An Algorithm for Frequent Pattern Mining Based On Apriori

Goswami D.N.*, Chaturvedi Anshu. **
Raghuvanshi C.S.***
*SOS In Computer Science Jiwaji University Gwalior
** Computer Application Department MITS Gwalior
** *GICTS College of Professional Education Gwalior

Abstract:

Frequent pattern mining is a heavily researched area in the field of data mining with wide range of applications. Mining frequent patterns from large scale databases has emerged as an important problem in data mining and knowledge discovery community. A number of algorithms has been proposed to determine frequent pattern. Apriori algorithm is the first algorithm proposed in this field. With the time a number of changes proposed in Apriori to enhance the performance in term of time and number of database passes. In this paper three different frequent pattern mining approaches (Record filter, Intersection and Proposed Algorithm) are given based on classical Apriori algorithm. In these approaches Record filter approach proved better than classical Apriori Algorithm, Intersection approach proved better than Record filter approach and finally proposed algorithm proved that it is much better than other frequent pattern mining algorithm. In last we perform a comparative study of all approaches on dataset of 2000 transaction.

Keywords:

Data Mining, Frequent Patterns, Knowledge Discovery, Pattern mining.

1. Introduction:

This section introduces the basic concept of frequent pattern mining for the discovery of interesting associations and correlations between item sets in transactional and relational database. Frequent pattern are patterns that appear in a dataset frequently. For example, a set of items, such as milk and bread that appear frequently together in a transaction data set is a frequent item set. Frequent patterns are prevalent in real-life data, such as sets of items bought together in a superstore. Frequent pattern mining has been successfully applied to association rule mining, pattern-based classification, clustering, finding correlated items, and has become an essential data mining task.

Frequent item sets play an essential role in many data mining tasks that try to find interesting patterns from databases. The original motivation for searching frequent pattern came from the need to analyze so called supermarket transaction data, that is, to examine customer behavior in terms of the purchased products. Frequent Pattern describe how often items are purchased together. Since their introduction in 1993 by Argawal et al. [1], the frequent item set and association rule mining problems have received a great deal of attention. Within the past decade, hundreds of research papers have been published. Presenting new algorithms or improvements on existing algorithms to solve these mining problems more efficiently. In this chapter, we explain the basic frequent item set mining problems.

2. Problem

The problem is usually decomposed into two sub problems.

1. One is to find those item sets whose occurrences exceed a predefined threshold in the database; those item sets are called frequent or large item sets.
2. The second problem is to generate association rules from those large item sets with the constraints of minimal confidence.

Let $I = \{ i_1, i_2, i_3, i_4 \ldots\ldots\ldots i_m \}$ be a set of m distinct literals called items, D is a set of transactions (variable length) over I. Each transaction contains a set of items $i_1, i_2, i_3, i_4 \ldots\ldots\ldots i_k \subseteq I$. Each transaction is associated with an identifier, called TID. An association rule is an implication of the form $X \Rightarrow Y$, where $X, Y \subset I$ and $X \cap Y = 0$. Here X is called the antecedent and Y is called the

consequent of the rule. The rule $X \Rightarrow Y$ holds in the transaction set D with confidence α if among those transactions that contain X α% of them also contain Y. The rule $X \Rightarrow Y$ has support S in the transaction set D if S% of transactions in D contains $X \cup Y$. The selection of association rules is based on these two values (some additional constraints may also apply). These are two important measures of rule interestingness. They respectively reflect usefulness and certainty of a discovered rule. They can be described by the following equations:

Support ($X \Rightarrow Y$) = Frequency ($X \cup Y$) / /D/
Confidence ($X \Rightarrow Y$) = Frequency ($X \cup Y$) / Frequency (X) where /D/ represents the total number of transactions (tuples) in D.

Suppose one of the large item sets is Lk, Lk = {i1, i2, … , ik}, association rules with this item sets are generated in the following way:

The first rule is {i1, i2, … ,ik-1}$\Rightarrow$ {ik}, by checking the confidence. This rule can be determined as interesting or not. Then other rule are generated by deleting the last items in the antecedent and inserting it to the consequent, further the confidences of the new rules are checked to determine the interestingness of them. These processes iterated until the antecedent becomes empty. Since the second sub problem is quite straight forward, most of the researches focus on the first sub problem.

The first sub-problem can be further divided into two sub-problems: candidate large item sets generation process and frequent item sets generation process. We call those item sets whose support exceeds the support threshold as large or frequent item set.

3. Apriori Algorithm for Frequent Pattern Mining

Apriori is a algorithm proposed by R. Agrawal and R Srikant in 1994 [1] for mining frequent item sets for Boolean association rule. The name of algorithm is based on the fact that the algorithm uses prior knowledge of frequent item set properties, as we shall see following. Apriori employs an iterative approach known as level-wise search, where k item set are used to explore (k+1) item sets. There are two steps in each iteration.
The first step generates a set of candidate item sets. Then, in the second step we count the occurrence of each candidate set in database and prunes all disqualified candidates (i.e. all infrequent item sets). Apriori uses two pruning technique, first on the bases of support count (should be greater than user specified support threshold) and second for an item set to be frequent , all its subset should be in last frequent item set The iterations begin with size 2 item sets and the size is incremented after each iteration. The algorithm is based on the closure property of frequent item sets: if a set of items is frequent, then all its proper subsets are also frequent.

3.1 Apriori Algorithm

Initialize: k := 1, C_1 = all the 1- item sets;
read the database to count the support of C_1 to determine L_1.
L_1 := {frequent 1- item sets};
k:=2; //k represents the pass number//
while ($L_{k-1} \neq \varnothing$) do
begin
C_k := gen_candidate_itemsets with the given L_{k-1}
prune(C_k)
for all transactions $t \in T$ do
increment the count of all candidates in C_K that are contained in t;
L_k := All candidates in C_k with minimum support ;
k := k + 1;
end
Answer := $\cup_k L_k$;

The first weakness of this algorithm is the generation of a large number of candidate item sets. The second problem is the number of database passes which is equal to the max length of frequent item set.

4. Record Filter Approach based on Apriori Algorithm for Frequent Pattern Mining

Although we have made some changes in Apriori algorithm for frequent pattern mining and name this Record Filter approach and it is efficient as compare to the Apriori algorithm. We have suggest some changes which improve the efficiency of apriori, memory management and remove the complexity of process. Here we are presenting a different approach in Apriori algorithm to count the support of candidate item set. In the classical apriori algorithm, we check the occurrence of candidate item in each transaction of any length. In this when we count the support of candidate set of length k, we also check its occurrence in transaction whose length may be greater than , less than or equal to the k. But in the new approach we count the support of candidate set only in the transaction record whose length is greater than or equal to the length of candidate set, because candidate set of length k , can not exist in the transaction record of length k-1 , it may exist only in the transaction of length greater than or equal to k.

This approach has taken very less time as compared to classical Apriori.

4.1 Record filter approach based on Apriori Algorithm

Initialize: k := 1, C_1 = all the 1- item sets;
read the database to count the support of C_1 to determine L_1.
L_1 := {frequent 1- item sets};
k:=2; //k represents the pass number//
while ($L_{k-1} \neq \varnothing$) do
begin
C_k := gen_candidate_itemsets with the given L_{k-1}
prune(C_k)
for all transactions t whose length is greater than or equal to k $\in$ T do
increment the count of all candidates in C_K that are contained in t;
L_k := All candidates in C_k with minimum support ;
k := k + 1;
end
Answer := $\cup_k L_k$;

In this new approach we require very less time in comparison to classical apriori algorithm.

5. Intersection Approach Based On Apriori Algorithm for Frequent Pattern Mining

In the previous section we have describe the Record filter approach based on Apriori, now we are suggesting one another changes in Apriori which gives the better result as compare to the Record Filter approach. The Intersection Algorithm is designed to improve the efficiency, memory management and remove the complexity of Apriori. Here we are presenting a different approach in Apriori algorithm to count the support of candidate item set. Basically this approach is more appropriate for vertical data layout, since Apriori basically works on horizontal data layout. In this new approach, we use the set theory concept of intersection. In Classical Apriori algorithm, to count the support of candidate set each record is scanned one by one and check the existence of each candidate, if candidate exists then we increase the support by one. This process takes a lot of time, requires iterative scan of whole database for each candidate set, which is equal to the max length of candidate item set. In modified approach, to calculate the support we count the common transaction that contains in each element's of candidate set, by using the intersect query of SQL. This approach requires very less time as compared to classical Apriori.

5.1 Intersection Algorithm Based On Apriori

Initialize: K: = 1, C_1 = all the 1- item sets;
read the database to count the support of C_1 to determine L_1.
L_1 := {frequent 1- item sets};
k:=2; //k represents the pass number//
while ($L_{k-1} \neq \varnothing$) do
begin
C_k: = gen_candidate_itemsets with the given L_{k-1}
Prune (C_k)
for all candidates in C_k do
count the number of transactions that are common in each item $\in C_k$
L_k := All candidates in C_k with minimum support ;
k := k + 1;
end
Answer := $\cup_k L_k$;
This modified (Intersect Method) approach, takes only one data base pass to change the horizontal data layout to vertical datalayout..

6. Proposed Algorithm Based On Apriori for Frequent Pattern Mining

In this new approach we have determined changes that are going to serve the best in the field of frequent pattern mining. In this new approach, we are presenting an algorithm that uses the concept of both algorithm i.e. Record filter approach and Intersection approach in Apriori algorithm .To count the support of candidate item set ,we have considered both above mentioned approach. In this new approach, we use the set theory concept of intersection with the record filter approach.. In proposed algorithm, to calculate the support, we count the common transaction that contains in each element's of candidate set, with the help of the intersect query of SQL. In this approach, we have applied a constraints that we will consider only those transaction that contain at least k items, not less than k in process of support counting for candidate set of k length. This approach requires very less time as compared to all other approaches.

6.1 Proposed Algorithm Based On Apriori

Initialize: K: = 1, C_1 = all the 1- item sets;
read the database to count the support of C_1 to determine L_1.
L_1 := {frequent 1- item sets};
k:=2; //k represents the pass number//
while ($L_{k-1} \neq \varnothing$) do
begin
C_k: = gen_candidate_itemsets with the given L_{k-1}
Prune (C_k)

for all candidates in C_k do
count the number of transactions of atleast k length that are common in each item $\in C_k$
L_k := All candidates in C_k with minimum support ;
k := k + 1;
end
Answer := $\cup_k L_k$;

In this modified (Intersect Method) approach we require only one data base pass to change the horizontal

7. Time Comparison in Apriori, Record Filter, Intersection and Proposed Algorithm

7.1 Time Comparison between Apriori and Record Filter

It is clear from the figure1 and Table1 that Record filter approach is efficient than the classical apriori algorithm. In the Analysis process when we have taken the 400 record apriori takes 6 second while Record filter takes the 4 second when we increase the size of record we can see in the table the results are very efficient related to time.

Records	Apriori (Time in Seconds)	Record filter (Time in Seconds)
400	6	4
800	20	17
1200	38	33
1600	70	64
2000	143	132

Table –1

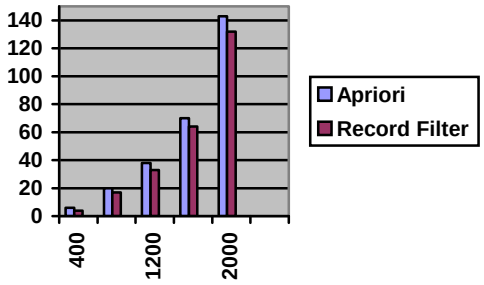

Figure-1

7.2 Time Comparison between Apriori and Intersection

It is clear from the figure2 and Table2 that Intersection approach is efficient than the apriori algorithm. In the Analysis process when we have taken the 400 record apriori takes 6 second while Intersection takes the 4 second when we increase the size of record we can see in the table the results are very efficient related to time.

Records (Time in Seconds)	Apriori (Time in Seconds)	Proposed (Time in Seconds)
400	6	3
800	20	9
1200	38	11
1600	70	18
2000	143	29

Table-2

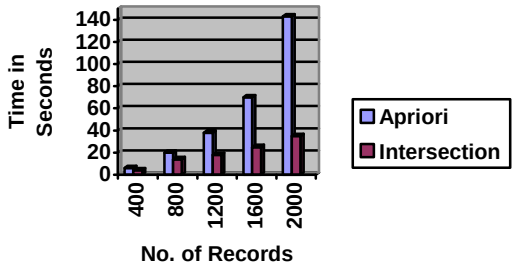

Figure-2

7.3 Time Comparison between Apriori and Proposed Algorithm

It is clear from the figure3 and Table3 the Proposed Algorithm is efficient than the apriori algorithm. In the Analysis process when we have taken the 400 record apriori takes 6 second while Proposed Algorithm takes the 3 second when we increase the size of record we can see in the table the results are very efficient related to time. In the conclusion we can say that Proposed Algorithm is better than the other three algorithms.

No. of Records	Apriori	Intersection
400	6	4
800	20	14
1200	38	18
1600	70	25
2000	143	35

Table-3

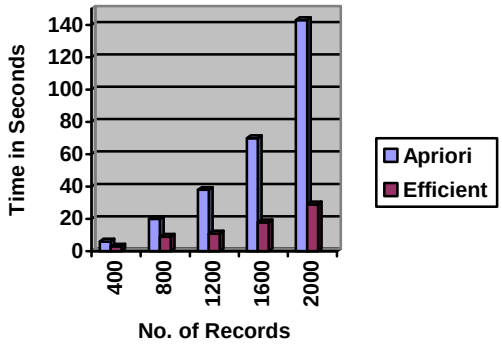

Figure-3

7.4 Time Comparison between Classical Apriori, Record Filter, Intersection and Proposed algorithm

For the comparative study of Classical Apriori, Record Filter, Intersection and Proposed Algorithm, we have taken a database of 2000 transaction of 50 items. In this analytical process we considered 500 transactions to generate the frequent pattern with the support count 10% .We have repeated the same process by increasing the transaction. Here we first compare the result of all approaches with Classical Apriori Algorithm because all approaches are based on Classical Apriori. After that we compare all approaches to find out the best. After the experiment on all approaches, we have designed a graph and summarized a result in the following table

Records (Time in Seconds)	Apriori (Time in Seconds)	Record Filter (Time in Seconds)	Intersection (Time in Seconds)	Proposed (Time in Seconds)
400	6	4	4	3
800	20	17	14	9
1200	38	33	18	11
1600	70	64	25	18
2000	143	132	35	29

Table4

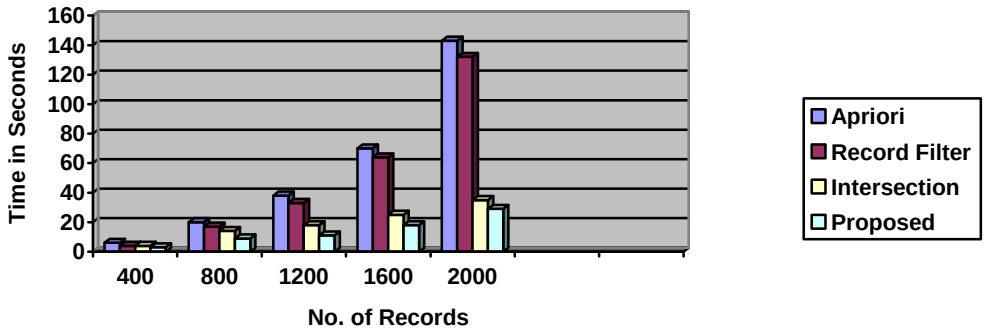

Figure-4

Conclusion:

Association rule mining has a wide range of applicability such as market basket analysis, medical diagnosis/ research, website navigation analysis, homeland security and so on. In this paper, we surveyed the list of existing association rule mining techniques and compare these algorithms with our modified approach. The conventional algorithm of association rules discovery proceeds in two and more steps but in our approach discovery of all frequent item will take the same steps but it will take the less time as compare to the conventional algorithm. We can conclude that in this new approach, we have the key ideas of reducing time. As we have proved above how the proposed Apriori algorithm take less time than that of classical apriori algorithms. That is really going to be fruitful in saving the time in case of large database. This key idea is surely going to open a new gateway for the upcoming researcher to work in the filed of the data mining.

References

[1]Agrawal, R., Imielinski, T., and Swami, A. N. 1993. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 207-216.

[2] Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules. In Proc. 20th Int. Conf. Very Large Data Bases, 487-499.

[3] Agarwal, R. Agarwal, C. and Prasad V., A tree projection algorithm for generation of frequent itemsets. In J. Parallel and Distributed Computing, 2000.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. IBM Research Report RJ9839, IBM Almaden Research Center, San Jose, California, June 1994.

[5] R.J. Bayardo, Jr. Efficiently mining long patterns from databases. In L.M. Haas and A. Tiwary, editors, Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, volume 27(2) of SIGMOD Record, pages 85–93. ACM Press, 1998.

[6] S. Parthasarathy, M. J. Zaki, M. Ogihara, S. Dwarkadas; Incremental and interactive sequence mining; Int'l Conf. on Information and Knowledge Management; 1999.

[7] Helen Pinto, Jiawei Han, Jian Pei, Ke Wang, Qiming Chen, Umeshwar Dayal; Multi-Dimensional Sequential Pattern Mining; Int'l Conf. on Information and Knowledge Management; 2001.

[8] Assaf Schuster, Ran Wolff, and Dan Trock; Distributed Algorithm for Mining Association Rules; IEEE Int'l Conf. on Data Mining; November 2003.

[9] Wei-Guang Teng, Ming-Syan Chen, and Philip S. Yu; Resource-Aware Mining with Variable Granularities in Data Streams; SIAM Int'l Conf. on Data Mining; 2004.

[10] Adriano Veloso, Wagner Meira Jr., Marcio Carvalho, Srini Parthasarathy, Mohammed J. Zaki; Parallel, Incremental and Interactive Mining for Frequent Itemsets in Evolving Databases; Int'l Workshop on High Performance Data Mining: Pervasive and Data Stream Mining; May 2003.

[11] Adriano Veloso, Matthew Eric Otey, Srinivasan Parthasarathy, Wagner Meira Jr.; Parallel and Distributed Frequent Itemset Mining on Dynamic Datasets; Int'l Conf. on High Performance Computing; 2003.

[12] Haixun Wang, Wei Fan, Philip S. Yu, Jiawei Han; Mining Concept-Drifting Data Streams using Ensemble Classifiers; ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining; August 2003.

[13] Li Yang, Mustafa Sanver; Mining Short Association Rules with One Database Scan; Int'l Conf. on Information and Knowledge Engineering; June 2004.

[14] Jeffrey Xu Yu, Zhihong Chong, Hongjun Lu, Aoying Zhou; False Positive or False Negative: Mining Frequent Itemsets from High Speed ATransactional Data Streams; Int'l Conf. on Very Large Databases; 2004.

[15] Qingguo Zheng, Ke Xu, Shilong Ma; When to Update the Sequential Patterns of Stream Data; Pacific-Asia Conf. on Knowledge Discovery and Data Mining; 2003.

[16] Yunyue Zhu, Dennis Shasha; StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time; Int'l Conf. on Very Large Data Bases; 2002.