itemset to be 3, so this item set is a frequent itemset. However, since there is only one vector generated by this group, new vectors cannot be generated by connecting within the group at this time, so it does not need to be placed at the end of the queue. Loop in this way until the queue is empty, exit the loop, and find all frequent itemsets.

4.3. Analysis of mining results

In order to compare the efficiency of the improved algorithm and the classic algorithm, this paper adopts the method of controlling variables, respectively controlling the amount of transactions and the basis of support, controlling the number of items and counting of support, and controlling the amount of transactions and the number of items. These three experiments verify that the improved algorithm is running There are certain advantages in terms of efficiency.

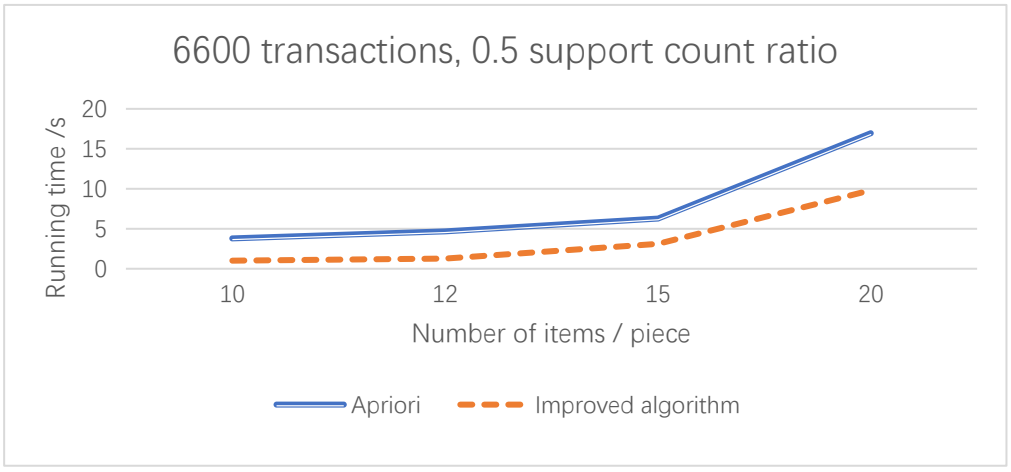

Figure 1. Comparison of efficiency of different item numbers

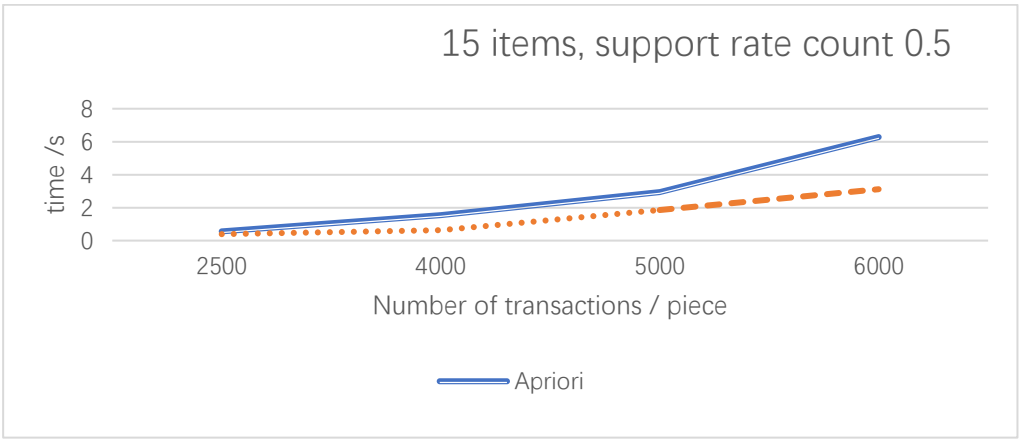

Figure 2. Comparison of the number of items of different things

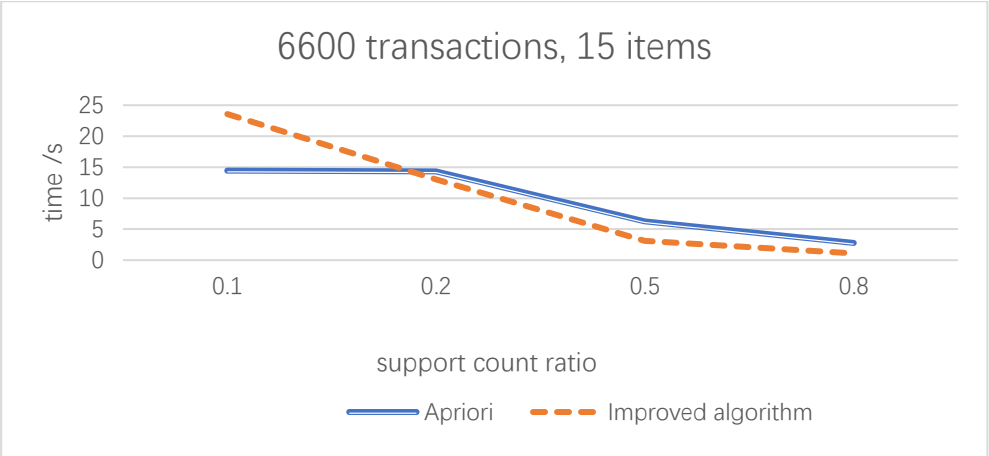

Figure 3. Comparison of counting efficiency with different support degrees

It can be seen from Figure 1 and Figure 2 that the improved algorithm is more efficient than the classical algorithm. However, in the case of the same transaction volume and number of items in Figure 3, the efficiency of the improved algorithm is lower than that of the classical algorithm when the support count is low. This may be because the improved algorithm is represented by a matrix. When the support count is relatively low, it will make The amount of matrix simplification in the connection process is small, which increases the amount of calculation and thus increases the operation time. In the follow-up improvement, it can be considered to carry out targeted processing on the case where the number of items is large and the number is large. For example, when the number of items and the number are large, the generated data matrix is likely to be a sparse matrix. Without changing the original matrix Optimizing the matrix in the absence of meaning should greatly improve efficiency.

5. Conclusion

This paper has made some improvements to Apriori, which effectively enhances the efficiency of the algorithm and reduces the time and space consumption of the algorithm, but there are still some defects, such as matrix redundancy under a large amount of data, etc., and then we will address these deficiencies Further improvement, gradually improve the improved algorithm, and at the same time think about the applicable direction of the algorithm to improve the flexibility of the algorithm application.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. Proc, 1993 ACM-SIGMOD Int, Conf. Management of Data (SIGMOD' 9 3), pp. 207-216, Washington, DC, May 1993.

[2] M. Zaki. Scalable algorithms for association mining. IEEE Trans. Knowledge and Data Engineering, 12:372-390, 2000.

[3] Ian H. Witten, Eibe Frank and Mark A. Hall. Data Mining: Practical Machine Learning Tools and Techniques [M]. China Machine Press. 2014.

[4] Jiawei Han, Micheline Kamber, Jian Pei, Data Mining Concepts and Techniques Third Edition [M]. China Machine Press. 2019.

[5] Cui Yan, Bao Zhiqiang. A survey of association rule mining[J]. Application Research of Computers, 2016,33(02):330-334.

[6] Cai Weijie, Zhang Xiaohui, Zhu Jianqiu, Zhu Yangyong. A Survey of Association Rules Mining [J]. computer engineering ,2001(05): 31- 33+ 49.