Performance Analysis of Apriori Algorithm with Progressive Approach for Mining Data

Shilpa
Department of Computer Science & Engineering
Haryana College of Technology & Management, Kaithal, Haryana, India

Sunita Parashar
Associate Professor
Department of Computer Science & Engineering
Haryana College of Technology & Management, Kaithal, Haryana, India

ABSTRACT
The Data Mining refers to extract or mine knowledge from huge volume of data. Association Rule mining is the technique for knowledge discovery. It is a well-known method for discovering correlations between variables in large databases. One of the most famous association rule learning algorithm is Apriori. The Apriori algorithm is based upon candidate set generation and test method. The problem that always appears during mining frequent relations is its exponential complexity. In this paper, we propose a new algorithm named progressive APRIORI (PAPRIORI) that will work rapidly`. This algorithm generates frequent itemsets by means of reading a particular set of transactions at a time while the size of original database is known.

Keywords
Data mining, Minimum Support, Number of transactions (K), Execution time.

1. INTRODUCTION
Data mining is the extraction of hidden descriptive or predictive information from large databases. It has been described as "the task of discovering interesting patterns from large amount of data where the data can be stored in databases, data warehouses or other information repositories"[1]. Thus goal of data mining is to turn "data into knowledge" [2]. It can be known by different names like knowledge discovery (mining) in databases (KDD), knowledge extraction, data/pattern analysis, information harvesting, business intelligence and others. Data mining examine the sample data of a problem and determine a model (predictive or descriptive) that fits close to solve the problem. Different data mining techniques available to solve data mining problems are classification, association rule mining, time series analysis, clustering, summarization, sequence discovery. Out of these Association rule mining is popular and well researched data mining technique for discovering interesting relations between variables in large databases. There are various association rule mining algorithms like Apriori [3][4], FP-Growth [5], Partition based algorithm [6], Incremental update (FUp based [7], probability based [8], promising based[9]), Haskell based approach[10], Fast algorithm[11] and other Apriori based algorithms etc**.** These algorithms try to find out correlation or association among data in large volume of database. Most of the previous studies for frequent itemsets generation adopt an Apriori algorithm that has exponential complexity (high execution time). In this study, we propose an algorithm that will reduce execution time by means of generating itemsets progressively from static database.

2. METHOD FOR MINING ASSOCIATION RULES
A new method for generating frequent itemsets is introduced in this section, which is based on the existing mining algorithms. This will generate itemsets progressively hence named as PAPRIORI algorithm.

Initially all 1-itemsets are set as Estimated Infrequent (EI) itemsets. Now read database with K transactions at a time. Calculate large itemsets until number of transactions read is less than total number of transactions in database with increase in transactions read with a parameter K. This is repeated until Estimated Frequent (EF) and Estimated Infrequent (EI) itemsets are present.

In order to obtain large itemsets among EF and EI, read K transactions at a time. If candidate belongs to transaction then increase value of counter for that candidate. For each candidate that belongs to EI if value of counter satisfies minimum support then set candidate as EF. If candidates belong to EF or CF (Confirmed Frequent) then their immediate superset is set as EI. For each candidate that belongs to EF if it is read throughout the whole database once move that into CF. On the other hand if candidate belongs to EI, if it is read throughout the whole database once move it into CI (Confirmed Infrequent).

EF=CI=CF=Φ;

EI={all 1-itemset};

Set dbrun=0,K,minsup;

do

```
{
        dbrun++;
        Set NumRead=0;
        while(NumRead<Totaltransactions)
        {
                getLarge_itemset(EI,EF);
                NumRead=NumRead+K;
        }
}while((EF!=0)||(EI!=0));
getLarge_itemset(EI,EF)
{
        for(int  i=0; i<K; i++)
        {
        for each itemset C Є(EI||EF)
        {
                If(C Є transaction)
                C.counter++;
        }
        }
        for each itemset C Є EI
        {
                If(C.counter ≥ minsup)
                        move C  from EI to EF;
                If((Sset ⊐ C) Є (EF||CF))
                        add  Sset   in EI;
        }
        for each itemset C Є EF
        {
                If((C.startNum= =NumRead)&&
(C.dbrun= =dbrun-1))
                {
                        move C  into CF;
                }
        }
        for each itemset C Є EI
        {
                If((C.startNum==NumRead) && (C.dbrun==dbrun-1))
                {
                        move C  into CI;
                }
        }
}
```

3. AN ILLUSTRATIVE EXAMPLE

Suppose total number of transactions in a database is 6 with minimum support equal to 15% having total number of items equal to 3 and the number of transactions read at a time (K) is 2, then proposed algorithm will work in following steps:-

Tran./ Item	A	B	C
T1	1	1	0
T2	0	1	0
T3	1	1	0
T4	1	1	1
T5	0	0	1
T6	0	1	1

Step 1) Initially A=B=C=0;

EI	EF	CF	CI
A		NULL	
B			
C			

Initially counter for A, B, C starts and they are set as EI. Only NULL is set as CF.

Step 2) After K transactions are read
A=1, B=2, C=0, AB=0;

Tran./ Item	A	B	C
T1	1	1	0
T2	0	1	0
T3	1	1	0
T4	1	1	1
T5	0	0	1
T6	0	1	1

EI
C
AB

EF
A
B

CF
NULL

CI

As counter of A, B satisfy min. support so move them from EI to EF and counter for AB is created and set AB as EI.

Step 3) After 2K transactions are read
A=3, B=4, C=1, AB=2, BC=0, AC=0;

Tran./ Item	A	B	C
T1	1	1	0
T2	0	1	0
T3	1	1	0
T4	1	1	1
T5	0	0	1
T6	0	1	1

EI
BC
AC

EF
A
B
C
AB

CF
NULL

CI

As C, AB satisfy min. support move them from EI to EF. Counter for BC and AC starts and they are set as EI because C is set as EF so its combination with other 1-itemset (either EF or CF) can be created.

Step 4) After 3K transactions are read
A=3, B=5, C=3, AB=2, BC=1, AC=0, ABC=0;

Tran./ Item	A	B	C
T1	1	1	0
T2	0	1	0
T3	1	1	0
T4	1	1	1
T5	0	0	1
T6	0	1	1

EI
AC
ABC

EF
AB
BC

CF
NULL
A
B
C

CI

BC satisfy min. support so move it from EI to EF. A,B,C are counted through all the transactions once so set them as CF. Start counter for ABC and set it as EI because AB, BC are EF so it's combination ABC is set as EI.

Step 5) After 4K transactions are read
A=3, B=5, C=3, AB=3, BC=1, AC=0, ABC=0;

Tran./ Item	A	B	C
T1	1	1	0
T2	0	1	0
T3	1	1	0
T4	1	1	1
T5	0	0	1
T6	0	1	1

EI
AC
ABC

EF
BC

CF
NULL
A
B
C
AB

CI

AB is counted throughout database once so set it as CF.

Step 6) After 5K transactions are read
A=3, B=5, C=3, AB=3, BC=2, AC=1, ABC=1;

Tran./ Item	A	B	C
T1	1	1	0
T2	0	1	0
T3	1	1	0
T4	1	1	1
T5	0	0	1
T6	0	1	1

EI	EF	CF	CI
	ABC	NULL	
		A	
		B	
		C	
		AB	
		BC	
		AC	

AC satisfy min. support as well as counted through whole database once, move it from EI to CF. ABC satisfy minimum support so move it from EI to EF.As BC is counted throughout database once so move it from EF to CF.

Step 7) After 6K transactions are read
A=3, B=5, C=3, AB=3, BC=2, AC=1, ABC=1;

Tran./ Item	A	B	C
T1	1	1	0
T2	0	1	0
T3	1	1	0
T4	1	1	1
T5	0	0	1
T6	0	1	1

EI	EF	CF	CI
		NULL	
		A	
		B	
		C	
		AB	
		BC	
		AC	
		ABC	

ABC is counted through whole database once move it from EF to CF.

Hence all frequent itemsets are counted.

4. IMPLEMENTATION AND ANALYSIS

The target of this section is to implement Apriori and proposed algorithm. Here the major concern includes comparison of execution time of both the algorithms.

4.1 Environment and Database

The programming language used for the implementation of algorithm is java (jdk 1.5.0) and tested on workstation with Pentium(R) Dual-Core CPU, 2.19 GHz and 2.93GB main memory.

4.2 Result analysis

For the experiment, we use datasets of different size shown in table 1. One is synthetic dataset obtained from dataset generator [12] and other dataset (Zoo) is obtained from the UCI repository of machine learning databases [13].

Table 1. Details of datasets used for comparison

Name of Dataset	Number of transactions	Number of items
Zoo	101	15
Synthetic Database	1000	10

4.2.1 Comparison of APRIORI & PAPRIORI on the basis of minimum support:

To study the relative performance of both algorithms, datasets transactions and support threshold with 35% to 75% are used. On the basis of varying support, execution time (in sec) for two datasets is drawn. Following are the graphs drawn for analysing the results.

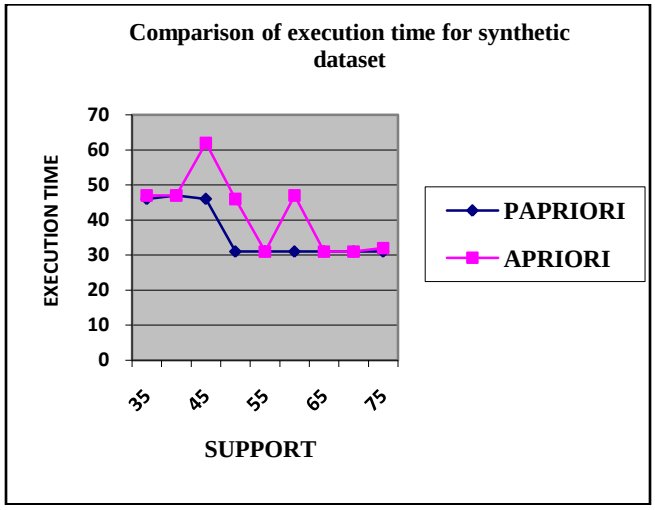

Fig. 1 Comparison of execution time for Synthetic Dataset

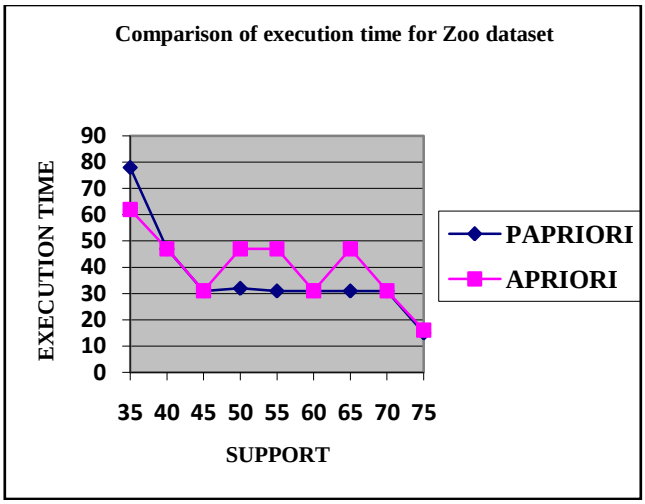

Fig. 2 Comparison of execution time for Zoo Dataset

It is clear from the Fig. 1 and Fig. 2 that at intermediate value of Support, execution time of PAPRIORI is less than APRIORI algorithm. If support is very less, overhead of PAPRIORI increases because more frequent itemsets will be generated that will give rise to more combinations of those frequent itemsets which in turn increases execution time. If support is very high, no frequent Itemset will be generated easily, which increases and execution time.

4.2.2 Execution time of PAPRIORI on varying K - Factor (Number of transaction read at a time from database:

Set intermediate value of support equal to 50% on the basis of Fig. 1 & Fig. 2. On the basis of K and fixed value of support, execution time (in sec.) can be drawn for analyzing the results. The graphs are as follows.

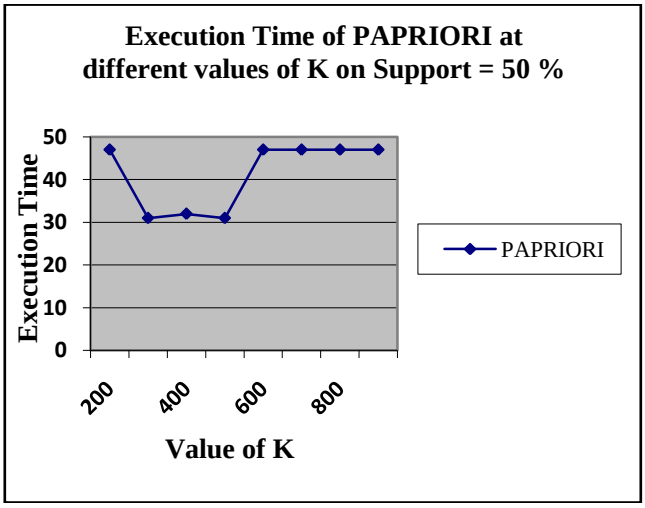

Fig. 3 Execution time for Synthetic Dataset on varying K

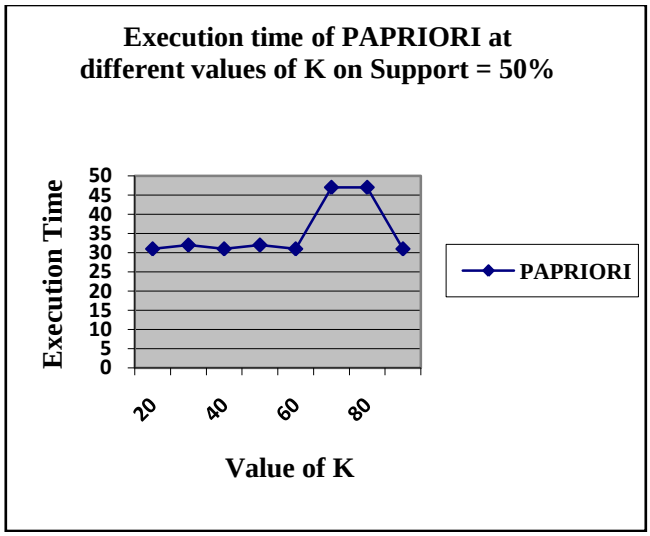

Fig. 4 Execution time for Zoo Dataset on varying K

It is obtained from the Fig. 3 and Fig. 4 that at intermediate value of K, execution time of PAPRIORI algorithm is less. So selection of right value of K is required. If value of K is very less, no frequent itemset can be obtained easily and execution time will increase. On the other hand, if value of K is very large then it will work as APRIORI algorithm.

5. CONCLUSION

We have proposed mining algorithm for incremental generation of large itemsets. Frequent itemsets discovered depends on value of parameters like support and number of transactions read at a time. Thus execution time of the algorithm depends on transactional datasets, minimum support value and value of K. At the end, it is concluded from experimental analysis that proposed algorithm (PAPRIORI) works effectively and efficiently as compared to APRIORI algorithm.

6. REFERENCES

[1] Herbert A. Edelstein, "Introduction to Data Mining and Knowledge Discovery," 3rd Edition, pp. 22-26, Oct. 1999.

[2] Tian Lan, Runtong Zhang and Hong Dai, "A New Frame of Knowledge Discovery," in Proc. 1st International Workshop on Knowledge Discovery and Data Mining, WKDD 2008, pp 607 – 611, Jan. 2008.

[3] R.Agrawal, T.Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," In Proc. of the ACM SIGMOD Conference on Management of data, Washington, D.C., pp 207-216, May 1993.

[4] *Rakesh Agrawal & Ramakrishan Srikant*, "Fast algorithm for mining Association rules," *IBM Almaden Research Center, 650 Harry road, San Jose, CA 95120: In proceedings of the 20th VLDB conference Santiago, Chile, pp 487-499, 1994.*

[5] J. Han, J. Pei, and Y. Yin., "Mining frequent patterns without candidate generation," In W.Chen, J. Naughton, and P. A.Bernstein, editors, 2000 ACM

SIGMOD Intl. Conference on Management of Data, pp 1-12, ACM Press, 2000.

[6] Ashok Savasere, Edward Omiecinski, Shamkant Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases," in proceedings of 21st VLDB Conference , Zurich , Switzerland, pp432-444, 1995.

[7] David W. Cheungt Jiawei Hant Vincent T. Ngtt C.Y. Wongj, "Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique," in proceedings of the 12th ICDE, New Orleans, Louisiania (IEEE) , pp 106-114,February 1996.

[8] Ratchadaporn Amornchewin, Worapoj Kreesuradej, "Mining Dynamic Databases using Probability-Based Incremental Association Rule Discovery Algorithm," Journal of Universal Computer Science, pp 2409-2428,Vol. 15, No.12, 28 June 2009

[9] Ratchadaporn Amornchewin, Worapoj Kreesuradej," Incremental Association Rule Mining Using Promising Frequent Itemset Algorithm", ICICS , 2007, IEEE.

[10] Nittaya Kerdprasop, and Kittisak Kerdprasop," Mining Frequent Patterns with Functional Programming", World Academy of Science, Engineering and Technology 2007.

[11] M.H.Margahny and A.A.Mitwaly," Fast Algorithm for Mining Association Rules", AIML 05 Conference, pp 19-21, December 2005, CICC, Cairo, Egypt.

[12] Dataset Generator (datgen) perfect data for an imperfect world, http://www.datasetgenerator.com/

[13] UCI repository of machine learning databases, http://archive.ics.uci.edu/