IJCES International Journal of Computer Engineering Science , Volume1 Issue 3, December 2011
ISSN : 2250:3439
https://sites.google.com/site/ijcesjournal
http://www.ijces.com/

An Evolutionary Algorithm for Mining Association Rules Using Boolean Approach

G.Ravi Kumar[1] Dr.G.A. Ramachandra[2] G.Sunitha[3]

1. Research Scholar, Department of Computer Science &Technology, S K University, Anantapur,
2. Associate Professor, Department of Computer Science & Technology, S K University, Anantapur
3. Research Scholar, Rayalaseema University, Kurnool

ABSTRACT

Frequent pattern mining is one of the active research themes in data mining. It plays an important role in all data mining tasks such as clustering, classification, prediction, and association analysis. Identifying all frequent patterns is the most time consuming process due to a massive number of patterns generated. A reasonable solution is identifying efficient method to finding frequent patterns without candidate generation. In this paper, we present An Evolutionary algorithm for mining association rules using Boolean approach for mining association rules in large databases of sales transactions. Traditional association rule algorithms adopt an iterative method to discovery, which requires very large calculations and a complicated transaction process. Because of this, a new association rule algorithm is proposed in this paper. This new algorithm scanning the database once and avoiding generating candidate itemsets in computing frequent itemset and adopts a Boolean vector "relational calculus" method to discovering frequent itemsets. Experimental results show that this algorithm can quickly discover frequent itemsets and effectively mine potential association rules.

Keywords: Data mining, Association rule, Frequent itemsets, Boolean item table

1. Introduction

Knowledge discovery in databases (KDD) is defined as the non-trivial extraction of valid, implicit, potentially useful and ultimately understandable information in large databases [1]. For several years, a wide range of applications in various domains have benefited from KDD techniques and many works has been conducted on this topic. The problem of mining frequent itemsets arose first as a sub-problem of mining association rules [2].

Association rule mining is one of the most important techniques of data mining. It aims to extract interesting correlations, frequent patterns, associations or casual structures among a large set of data items. A typical application is market basket analysis, which studies the buying habits of customers by searching for sets of items that are frequently purchased together [1]. Other application areas include customer segmentation, store layout, web usage mining, software defect detection, telecommunication alarm prediction, and bioinformatics.

2. Association rules

2.1. Problem definition

Association rule mining is a data mining method to find the interesting association or correlation among a large set of data items. A formal statement of the association rule mining problem is as follows [2]. Let { I=□I_i, I_2,…I_m } be a set of items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq$ □I .Associated with each transaction is a unique identifier, called TID. A transaction T contains X, a set of items in I, if $X \in$ □T .An association rule is an implication of the form $X \Rightarrow Y$, where $X \subset$ □□I ,$Y \subset$ □□I , and $X \cap Y = \phi$ □. The rule $X \Rightarrow Y$ holds in the transaction set D with confidence C if C% of the transactions in *D* that contain X also contain Y. The rule $X \Rightarrow Y$ has support S in the transaction set D if S% of the transactions in D contain $X \cup Y$. Confidence determines the strength of the rule and support measures the frequency of the occurring pattern. A set of items is referred to as *itemset*. An itemset that contains *k* items is a k-itemset. The occurrence frequency or count of an itemset is the number of transactions that contain the itemset. If an itemset has a transaction support higher than a user-specified minimum support threshold, it is a frequent itemset. Given a set of transactions, *D*, the problem of association mining is to find strong rules with support and confidence greater than the given minimum support and confidence thresholds, respectively. An association rule discovery algorithm can be decomposed into two successive stages [3]. In the first stage, all sets of frequent items are discovered. In the second stage, rules are derived from these itemsets. It is important to generate all itemsets efficiently. An efficient association rule discovery algorithm is discussed in the next section.

In this paper, we proposed a new algorithm based on Boolean method called An Evolutionary algorithm for mining association rules using Boolean approach. This algorithm transforms a transaction database into a Boolean data stored in bits using bit streams. Meanwhile it uses the Boolean vector "relational calculus" method to discover frequent itemsets. We use the fast and simple "and calculus'' in the Boolean table to replace the calculations and complicated transactions that deal with large numbers of itemsets. Experimental results show that this algorithm is more effective than the Apriori-like algorithms.

3. Related works

The Apriori algorithm[3] proposed by Agrawal et al. is a classical algorithm for association rules mining. The name of the algorithm comes after a prior knowledge about frequent itemsets was used. The prior knowledge is that any non-empty subset of a frequent itemset is also frequent. Apriori algorithm uses a level-wised and iterative approach, it first generates the candidates then test them to delete the non-frequent itemsets. Most of previous studies adopted an Apriori-like candidates generation-and-test approach.

Han et al. developed the FP-growth algorithm[4] that is based on frequent pattern tree. Comparing with Apriori algorithm, this algorithm has following features. (1) It uses FP-tree to store the main information of the database. The algorithm scans the database only twice, avoids multiple database scans and reduces I/O time. (2) It does not need to generate candidates, reduces the large amount of time that is consumed in candidates generation and test. (3) It uses a divide-and-conquer approach in the mining process, so the searching space is significantly decreased. The efficiency of the Fpgrowth algorithm is about an order of magnitude faster than the Apriori algorithm.

However, there still exist some aspects in FP-growth algorithm that can be improved. For example, it needs to recursively generate huge number of conditional FP-trees that consumes much more memory and more time. It has appeared several improved FP-growth algorithms based on original one. Fan and Li [5] presented a constrained subtree based approach to avoid the generation of huge number of conditional FP-trees recursively in the mining process. They also reduced the fields in each FP-tree node. Their approach has better time and space scalability than FP-growth algorithm. Grahne and Zhu [5] proposed an array-based technique to reduce the time cost in FP-tree traverse. They also implemented their own memory management for allocating and deallocating tree nodes.Qin et al. combined several advanced techniques above, presented a compact FP-tree based frequent patterns mining algorithm, CFPmine [6]. It uses constrained subtrees of a compact FP-tree to mine frequent pattern, so that it is doesn't need to construct conditional FP-trees in the mining process, so the memory cost is reduecd. It also uses an array-based technique to reduce the time on traverse of the CFP-tree, the efficiency is improved.

4. Proposed Algorithm

Amount of data in the database is increasing day-by-day. So, the process of generating frequent itemsets turns out to be the bottleneck in mining association rules. Therefore, researchers have focused on developing efficient and effective algorithms for the generation of frequent itemsets.

Our algorithm is an effective algorithm for mining association rules in large databases .Like the Apriori algorithm, our algorithm mines association rules in two steps. In the first step compute frequent itemsets using logic OR and AND operations . In the second step, to derive all interesting association rules based on the computed frequent itemsets using logic AND and XOR operations. By scanning the database only once and avoiding generating candidate itemsets in computing frequent itemsets, the Implemented algorithm gains significant performance improvement over the Apriori algorithm.

4.1 Generation of Frequent Itemsets

The implemented algorithm generates frequent itemsets through evolutionary iterations based on two tables, the item details table and the transaction table. Section 4.1.1 describes the transforming transaction data into boolean data. Then, Section 4.2 describes the process of generating frequent $k-$ *itemsets.*

4.1.1 Transforming a transaction details into a Boolean Table

The Boolean table is a matrix with element values of '1' or '0' ,where items are present in the transaction means 1 otherwise 0.Finally, a column vector C_k is utilized to store the reference count of all frequent *k-itemsets* in the kth iteration.The reference count on a *k-itemset* can be obtained by counting the number of l's in the corresponding row of boolean table .

4.2 Generation of Frequent *k-itemsets*:

Frequent *k-itemsets* can be generated through the following iteration:

Repeat

1. Read a pair of different rows from a Boolean table .

2. Applying OR operation on these two rows will get a new temporary itemset. If the temporary itemset contains more than k different items or is already produced by a previous OR operation, proceeding to step l (i.e., ignore this new itemset); otherwise, go to step 3 (i.e., until a new *k-itemset* has been found).

3. Performing AND operation on the two rows of Boolean table , correspond to the rows of step2. The result shows that, which transactions contain this new *k-itemset*. And then counting the number of 1's in the result to get the reference count of this new *k-itemset*. If the count is less than the number of transactions required by the minimum support, the new *k-itemset* is discarded.

After the generation of frequent *k-itemset*, the Boolean table of the *k-itemset* and its corresponding reference count vector are kept in frequent itemset table for generating association rules.

4.3 Generation of Association Rules

In this section, the way that our algorithm mines association rules from the final frequent itemsets table is presented.

Based on the above-mentioned observation, the implemented algorithm mines an association rule by first identifying the potential antecedent and potential consequent, and then validates if such a rule satisfies the minimum confidence requirement. The basic ideas for mining association rules are described as follows.

1. Elimination of the rows of *1-itemsets* from frequent item table which have no opportunity to be an antecedent or a consequent of any association rules. The algorithm simply counts the occurrences of 1s in each column of Boolean t*able*. A *1-itemset* whose corresponding column has only one 1 in the whole column should be eliminated.
2. Performing an AND operation on two rows, say X and Z, of item details in frequent item table , and then comparing the results with the one which has less number of items, assuming it is X. If they are equal, then the frequent itemset with less number of items (i.e., X in this case) is a potential antecedent.
3. Applying an XOR operation on the two rows chosen in step 2. The result, denoted as Y, of the XOR operation constitutes a potential consequent with X being its corresponding antecedent.
4. If support (Z) / support(X) is greater than or equal to the minimum confidence, then the association rule X →Y is generated.
Step 2 through step 4 is referred for any combination of X and Z, until no new rules is found. Number of association rules, are created, to store the antecedent X, consequent Y support and confidence for each association rule X→Y.

If the association rule X →Y holds, then all X, Y and X *U* Y must be frequent itemsets. Since X *U* Y contains both X and Y, it can be inferred that if a frequent itemset is not a subset of any another frequent itemset in frequent item details, then it can be neither an antecedent nor a consequent of any association rule. This observation is the foundation of the implemented algorithm in mining association rules. To expedite the mining process, from the frequent item table, those frequent itemsets that are not subset of any other frequent itemsets can be eliminated at first. However, it has been found that only frequent *1-itemsets* are candidates for the elimination.

4.4. ILLUSTRATIVE EXAMPLE

Let $I = \{i_1, i_2, \cdots, i_n\}$ be a set of items. Given a set of sales transactions D, where each transaction T is a subset of *I*, an association rule is an expression of the form

$$X \rightarrow Y$$

Where X and Y are subsets of I. An association rule X →Y holds in the transaction set D with a confidence c and support s.

Sample database D in figure 4.1 is taken for consideration. The database has 4 sales transactions with 5 items, i.e. {Bread, Milk, Jam, Sugar, and Butter}.

All items have individual code numbers, i.e.

Item	Code
Bread	1
Milk	2
Jam	3
Juice	4
Coke	5

Figure 4.1 Items with code

Transaction T1 of the sample database D shown in figure 4.2 states that items 1, 3 and 4 were purchased. According to the figure 4.1, item codes 1, 3 and 4 significantly show items Bread, Jam and Sugar.

TID	Items
T1	1,3,4
T2	2,3,5
T3	1,2,3,5
T4	2,5

Figure 4.2 Transactional Database

4.4.1 Transforming a transaction details into a Boolean Table :

The item details table , transaction table and Count(C) are first initialized.

	1	2	3	4	5	T1	T2	T3	T4	C

1	1	0	0	0	0	1	0	1	0	2
2	0	1	0	0	0	0	1	1	1	3
3	0	0	1	0	0	1	1	1	0	3
4	0	0	0	1	0	1	0	0	0	1
5	0	0	0	0	1	0	1	1	1	3

Figure 4.3 Boolean table

4.4.2 Generate Frequent k-Itemsets:

After construction of the initial item and transaction table, the corresponding row which doesn't support minimum support requirement is removed.

	1	**2**	**3**	**4**	**5**	**T1**	**T2**	**T3**	**T4**	**C**
1	1	0	0	0	0	1	0	1	0	2
2	0	1	0	0	0	0	1	1	1	3
3	0	0	1	0	0	1	1	1	0	3
5	0	0	0	0	1	0	1	1	1	3

Figure 4.4 frequent-1 item table

4.4.3 Generation of Frequent *k-itemsets*:

By performing OR and AND operations on frequent item table (in Figure 4.2), respectively, frequent k-item table is derived in Figure 4.5(a). It is noted that the itemset {1, 5} is not in table , because it fails to satisfy the minimum support requirement. Similarly, frequent item-3 table is derived from frequent -2 item table which contains only one frequent *3-itemset*, as shown in Figure 4.5(b).

After the iteration is completed, the final frequent items, which is needed for generation of the association rules, is shown in figure 4.6

	1	**2**	**3**	**4**	**5**	**T1**	**T2**	**T3**	**T4**	**C**
1,3	1	0	1	0	0	1	0	1	0	2
2,3	0	1	1	0	0	0	1	1	0	3
2,5	0	0	1	0	1	0	1	1	1	3
3,5	0	0	1	0	1	0	1	1	0	3

a) frequent 2-item table

	1	**2**	**3**	**4**	**5**	**T1**	**T2**	**T3**	**T4**	**C**
2,3,5	0	1	1	0	1	0	1	1	0	2

b) frequent item-3 table

Figure 4.5 Generation of frequent *k-itemsets*

Upon concatenation all Items and count tables to form the final frequent item table. This table will have all frequent itemsets and their counts.

	1	**2**	**3**	**4**	**5**	**C**
1	1	0	0	0	0	2
2	0	1	0	0	0	3
3	0	0	1	0	0	3
5	0	0	0	0	1	3
1,3	1	0	1	0	0	2
2,3	0	1	1	0	0	3
2,5	0	0	1	0	1	3
3,5	0	0	1	0	1	3
2,3,5	0	1	1	0	1	2

Figure 4.6 frequent item table

4.5 Generation of Association Rules

In this section, the way that Boolean algorithm mines association rules from the final frequent itemsets table is presented.

Considering row {2} and row {2, 5} in the frequent item table of Figure 4.6 , by performing bit-wise AND operation on the row {2} and the row {2, 5}, a binary vector (01000) is obtained, which is exactly the same as that of row {2}. Therefore, {2} is an antecedent. By performing a bit-wise XOR operation on row {2} and row {2, 5} itemset {5} is derived as the corresponding consequent. A rule of 2→5 is therefore discovered. By checking the counts of {2, 5} and {2} in frequent item table and performing the necessary computations, the support and the confidence of this rule, which is 75% and 100% is calculated, respectively.

Antecedent	Consequent	Support	Confidence
1	3	50%	100%
2	3	50%	67%
2	5	75%	100%
2	3,5	50%	67%
3	1	50%	67%
3	2	50%	67%
3	5	50%	67%
3	2,5	50%	67%
5	2	75%	100%
5	3	50%	67%
5	2,3	50%	67%
2,3	5	50%	100%
2,5	3	50%	67%
3,5	2	50%	100%

Figure 4.7 Derived Association Rules

5. REULTS

In order to appraise the performance of the proposed algorithm, we conducted an experiment using the Apriori algorithm and our algorithm. The algorithms were implemented in java and tested on a WindowsXP Professional platform. The test database T20I4D100K was generated synthetically by an algorithm designed by the IBM Quest project. The number of items N is set to 1000; |D| is the number of transactions; |T| is the averages size of transactions, and |I| is the average size of the maximum frequent itemsets. Figure 5.1 presents the experimental results for different numbers of minimum supports. The results show that the performance of our algorithm is much better than that of the Apriori algorithm. Moreover, the better the performance efficiency of our algorithm is, the smaller the minimum support. This is because the smaller the minimum support, the more candidate itemsets the Apriori algorithm has to determine, and also the Apriori algorithm's join and pruning processes take more time to execute. However, the our algorithm does not produce candidate itemsets, and it spends less time calculating k-supports with the Boolean item table pruned.

T2014D100K

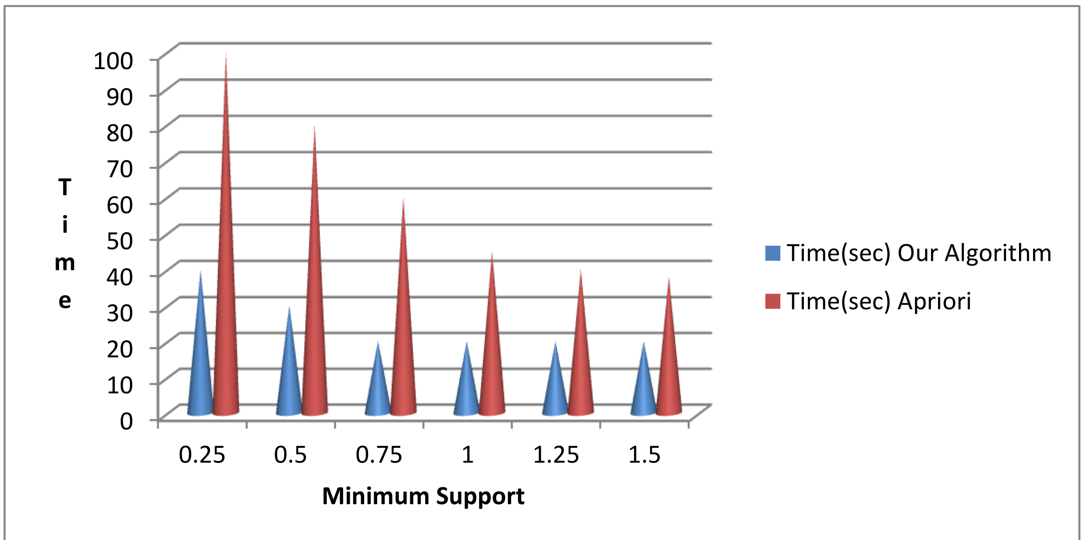

Figure 5.1 Performances of Apriori and our algorithm

6. Conclusions

Association rules are basic data mining tools for initial data exploration usually applied to large data sets, seeking to identify the most common groups of items occurring together. The most common application of association rule mining is market basket analysis. In this paper, An Evolutionary algorithm for mining association rules using Boolean approach is proposed. The main features of this algorithm are that it only scans the transaction database once, it does not produce candidate itemsets, and it adopts the Boolean vector "relational calculus" to discover frequent itemsets. In addition, it stores all transaction data in bits, so it needs less memory space and can be applied to mining large databases.

REFERENCES

[1]. J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, Sept. 2000.
[2]. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of Data (SIGMOD'93)*, pages 207{216. ACM Press, May 1993.
[3] Agrawal, R**.** and Srikant, R**.** Fast algorithms for mining association rules in large databases. In:*Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago de Chile, Chile, 1994,487-499
[4] J. Han, J. Pei, Y. Yin, "Mining frequent patterns without candidate generation".In: M. Dunham, J. Naughton, W. Chen eds. *Proc. of 2000 ACM-SIGMOD Int'l Conf on Management of Data* (SIGMOD'00). Dallas, TX, New York: ACM Press, 2000. pp. 1-12.
[5] M. Fan. and C.Li, "Mining frequent patterns in an FP-tree without conditional FP-tree generation". *Journal of computer research and development*, 2003, 40(8). pp. 1216-1222.
[6] L. Qin, P. Luo, Z. Shi, "Efficiently mining frequent itemsets with compact FP-tree". In: Z.Shi and Q.He eds. *Proc. of Int'l Conf. on Intelligent Information Processing 2004*(IIP2004),Beijing, China. Springer Press, 2004. pp. 397-406.
[7] Agrawal, R., Imielinski, T., & Swami, A. (1993) Mining association rules between sets of items in large databases. Proceedings of the *ACM SICMOD conference on management of data* pp. 207-216. Washington,D.C.
[8] Agrawal, R. & Srikant, R. (1994) Fast Algorithms for Mining Association Rules in large databases. In *Proceedings of the 20th International Conference on Very Large Databases* pp. 487-499..
[9] Han, J., Pei, J., & Yin, Y (2000) Mining frequent patterns Candidate generation. In *Proc. 2000 ACM-SIGMOD Int. Management of Data (SIGMOD'00)*, Dallas, TX.
[10] Klemetinen, L., Mannila, H., Ronkainen, P., et al. (1994) Finding interesting rules from large sets of discovered association rules. *Third International Conference on Information and Knowledge Management* pp.401-407.Gaithersburg, USA.
[11] Kotásek, P. & Zendulka J. (2000) Comparison of Three Mining Algorithms for Association Rules. *Proc. of 34th Spring Int. Conf. on Modelling and Simulation of Systems (MOSIS'2000)*, Workshop Proceedings Information Systems Modelling (ISM'2000), pp. 85-90. Rožnov pod Radhoštěm, CZ, MARQ.
[12] Liu, D. & Kedem, Z. (2002) An Efficient Algorithm for Discovering The Maximum Frequent Set. *IEEETransaction on Knowledge and Data Engineering 14(3)*, 553-566.
[13] Park, J., Chen, M., & Yu, P. (1995) An effective hash-based algorithm for mining association rules. *Proc 1995 ACM-SIGMOD Int. Conf Management of Data* pp. 175-186. San Jose: ACM Press.
[14] Toivonen, H. (1996) Sampling large databases for association rules. *22nd International Conference on Very Large Data Bases* pp. 134–145. Morgan Kaufmann.
[15] Tung, A., Lu, H., Han, J., & Feng, L. (2003) Efficient Mining of Intertransaction Association Rules. *IEEETransaction on Knowledge and Data Engineering 15(1)*, 43-56.