International Journal of Information Technology and Knowledge Management
July-December 2008, Volume 1, No. 2, pp. 267-279

FASTER GENERATION OF ASSOCIATION RULES[1]

B. Nath, D. K. Bhattacharyya & A. Ghosh

Abstract

Extraction of interesting hidden information from a large collection of data is termed as data mining. Association rule mining is a process of data mining where the relationships among the different attributes are extracted. A number of works have been carried out in this area. These algorithms work in two phases; frequent set generation and rule generation. Using a user given parameter minimum support, the first phase finds out the frequent set. This phase is the most time consuming one. Hence this phase has attracted a number of researchers for deriving the frequent sets in an efficient way. The second phase generates the rules using another user given parameter i.e. minimum confidence. For that it uses the frequent sets derived by the first phase. Depending on the output of the first phase this phase may also become time consuming. Moreover if proper care is not taken then we may miss a number of rules. This paper presents a technique to generate all possible rules from the given frequent sets based on the user given minimum confidence.

Keywords*:* Frequent set, association rule, support count, confidence.

1. Introduction

Extracting the hidden information from a large database is very much time consuming job. These types of jobs are commonly termed as data mining [1], [2]. Depending on the type of knowledge tried to explore, data mining techniques are categorized into different categories. Association rule mining is a data mining technique where relationships among different attributes are extracted in an unsupervised way. These rules are of the form "$A \rightarrow C$", representing "if [A] then [C]". The binary database, commonly termed as Market Basket [3] dataset may contain thousands of attributes in it. So finding the relationship among them is a very much challenging job as the search is exponential in nature. At the same time the large size of the database in terms of records brings more difficulty for it. Agrawal *et al*.'s Apriori[3] algorithm can extract these relationships in considerably less time. That algorithm works in two phases. In the first phase depending on a user given parameter minimum support, frequent item sets are derived[3][4][5]. For that the database has to be scanned a number of times. Since the database is too large, scanning it multiple numbers of times will waste a significant amount of time. The number of database scans is controlled by the minimum support. Smaller the value of it may give more frequent itemsets and may lead to more number of database scans.

In the second phase, association rules are generated from the frequent itemsets derived by the first phase. At the worst case the total number of frequent itemsets may be 2^n-1, where n is the number of attributes in the database. From every frequent itemsets having more than one attribute, candidate rules are generated. Among these candidate rules, those are declared as derived rules that can satisfy another user given parameter, minimum confidence. This phase may also waste a significant amount of time depending on the number of frequent itemsets and the size of maximal frequent item set.

A significant number of works have been found to be carried out to derive the frequent itemsets with less number of database scans to save a considerable amount of time. DIC [6], FP-tree Growth [7], Border algorithm [8], SETM[9], SETM*-MaxK [10] AIS[3], pincer search [11], Set based Apriori [12], Quantitative approach[13], etc. are some of the algorithms that derive the frequent itemsets from a database with lesser number of database scans than apriori. However, only a little effort has been found to derive the rules. This work presents a technique to derive all possible rules in a better way from the frequent item sets given by algorithms of the first phase.

The rest of the paper is organized as follows: Section 2 describes some of the popular algorithms for feature selection; section 3 presents the proposed algorithm. Finally, Section 4 gives some experimental results to establish that the proposed algorithm is good enough to generate the rules.

2. SOME EXISTING ALGORITHMS

In this section, some of the rule generation algorithms are reproduced with brief description. *Table 1* contains some frequent itemsets that are used to explain the working of the algorithms.

Table 1
Example of Frequent Itemsets

Itemset size	*Itemset (support count)*
1	1(2), 2 (6), 3 (6), 4 (4), 5 (8), 6 (5), 7 (7), 8 (4), 9 (2)
2	5 6 (3), 5 7 (5), 6 7 (3)
3	5 6 7 (1)

2.1. Agrawal's Algorithm

This algorithm of rule generation was proposed by Agrawal *et al*. [14]. From every frequent itemset of $k \geq 2$, two subsets, A and C, are constructed in such a way that one subset, C, contains exactly one item in it and remaining k–1 items will go to the other subset, A. By the downward closure property of the frequent itemsets these two subsets are also frequent and their support is already calculated. Now these two subsets may

generate a rule $A \rightarrow C$, if the confidence of the rule is greater than or equal to the specified minimum confidence. Confidence or predictive accuracy or relative support of a rule is defined as (SUP($A \bigcup C$) / SUP(A))

```
Forall l_k k ≥ 2 do
        Forall i < = k do
                c = l_k[i]
                a = l_k – c
        if ((SUP(l_k)/SUP(a)) ≥ minconfidence
                declare a → c is a rule
        enddo
enddo
```

Figure 1: Agrawal's Algorithm

Limitation of this rule generation method is that it is capable of generating rules with only one attribute in the consequent part. In that way from a frequent itemset of size k only k number of candidate rules will be checked. Hence we may miss a number of possible rules.

With a minimum confidence of 20% this algorithm will generate the following rules from the frequent itemsets given in table 2.

Table 2
Rules derived based on Table 1

Rule	*Support*	*Confidence*
5 → 6	3	0.375000
5 →7	5	0.625000
6 →7	3	0.600000
5, 6 →7	1	0.333333

2.2. Srikant's First Algorithm [15]

This algorithm is the generalization of the first algorithm. Here, the rules that are checked are not limited only to the single consequent. Multiple items may also come in the consequent part. Practically any non empty proper subset of the considered frequent itemset will come to consequent part of the candidate rule. This technique will check 2^k–2 number of candidate rules for a single frequent itemset of size k. Steps of this algorithm are given in *figure 2*.

```
Forall frequent item set l_k, k≥2 do
    Call genrules(l_k, l_k)
Procedure genrules(l_k : frequent k-itemset; a_m : frequent m-itemset)
1)  A={(m-1)-itemset a_m-1 | a_m-1⊂a_m}
2)  For all a_m-1 ∈ A do begin
3)      conf=support(l_k)/support(a_m-1)
4)      if(conf≥minconf) then begin
5)          output the rule a_m-1 → (l_k - a_m-1 )with confidence=conf and
    support= support(l_k)
6)      if(m-1 >1 ) then
7)          call genrules(l_k, a_m-1)
8)      end
9)  end
```

Figure 2: Srikant's First Algorithm

This algorithm is capable of generating the all possible rules from the given frequent itemsets subject to the user given minimum confidence. But this algorithm has to store the powerset of a frequent itemset excluding trivial two of them at some point of time. If $k = 30$, then the number of subsets will be 1073741822. To execute the algorithm these subsets will occupy some additional memory, although the frequent itemsets are already maintained in the memory. At the same time some candidate rules are checked unnecessarily wasting a significant amount of time.

With a minimum confidence of 20% this algorithm will generate the rules reported in *table 3* from the frequent itemsets given in table 1.

Table 3
Rules Derived based on Table 1

Rule	*Support*	*Confidence*
6 →5	3	0.600000
5 →6	3	0.375000
7 →5	5	0.714286
5 →7	5	0.625000
7 →6	3	0.428571
6 →7	3	0.600000
6, 7 →5	1	0.333333
5 ,7 →6	1	0.200000
5, 6 →7	1	0.333333
6 →5, 7	1	0.200000

2.3. Srikant's Second Algorithm

To save time of the rule generation algorithm Srikant [15] has proposed another faster algorithm. This algorithm eliminates the checking of many unnecessary candidate rules. If for an itemset l, x is a subset of it, and $x \rightarrow (l - x)$ could not meet the minimum confidence, then a subset y of x cannot meet the minimum confidence. For example if $ABC \rightarrow D$ does not hold, then it is needless to check whether $AB \rightarrow CD$ holds or not. In this way a lot of unnecessary check can be avoided resulting in a faster execution of the algorithm. The steps of the algorithm are given Fig. 3.

1. forall frequent k-itemsets l_k, $k \geq 2$ do begin
2. H_1={consequent of the rule derived from l_k with one item in the consequent}
3. call ap_genrules(l_k, H_1)
4. end

procedure ap_genrules(l_k: frequent k-itemset; H_m: set of m-item consequent)

1. if ($k > m+1$) then begin
2. H_{m+1}=result of calling candidate apriori generation function with H_m
3. Forall $h_{m+1} \in H_{m+1}$ do begin
4. conf=support(l_k)/support(l_k-h_{m+1})
5. if(conf $\geq$ *minconf*) then begin
6. output the rule (l_k-h_{m+1})$\rightarrow h_{m+1}$ with conf and support(l_k)
7. else
8. delete h_{m+1} from H_{m+1};
9. end
10. call ap_genrules(l_k, H_{m+1})
11. end

Figure 3: Srikant's Second Algorithm

The algorithm works faster because it avoids a number of unnecessary checking. But the problem of memory requirement still exists, although the requirement is reduced a bit, due to the removal of some subsets before the generation of the consequent part. But if the support count of the itemsets do not vary to much and the minimum confidence is given a too low value, the total number of subsets generated may again equal to earlier one.

The rules generated with a minimum confidence 20% by this algorithm based on the frequent itemsets given on Table 1 are tabulated in Table 4.

Table 4
Rules Derived based on Table 1

Rule	*Support*	*Confidence*
6 →5	3	0.600000
5 →6	3	0.375000
7 →5	5	0.714286
5 →7	5	0.625000
7 →6	3	0.428571
6 →7	3	0.600000
6, 7 →5	1	0.333333
6 →5, 7	1	0.200000
5, 7 →6	1	0.200000
5, 6 →7	1	0.333333

2.4. Discussion

From the descriptions of the above three algorithms it can be observed that the Agrawal's algorithm is very simple. But we may miss a number of rules that could have been discovered from the given frequent itemsts. But the other two algorithms can discover all the rules subject to the given threshold, minimum confidence. But Srikant's first algorithm takes longer time to generate the same rules from the same set of frequent itemsets. So, Srikant's second algorithm is considered to be the best among these three algorithms. Hence, the proposed algorithm is compared only with this algorithm

Although, Srikant's Second algorithm is found to be the best rule generation algorithm till now it suffers from the following difficulties. The candidate consequent that has to be generated for the rule discovery will need a significant amount of memory. And a considerable time is wasted by generating the same consequent several times for different antecedent. For example if A is a subset of B, while generating the rules using A, all the candidate consequent will be generated. The same operation will be repeated for the set B also, although many of them have been already generated in the earlier stage since B is a super set of A.

3. THE PROPOSED ALGORITHM

The proposed algorithm is meant for generating the association rules from a given set of frequent itemsets in an efficient manner both in time and space (memory) requirement point of view. This algorithm is capable of generating all possible rules subject to the user specified minimum confidence. During the rule generation process, it avoids the unnecessary checking of some candidate rules which results in significant reduction of the time required to generate the rules. All the rules that can be found out by the third

algorithm discussed above will be generated by the proposed algorithm. It will use the frequent itmesets that are already stored to the memory and will not generate the subsets of a given frequent itemset. Hence, the memory requirement for this algorithm is far less than the other one. Step 2 and step 10 of fig 3, are used to generate the subsets in Srikant's algorithm. The steps of the proposed algorithm are listed in Fig. 4.

Input: L={l_k |l_k is set of frequent k–itemsets with sorted on support count in descending order, 1≤k≤maxsize}; minconf- the minimum confidence specified by the user

Output: the strong association rules discovered with their support and confidence.

```
1. forall l_k, l_k ∈L 1≤k≤max_size-1 do begin
2    reqsup=support(l_k)*minconf
3    found=0
4.   forall l_m, l_m ∈L k+1≤m≤max_size do begin
5.       if(support(l_m)≥resup) then begin
6            if(l_k⊂ l_m) then begin
7.               found=found+1
8.               conf=support(l_m)/support(l_k)
9.               generate the rule l_k→( l_m- l_k) &=conf and support=support(l_m)
10.          end if
11.      else
12.          if(found<2)
13.              continue step 1 with next k
14.          else
15.              found=0;
16.          endif
17.      endif
18.  end do
19. end do
```

Figure 4: The Proposed Algorithm

This algorithm is capable of discovering all possible rules from the given set of frequent itemsets subject to a user specified minimum confidence. It discovers all rules with a fixed antecedent and with different consequents. For that it checks only those

frequent itemsets which can fulfill the minimum confidence. At the same time, the algorithm will go to the next level of frequent itemset with the same antecedent if in the current level at least two itemset fulfills the minimum confidence. This eliminates a number of unnecessary checks for the rule.

Lemma: If there exists two rules $A \rightarrow C$ and $A \rightarrow \{C \bigcup X\}$, where $X \notin A \bigcup C$, then the confidence of the second cannot be larger than first one.

Proof: Let D: the Dataset

R: a set of items

X: an item such that $X \in D$ but $X \notin R$

A: a set of items, such that $A \in D$ and $A \subset R$,

Support(R) = m

Support(X) = n

Support ($R \bigcup X$) = p

Support (A) = q

But, $p \leq \min(m, n)$

If $n < m$ then $p < m$

If $n \geq m$ then $p = m$

Hence, $p <= m$

$\Rightarrow$ $p/q <= m/q$

$\Rightarrow$ confidence of $A \rightarrow \{R - A\} \leq$ confidence of $A \rightarrow \{(R \bigcup X) - A\}$

The proposed algorithm is avoiding the unnecessary checking for the rules based on the above lemma. But still the algorithm is capable of generating all possible rules subject to the minimum confidence. The algorithm is generating the rules with a fixed antecedent first. When all the rules with that antecedent are generated then it will go to the next antecedent. For a fixed antecedent it checks for the rules with equal size consequent, then go to the next level. For, a given antecedent if all rules in the level k, where k is the number of items in the consequent, have confidence less than the threshold, i.e. no rules are generated, then the confidence of any rule in k+1 level also cannot be more than threshold. So checking for rules from this level onward can be avoided without missing any rules. If in level k, one rule fulfilled the threshold, then also in k+1 level there will be no rule. Because in the k+1 level itemsets are generated form two of the k level itemsets. Now the maximum possible confidence of the rule in the k+1 level will be minimum confidence of the two itemsets from which this is constructed. Since the

confidence of only one of them is larger than the threshold, others must be less then the threshold. So the confidence of the rule in $k + 1$ will be less than the threshold. So, it is needless to check for the rules in the next level without missing any valid rule. So, it can be concluded that the proposed algorithm is complete.

In this rule generation process a significant amount of time may be wasted in the step 6 of the proposed algorithm by checking whether a set is a subset of another or not, step 6 of the proposed algorithm. If the itemsets are represented in array, it will take n comparisons at the worst case where n is the size of the larger set. To save this time we used a different representation of the itemsets. By this representation we can check for the subset within constant time, involving only two basic integer operation. In this representation every itemset is represented as an unsigned integer, where a bit corresponding to an item is set to 1. For example the itemset {1,3,4} is represented as 26. If an itemset is to be checked whether it is a subset of another or not, it can be done by one bitwise AND operation followed by a comparison. For example if we have to check {1,3} is a subset of {1,3,4} or not, we have to evaluate 10&26, where 10 and 26 represents the two sets, then the result has to be checked for equality with the representation of the first itemset. In this case the result is 10 and the first itemset is also represented as 10, hence {1,3} is a subset of {1,3,4}. In the similar way we can find the difference of two sets that will be needed during the production of the rule.

The algorithm has been tested with several test datasets. And it produced all the rules that were discovered by the above mentioned algorithm in quick time.

Discussion

This algorithm uses the above discussed concept during the rule generation to eliminate the unnecessary checking for the rules. Since a number of unnecessary checking are avoided, it produces the rules in a faster way.

4. EXPERIMENTAL RESULTS

For the implementation of different algorithms a machine with Intel Xeon processor of 3.0 GHz speed and 1 GB memory was used. The proposed algorithm was tested with several test datasets, analysis on some of them is given below. Initially, using the Apriori algorithm frequent itemsets were discovered from the dataset. For that a synthetic dataset having 10000 records and 20 attribute was considered. Summary of those frequent itemsets with a minimum support of 20% are given in the table 5.

These frequent itemsets were used discover the association rules. The proposed algorithm and Srikant's second algorithm was applied separately on these frequent itemsets. Along with the discovered rules, time taken to discover the rules also produced as the output. Table 6 contains the comparison of the two algorithms on this synthetic dataset.

Table 5
Summary of Frequent Itemsets from a Synthetic Dataset

Itemset size	*Number of itemsets*
1	18
2	133
3	216
4	43

Table 6
Comparison on Synthetic Dataset

Min Conf.	*Our Method*		*Srikanth's Method*	
	Rules	*Time(ns)*	*Rules*	*Time(ns)*
20%	2250	89,666,000	2250	1,263,772,000
30%	1822	25,691,000	1822	97,006,000
40%	1379	21,022,000	1379	47,039,000
50%	951	17,669,000	951	36,186,000
60%	580	12,605,000	580	25,487,000

Similar comparison was done for the Monks-1 and Monks-3 datasets also. These datasets are available in UCI machine learning data repository [16]. Both the dataset contains 8 attributes, first and last one being the class label and instance id respectively. All other six attributes are numeric in nature. The datasets were converted to market basket form for every unique value of the attributes. 17 attributes were resulted in the market basket dataset. Monks-1 dataset contains 124 records and Monks-3 contains 122. From these datasets frequent itemsets were discovered by using Apriori algorithm, with a support count of 10% and used them to discover the rules. Table 7 and Table 9 contains the summary of frequent itemsets of Monks-1 and Monks-3 datasets respectively. Table 8 and Table 10 contain the comparison of the two algorithms on these two datasets respectively.

Table 7
Summary of Frequent Itemsets from Monks-1 Datasets

Itemset size	*Number of itemsets*
1	*17*
2	*94*
3	*20*

Table 8
Comparison on Monks-1 Dataset

Min Conf.	*Our Method*		*Srikanth's Method*	
	Rules	*Time(ns)*	*Rules*	*Time(ns)*
20%	252	7,989,000	252	8,474,000
30%	180	7,307,000	180	7,616,000
40%	99	6,453,000	99	6,674,000
50%	55	6,016,000	55	6,242,000
60%	13	5,498,000	13	5,830,000

Table 9
Summary of Frequent Itemsets from Monks-3 Datasets

Itemset size	*Number of itemsets*
1	17
2	93
3	19

Table 10
Comparison on Monks-3 Dataset

Min conf.	*Our Method*		*Srikanth's Method*	
	Rules	*Time(ns)*	*Rules*	*Time (ns)*
20%	241	7,912,000	241	8,285,000
30%	172	7,280,000	172	7,433,000
40%	90	6,391,000	90	6,569,000
50%	52	6,025,000	52	6,251,000
60%	11	5,506,000	11	5,734,000

Based on our experimental study, following observations were made:

- Same numbers of rules are generated by both the algorithms even if the orders of generation are different. It is evident from the columns 2 and 4 of Table 6, 8 and 10.
- The time taken by the proposed algorithm is significantly less than the other. It can be seen from the columns 3 and 5 of Table 6, 8 and 10.
- The time saved by the proposed algorithm is more significant when the maximal frequent sets size is larger and total number of frequent itemset is larger.

5. CONCLUSIONS

This paper has presented a fast, memory efficient algorithm for generating rules from a given set of frequent itemsets. The algorithm has been established to be superior to its other counterparts in light of several real life dataset. Based on experimentation, it has been found that the proposed algorithm is good enough for generating the rules from the market basket dataset.

NOTES

[1] This work has been carried out under a project Funded by AICTE

REFERENCES

[1] Pujari A. K., Data Mining Techniques, Universities Press, 1st edition 2001.

[2] Han J., Kamber M., Data Mining Concepts and Techniques, Morgan Kaufmann, First Indian Reprint, (2002).

[3] Agrawal R, Imielinski T, Swami A. Database Mining: A Performance Perspective, *IEEE Transactions on Knowledge Discovery and Data Engineering*, 5(6), (1993) 914–925.

[4] Agarwal R. and Shrikant R., Fast Algorithms for Mining Association Rules, in *20th VLDB Conf*, (Sept. 1994.)

[5] Agarwal R., Mannila H., Shrikant R., Toivonan H. and Verkamo A .I, Fast Discovery of Association Rules, in U. Fayyad and *et al*, editors *Advances in Knowledge Discovery and Data Mining*, MIT Press, (1996).

[6] S. Brin, R. Motwani, J. D. Ullman, S. Tsur, Dynamic Itemset Counting and Implication Rules for Market Basket Data, *SIGMOD Record*, New York, **6**, (2), (June 1997), 255–264.

[7] Han, J., Pei, J., and Yin, Y. Mining Frequent Patterns without Candidate Generation. In *Proc. 2000 ACM SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, Dallas, TX, (2000), 1–12.

[8] Mannila, H., Toivonen, H., and Verkamo, A. Efficient Algorithms for Discovering Association Rules. In *Proceedings of KDD94: AAAI Workshop on Knowledge Discovery in Databases* (1994), 181–192.

[9] Houstma M., Swami A., Set-oriented Mining for Association Rules in Relational Databases, *11th IEEE int'l Conf. on Data Engineering*, (1995), 25–33.

[10] Chang Y-I, Hsieh Y-M, SETM*-MaxK: A Set-based Approach to Find Largest Itemset, PAKDD, Springer Verlag (editors Chen M.S, Yu P.S., Liu B) 2002, LNAI 2336, 316–321,

[11] Lin D-I, Kedem Z. M., Pincer Search: A New Algorithm for Discovering Maximum Frequent Set, in *6th International Conference on Extending Database Technology*, (March 1998).

[12] Shoemaker C. A., Ruiz C., Association Rule Mining Algorithms for Set Valued Data, *IDEAL,* editors Liu J. *et al*. LNCS 2690, (2003), 669–676,

[13] Srikant R., Agrawal A., Mining Quantitative Association Rules in Large Relational Tables, in *Proc. of ACM SIGMOD Conf. on Management of Data*, (1996)

[14] Agrawal R, Imielinski T, Swami A. Mining Association Rules between Sets of Items in Large Databases, in *Proc. of ACM SIGMOD Conf. on Management of Data*, (1993).

[15] Srikant R, Fast Algorithms for Mining Association Rules and Sequential Patterns, *Phd Thesis,* University of Wisconsin-Madison, (1996).

[16] *www.ics.uci.edu/~mlearn/MLRepository.html* UCI Machine Learning Repository

B. Nath & D. K. Bhattacharyya
Dept of Computer Science and Engineering
Tezpur University
Assam, INDIA
E-mail: bnath@tezu.ernet.in; dkb@tezu.ernet.in

A Ghosh
Machine Intelligence Unit and Center for Soft Computing Research
Indian Statistical Institute
West Bengal, INDIA
E-mail: ash@isical.ac.in