Algorithms for Mining Association Rules: An Overview

Sonam S. Chauhan
Department of Information Technology
Sipna College of Engineering and Technology
Amravati, India
Email: sonams.chauhan@yahoo.in

Dr. Prashant R. Deshmukh
Department of Computer Science and Engineering

Abstract— In this paper, we provide the basic concepts about association rule mining and compared existing algorithms for association rule mining techniques. Of course, a single article cannot describe all the algorithms in detailed, yet we tried to cover the major theoretical issues, which can help the researcher in their researches.

Keywords- Association rules, algorithm, itemsets, database.

I. INTRODUCTION

In [1] an researched techniques of data mining was presented called Association rule mining. It aims to extract interesting frequent patterns, correlations, associations among sets of items in the transaction databases or other data repositories. Various association mining algorithms will be briefly review and compared later.

The problem of finding association rules falls within the purview of database mining [2, 3], also called knowledge discovery in databases [4]. Related, but not directly applicable, work includes the induction of classification rules [4, 5, 6], discovery of causal rules [7], learning of logical definitions [8], fitting of functions to data [9], and clustering [10]. The closest work in the machine learning literature is the KID3 algorithm presented in [11]. If used for finding all association rules, this algorithm will make as many passes over the data as the number of combinations of items in the antecedent, which is exponentially large. Related work in the database literature is the work on inferring functional dependencies from data [12]. Functional dependencies are rules that require strict satisfaction.

In this paper, we surveyed some of the existing association rule mining techniques. The rest of the paper is organization as follows. Section 2 describes the problem of discovering association rules. Section 3 reviews some well known algorithm. In section 4, comprehensive studies on various algorithms to determine large item sets are presented. Finally, Section 4 concludes the paper.

II. PROBLEM DECOMPOSITION

The problem of finding all association rules can be divided into two sub problems [4]:

1. All sets of items (itemsets) that have transaction support more than minimum support is identified. The support for an itemset means, the number of transactions that contain the itemset. Itemsets with minimum support are referred as large itemsets, and all others items as small itemsets. In Section 3, we describe some of the existing algorithms like Apriori and AprioriTid and DHP, for solving this problem.
2. Using the large itemsets to get the desired association rules.

III. ALGORITHMS FOR MINING ASSOCIATION RULES

A. Algorithm Apriori

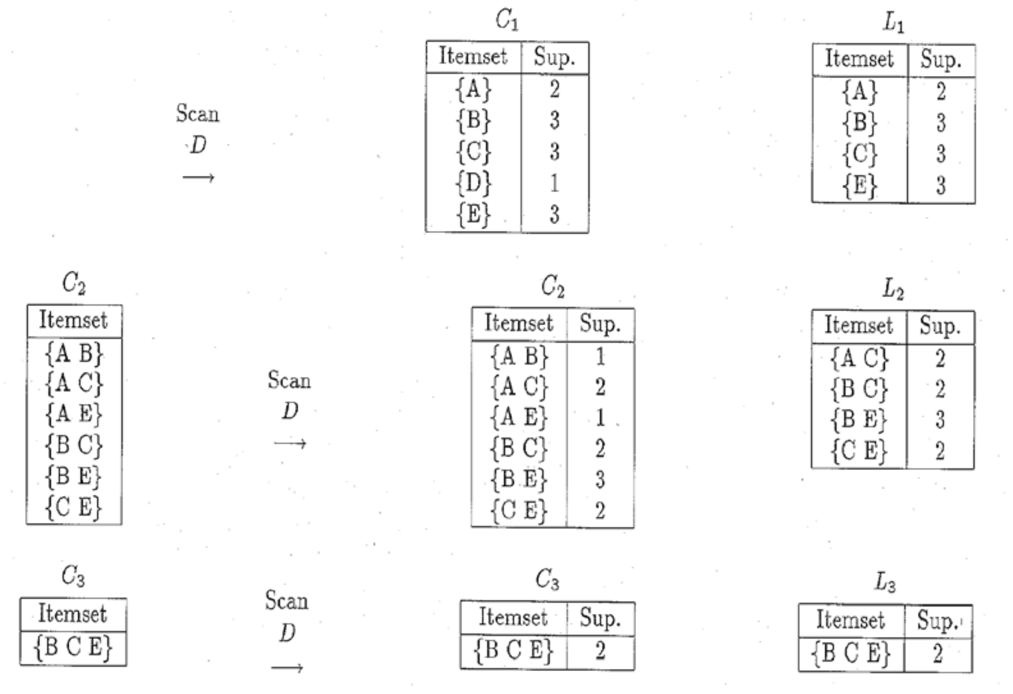
In Apriori algorithm the first pass simply counts the presence of item to get the large 1-itemsets. A subsequent pass, say pass 2, consists of two phases. First, the large itemsets L_{k-1} found in the (k-1) Th pass are used to generate the candidate itemsets C_k. Next, the database is scanned and the support of candidates in C_k is counted. For fast counting, it needs to efficiently determine the candidates in C_k that are contained in a given transaction t.

K-itemset	**An itemset having k items**
L_k	Set of large k-itemsets (Those with minimum support) Each member of this set has two fields: i) Itemset and ii) support count.
C_k	Set of candidate k-itemsets (Potentially large itemsets) Each member of this set has two fields: i) Itemset and ii) support count.

1) Examlpe

Consider an example of database given in fig2. In each insertion (or for each pass), it constructed a candidate set of large itemsets, count the number of time each candidate item is present and then determine large item set based on predetermined minimum support[7].In the first interaction ,simply scan all the transaction to count the number of occurrence for each item. The set of candidate 1-itemsets, C1 got is as shown in the fig2.Assume the minimum transaction support required is two, the set of large 1-itemset, L1, composed of 1-itemset with minimum support required can be determined. To discovered the itemset of 2-items such that any subset of a large itemset must also have minimum support, we use L1*L1 to determine a candidate set of itemsets C2 where * is concatenation operation in third case. C2 consist of **(|L1|/2)** 2-itemset. Next, the four transactions in D are scanned in D and scanned and support of each candidate item is encounter. The middle table of the second row in fig2 represents the result from such counting in C2. The itemset of 2-items, L2, is therefore generated based on the support of each candidate 2-itemset in C2.The set of candidate itemset, C3, is generated from L2 as follows. From L2, two large 2-itemset with the first same item, such as {BC} and {BE} are identified first. Then it test whether the 2-itemset {CE}, which consist of their second items, constitute large 2-itemset or not. Since {CE} is a large item set for itself, we know that all the subsets of {BCE} are large and then {BCE} becomes a candidate 3 itemset. There is no other 3-itemset from L2. It then scans all the transactions and discovers the larger 3-itemsetL3 in Fig2. Since there is no 4-itemset to be constituted from L3, it ends the Process.

TID	Itemset
100	ACD
200	BCE
300	ABCE
400	BE



C_1

Itemset	Sup.
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	Sup.
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A B}
{A C}
{A E}
{B C}
{B E}
{C E}

C_2

Itemset	Sup.
{A B}	1
{A C}	2
{A E}	1
{B C}	2
{B E}	3
{C E}	2

L_2

Itemset	Sup.
{A C}	2
{B C}	2
{B E}	3
{C E}	2

C_3

Itemset
{B C E}

C_3

Itemset	Sup.
{B C E}	2

L_3

Itemset	Sup.
{B C E}	2

Figure 2: Generation of candidate itemsets and large itemset

B. Algorithm AprioriTid

In AprioriTid algorithm after the first pass; database D is not used for counting support. Rather, the set $\overline{C}_k$ is used for this purpose, were $\overline{C}_k$ is the set of candidate K-itemsets when the TIDs of the generating transactions are kept associated with the candidates. Each member of the set $\overline{C}_k$ is of the form < TID, $\{X_k\}$> where each X_k is a potentially large k-itemset present in the transaction with identifier TID. If a transaction does not contain any candidate k-itemset, then C_k will not have an entry for this transaction. Thus, the number of entries in C_k may be lesser than the number of transactions in the database, particularly for large values of k. In addition, for large values of k, each entry may be smaller than the corresponding Transaction, because very few candidates may be contained in the transaction. However, for small values for k, each entry may be larger than the corresponding transaction because an entry in C_k includes all candidate k-itemsets contained in the transaction.

C. Algorithm DHP

DHP also generates a K-itemset by L_{k+1}. However, DHP is unique in that it employs the bit vector, which is built in the previous pass, to test the validity of each k-itemset. Instead of including all k-itemsets from $L_{k-1}*L_{k-1}$ into Ck, DHP adds a k-itemsets from C_k only if that k-itemset passes the hash Filtering. It can be seen later, such hash Filtering reduces the size of C_k. It later counts the support of candidate itemsets and to reduce the size of each transaction. A subset function is used to determine all candidate itemsets contained in each transaction.

IV. PERFORMANCE COMPARISION

As per the above discussion we tried to compare the three algorithms Apriori, Aprior Tid, and DHP for 4 to 6 passes against their execution time (sec). We graphically represent the comparison of all three algorithms as follow.

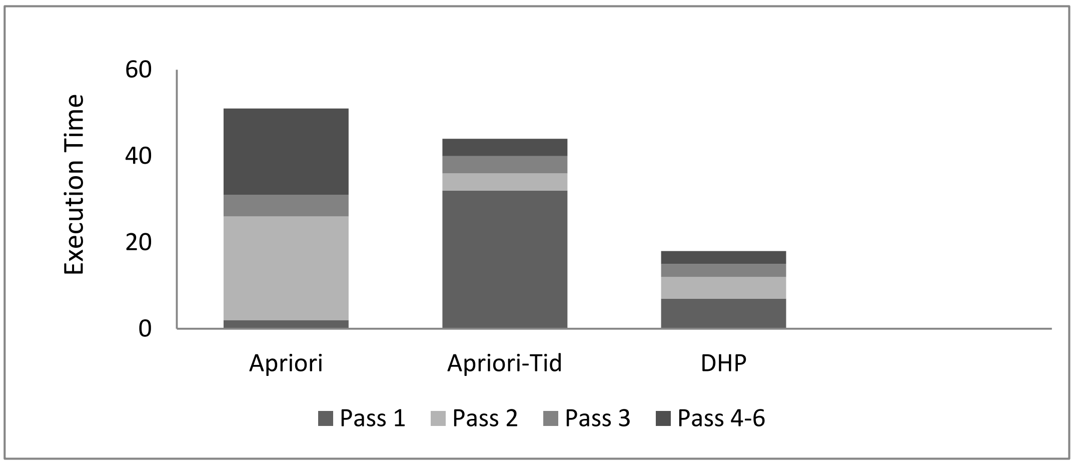


Figure3: Performance Comparison between algorithms

As per the discussion before and graphical explained we could clearly see that the Apriori algorithm performs the best in 1 pass as it comparatively taken very less time than other two to compute the itemset. However its performance decreases in the 2 and 4-6 passes. It can also be seen that after the first pass the performance of Apriori-Tid increases but it consumes a lot of time in its first pass. DHP beats both the algorithms and takes the least time to compute. It can be seen clearly that the execution time of the first two passes by Apriori is larger than the total execution time of DHP. Thus DHP algorithm provides the best performance.

V. CONCLUSION

In this paper, we listed out some of the existing algorithm for mining association rule and study their working. The algorithms work in two steps. In first step frequent items are found. Large itemsets are computed in second step. However while practically using the association rule mining tools several problems occurred such as many a time we do not get results in a reasonable time. It is widely identified that the set of association rules can rapidly grow which become very difficult to handle, especially as we decrease the frequency requirements. The larger the set of frequent itemsets the greater the number of rules given to the user, many of which are redundant. This is true even for smaller datasets, but for larger datasets it is simply not probable to mine all possible frequent itemsets, let alone to generate rules, since they typically produce very large number of frequent itemsets. Although several different solutions have been proposed to solve efficiency issues, they are not always successful.

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] R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. IEEE Transactions on Knowledge and Data Engineering, 5(6):914{925, December 1993. Special Issue on Learning and Discovery in Knowledge-
[3] Based Databases. M. Holsheimer and A. Siebes. Data mining: The search for knowledge in databases. Technical Report CS-R9406, CWI, Netherlands, 1994.
[4] G. Piatestsky-Shapiro, editor. Knowledge Discovery in Databases. AAAI/MIT Press, 1991.
[5] R. Agrawal and R. Srikant. "Fast algorithms for mining association rules in large databases". Research Report RJ 9839, IBM Almaden Research Center, San Jose, California, June 1994.
[6] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. Wadsworth, Belmont, 1984.
[7] J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases: An attribute oriented approach. In Proc. of the VLDB Conference pages 547{559, Vancouver, British Columbia, Canada, 1992.
[8] J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufman, 1993.
[9] P. Langley, H. Simon, G. Bradshaw, and J. Zytkow. Scienti_c Discovery: Computational Explorations of the Creative Process. MIT Press, 1987.
[10] S. Muggleton and C. Feng. Efficient induction of logic programs. In S. Muggleton, editor, Inductive Logic Programming. Academic Press, 1992.
[11] J. Pearl. Probabilistic reasoning in intelligent systems: Networks of plausible inference, 1992.
[12] G. Piatestsky-Shapiro. Discovery, analy- sis, and presentation of strong rules. In G. Piatestsky-Shapiro, editor, Knowledge Discovery in Databases. AAAI/MIT Press, 1991.
[13] R. Brachman et al. Integrated support for data archeology. In AAAI-93 Workshop on Knowledge Discovery in Databases, July 1993.
[14] R. Krishnamurthy and T. Imielinski. Practitioner problems in need of database research: Re- search directions in knowledge discovery. SIG- MOD RECORD, 20(3):76{78, September 1991.