ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Apriori Algorithm for Vertical Association Rule Mining

Mohammed Karimuddin*, M.Prudhvi Ravi Raja Reddy
*Final M.Tech, Dept of CSE, Sri Aditya Engineering College, Surampalem, Kakinada, India
Assistant Professor, Dept of CSE, Sri Aditya Engineering College, Surampalem, Kakinada, India
karim.java2@gmail.com

Abstracts

Association rule mining is one of the important concepts in data mining domain for analyzing customer's data. The association rule mining is a process of finding correlation among the items involved in different transactions. Traditionally association rule mining is implemented horizontally. For this we have plenty of different algorithms in research like Apriori based, FP tree based so on. Recently we have a new method in association rule mining which generates vertical association rules. In horizontal association rule mining we read transaction items record by record basis and computes support of each frequent item or candidate item. We repeat this process to generate frequent item sets. The vertical association rule mining evaluates support frequency of each item column wise. For this it uses bitmap matrix that saves support sets of frequent item sets in memory which is used to calculate candidate item sets. The Item are read from Data set using BitMap Matrix format which uses 1 or 0 to represent the presence or absence of item in record.

In our system it is proposed to combine both horizontal mining and vertical mining in generating association rules. The horizontal and vertical mining are implemented in parallel using multithreading concept. For this we propose a modified parallel multithreaded Apriori algorithm. The algorithm saves time and decreases memory space as the process is running because of bitmap representation of dataset and bitmap compression algorithms.

Keywords: Data mining, Association Rule Mining, Frequent item sets, Candidate item sets, Horizontal mining, Vertical mining, Apriori, Bitmap Apriori, Parallel multithreaded Apriori.

Introduction

sData Mining is an emerging concept which is powerful and promising and advances in data collection and storage technologies have led organizations to store vast amount of data pertaining to their businesses and extracting useful information from such vast amount of data is an important activity and is referred to as Data Mining or Knowledge Discovery in Databases (KDD) [1][2][3], which includes the key design aspects like Knowledge Discovery, Classification, Clustering and Association Rule Mining [1][2][3][5][6][7][8][9][10][11][12]. Association Analysis is the process of discovering the association rules attribute-value conditions that occur frequently together in the given data set which does by analyzing the data warehouses or the vast amount of data [5][9]. Association Rule Mining represents a data mining technique and its goal is to find the interesting association relationships among the large set of data which are used by the businesses in decision making processes their by getting the profits. The choice of making an algorithm is a task which is based on many conditions like accuracy, efficiency, security, scalability and number of database scans. Rule Support and confidence are the two measures of interestingness [5][9][10]. Strong association rules are the association rules which hold the minimum support value and minimum confidence value and are considered in the decision making processes. Association rule mining is best explained by one of its common applications Market Basket Analysis [1][2][5], where in retrieving the association between the items that the customers purchase. Many Algorithms are proposed for the association rule mining in finding the association rules and several optimizations are available in generation of association rules. This paper proposes the generation of association rules using horizontal apriori as well as vertical apriori in parallel combining both the approaches using the multithreading concept available in java [4].

This paper is organized as follows: Section 2 describes the Preliminaries. Section 3 describes Vertical Apriori Section 4 describes Multithreading Apriori Section 5 describes Implementation Details. Section 6 presents experimental details and performance analysis. Section 7 presents conclusion.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

Preliminaries

Advances in the data collection and storage technologies has increased the sizes of the data sets drastically up to an extent that the algorithms which are used to analyze the data set need to choose depend upon the factor of scalability, which increases its execution time linearly with the increase in the size of the data set or the number of records with sustained capacity of main memory. Hence efficient scalable algorithms for data mining in very large data set are widely studied.

A. Market Basket Analysis

Finding frequent item sets plays an important role in data mining and is regarded as the first step in the generation of association rules. Generation of Association rules are mostly explained by the market basket analysis [1][2][5], retrieving the association between the items at the customer's purchases. Here products or sets of items (item-sets) which occur in many transactions are found.

Table 1.Transactional Data

TID	*List of Item_IDs*
T10	I1 I2 I3
T20	I1 I2 I3 I4 I5
T30	I1 I3 I4
T40	I1 I3 I4 I5
T50	I1 I2 I3 I4
T60	I2 I3 I4 I5
T70	I2 I3 I4
T80	I2 I4 I5
T90	I2 I4
T100	I2 I3
T110	I3 I4 I5
T120	I3 I4
T130	I3 I5
T140	I3 I4 I5
T150	I2 I3 I4 I5

Table 1 describes several transactions (T10, T20...) stored in relational database and the corresponding column mentions the list of item ids for the particular transaction. Frequent sets are those sets whose support is above the user defined minimum support. In Association rule mining the minimum support and the minimum confidence are the two measures of interest and those rules which satisfies the user defined minimum support and confidence values are considered as strong and are considered in decision making process. The Market Basket Analysis plays important role in understanding customer purchasing interests and behavior which helps in increasing company product sales.

B. Apriori Algorithm

A set of items is refereed as itemset. An itemset that contains k items is a k-itemset. The occurrence frequency of an itemset is the number of transactions that contain the itemset. This is known as frequency or support count or count. If the support count of an itemset satisfies the user defined minimum support then the item set is frequent item set. Apriori algorithm [1][2][3][5][9][10] proceeds by first by finding all the frequent itemsets and then generating the strong association rules by considering the minimum support and confidence measures specified by the user.

Finding the frequent itemsets among the given transactional data is again a two step process, first the formation of candidates (candidates Generation) in which the candidate frequent itemsets are generated and counting of the candidates (candidate Counting) in which the generated candidate itemsets count is calculated from the transactional data and only those candidate itemsets are considered as frequent itemsets which satisfies the user specified minimum support [5][9]. This algorithm finds all the i-frequent element itemsets at i^{th}-stage.

During the 1^{st} phase or stage algorithm finds the 1-frequent element itemsets by calcuting the support of the itemsets from the transactional database. Thus, after the first phase all frequent 1-element sets are known. However, Apriori reduces the number of sets of candidate sifting - a priori - those candidates who may not be frequent, based on information received at previous stages of the information on which of the sets are the most abundant. Screenings are based on the simple assumption that if the set is frequent, all its subsets must also be frequent. Thus, before the counting of candidates step algorithm can reject any candidate set, a subset of which is not frequent. This process is continued until the number of frequent n-item-sets becomes zero, where n determines the no. of children in the item-set [2][5].

Consider the database presented in Table 1. Suppose that the minimum support count threshold is 5. That is, to be a frequent item-set, there should be at least two transactions, consisted of the particular item-set. In the first stage, all the products individually are sets of candidates and counted during the counting step, the candidate. At the second stage, the candidate may be only a couple of items, each of which is frequently encountered. For example, initially all the sets of single items ({I1}, {I2}, {I3}, {I4} and {I5}) have a support count of 5, 9, 13, 12, and 8 respectively. So initially all five items become frequent item-sets. Thus, the second stage of the algorithm will form the following list of sets of candidates. Table 2 shows the item sets along with their support counts. Now

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

frequent 2-item- sets are {I1, I3}, {I2, I3}, {I2, I4}, {I3, I4}, {I3, I5} and {I4, I5} as the other item-sets don't have the minimum support count.

Table 2. Itemsets and the support counts

Itemset	*Sup. Count*
{I1, I2}	3
{I1, I3}	5
{I1, I4}	4
{I1, I5}	2
{I2, I3}	7
{I2, I4}	6
{I2, I5}	4
{I3, I4}	10
{I3, I5}	7
{I4, I5}	7

Likewise this process is continued until the no of frequent n-item-sets become zero, where n determines the no. of children in the item-set. Apriori counts not only the provision of all frequent sets, but also ensuring those sets of candidates, which could not be discarded a priori.

C. Algorithm Optimizations

Many optimizations to the primary data mining algorithms are proposed over the time. They focus on a narrow sub set of datasets or cater the mining of the data broadly.

1) Pruning

In this Pruning techniques are used to reduce the size of candidate set which is an optimization [6]. Pruning strategy for the algorithm is based on a characteristic: a frequent item-set is a set, if and only if all its subsets are frequent.

2) Dynamic hashing and pruning

A hash based algorithm (DHP) [7], which generates candidate large itemsets efficiently, while reducing the size of the transaction data base. Number of candidate itemsets generated by DHP is smaller than that generated using existing algorithms, making it efficient for large itemsets, specifically for the large 2-itemsets.

3) Parallel Data mining

DHP is extended into a parallel data mining algorithm, facilitating parallel generation of the candidate Itemsets and parallel determination of large itemsets [8].

4) Transaction Reduction

In this algorithm we reduce the number of transactions scanned in the future iterations [9][10]. A transaction that does not contain any frequent k-itemsets can not contain any frequent (K +1) itemsets, hence can be removed.

5) Partition

In this Optimization technique Partition-based highly parallel algorithm [11], which logically divides the data base into several disjoint blocks? Each considered separately a sub- block and it generates all the frequent sets. The generated frequency of collection is used to generate all possible frequent sets, and followed by the final calculation of the degree of support of these itemsets. Sub-block size is chosen to allow each sub-block be placed in the main memory, each stage scanned only once. This algorithm is highly parallel, and can be allocated to each sub-block of a processor that generates a frequency set. Frequency set is resulting, after the end of each cycle. The processor to generate the overall communication between the candidate k-itemsets. Usually here the communication process is a major bottleneck in algorithm execution time; while on the other hand, each individual processor generates frequent set time is also a bottleneck. Other methods are shared between multiple processors in a hash tree to generate frequent sets.

6) Sampling

A subset of the sample is taken for mining in the sampling based mining techniques. In which the rules are produced with the first extracted sample from the whole dataset [12]. The remaining of the dataset is used to authenticate the dataset that is chosen. Sampling based algorithms significantly reduced the I/O costs, nevertheless with inaccurate results often and distortions in the data, known as data-skew.

Vertical data mining

Vertical data mining Apriori algorithm [1] mines frequent patterns from a horizontal data format which represents the items categorized into particular transactions, where vertical data format represents data as transactions categorized into particular items. Hence, for a particular item, there is a set of corresponding transaction ids. Instead of horizontal data format, Apriori can be extended to use vertical data format for efficient mining. Table III shows the transactional data represented in Table I, in the vertical format. As can be seen from the Table 3, the vertical format data mining only has to parse the dataset once to get the itemsets. For the itemset generation from the 2nd itemset, it only needs to refer the previous itemset [1]. This eliminates the need to parse through the dataset each time to count the frequency of itemsets, for each round. Hence, relative to the algorithms developed for the use on databases with the horizontal layout of data, the algorithms developed for the vertical representation tend to be more optimal.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

Table 3. Transactional data represented in the Vertical Format

Item ID	List of TIDs	Support Count
I1	T10, T20, T30, T40, T50	5
I2	T10, T20, T50, T60, T70, T80, T90, T100, T150	9
I3	T10, T20, T30, T40, T50, T60, T70, T100, T110, T120, T130, T140, T150	13
I4	T20, T30, T40, T50, T60, T70, T80, T90, T110, T120, T140, T150	12
I5	T20, T40, T60, T80, T110, T130, T140, T150	8

As the minimum support specified by the user is 5 all the 1-itemset are frequent and now consider Table 3 in order to calculate the 2-candiadate itemset first then pruning based on the support gives us the 2-frequent itemset without scanning the databases again and again which reduces the number of database scan's required in order to calculate the frequent item sets and this process is continued until the no of frequent n-item-sets become zero. This optimization reduces the number of database scans which also reduces the amount of time required to calculate the frequent item sets and correspondingly reduces the amount of space required in order to find the frequent item sets and their by improves the efficiency of the algorithm.

Multithreaded apriori

Java provides built-in support for multithreaded programming [4] . A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread, and each thread defines a separate path of execution. Thus, multithreading is a specialized form of multitasking. However, there are two distinct types of multitasking: process- based and thread-based. It is important to understand the difference between the two. Process-based multitasking is the more familiar form. A process is, in essence, a program that is executing. Thus, process-based multitasking is the feature that allows your computer to run two or more programs concurrently. For example, process-based multitasking enables you to run the Java compiler at the same time that you are using at text editor. In process- based multitasking, a program is the smallest unit of code that can be dispatched by the scheduler. In a thread based multi tasking environment, the thread is the smallest unit of dispatchable code. This means that a single program can perform two or more tasks simultaneously. For instance, a text editor can format text at the same time that it is printing, as long as these two actions are being performed by two separate threads. Thus, process-based multitasking deals with the "big picture," and thread-based multitasking handles the details.

Multitasking threads require less overhead than multitasking processes. Processes are heavy weight tasks that require their own separate address spaces. Inter process communication is expensive and limited. Context switching from one process to another is also costly. Threads, on the other hand, are lightweight. They share the same address space and cooperatively share the same heavy weight process. Inter thread communication is inexpensive, and context switching from one thread to the next is low cost. While Java programs make use of process- based multitasking environments, process-based multitasking is not under the control of Java. However, multithreaded multitasking is. Multithreading enables you to write very efficient programs that make maximum use of the CPU, because idle time can be kept to a minimum. This is especially important for the interactive, networked environment in which Java operates, because idle time is common.

The Java run-time system depends on threads for many things, and all the class libraries are designed with multithreading in mind. In fact, Java uses threads to enable the entire environment to be asynchronous. This helps reduce inefficiency by preventing the waste of CPU cycles. The value of a multithreaded environment is best understood in contrast to its counterpart. Single-threaded systems use an approach called an event loop with polling. In this model, a single thread of control runs in an infinite loop, polling a single event queue to decide what to do next. Once this polling mechanism returns with, say, a signal that a network file is ready to be read, then the event loop dispatches control to the appropriate event handler. Until this event handler returns, nothing else can happen in the system. This wastes CPU time. It can also result in one part of a program dominating the system and preventing any other events from being processed. In general, in a singled-threaded environment, when a thread blocks (that is, suspends execution) because it is waiting for some resource, the entire program stops running. The benefit of Java's multithreading is that the main loop/polling mechanism is eliminated. One thread can pause without stopping other parts of your program. For

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

example, the idle time created when a thread reads data from a network or waits for user input can be utilized elsewhere. Multithreading allows animation loops to sleep for a second between each frame without causing the whole system to pause. When a thread blocks in a Java program, only the single thread that is blocked pauses. All other threads continue to run. Threads exist in several states. A thread can be running. It can be ready to run as soon as it gets CPU time. A running thread can be suspended, which temporarily suspends its activity. A suspended thread can then be resumed, allowing it to pick up where it left off. A thread can be blocked when waiting for a resource. At any time, a thread can be terminated, which halts its execution immediately. Once terminated, a thread cannot be resumed.

In this approach we are combining the both the horizontal association rule mining using apriori as well as the vertical association rule mining using apriori algorithm in order to find the frequent itemset their by finding the association rules running them each as thread and by multithreading.

Implementation details

In this section, we analyze the performance of our parallel Multithreaded Apriori Algorithm for vertical Association rule mining for mining frequent itemsets and their by generating the association rules. The algorithms were implemented in java language. Swing framework is used for designing GUI. We have placed transactional data records in data sets. The MS Access data base is used for managing the performance results.

Experimental results

In order to evaluate the performance of our proposed algorithm, we have conducted experiments on a PC (CPU: Intel(R) Core2Duo, 3.16GHz) with 4GByte of main memory running Windows XP. We have used java Swing to design front end, java libraries to construct code for building association rule generation system. We also used JFree Chart a third party library product for generating the charts for showing the performance analysis of algorithms.

The following shows the results of Horizontal Association Rule Mining Using Apriori

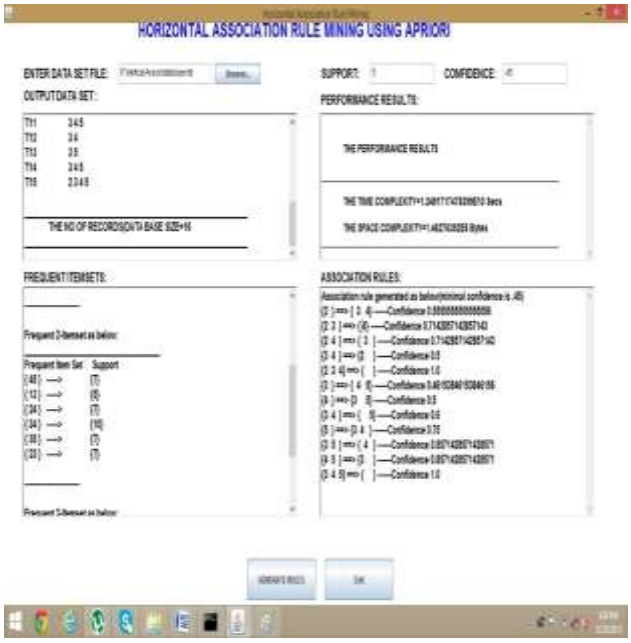

Fig 6.1 Results of Horizontal Association Rule Mining Using Apriori

The following shows the results of Vertical Association Rule Mining Using Apriori

Fig 6.2 Results of Vertical Association Rule Mining Using Apriori

The following shows the results of Horizontal Association Rule Mining and Vertical Association Rule Mining Using Apriori.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

The following shows the Analysis of Time complexity when comparing Horizontal and Vertical Apriori.

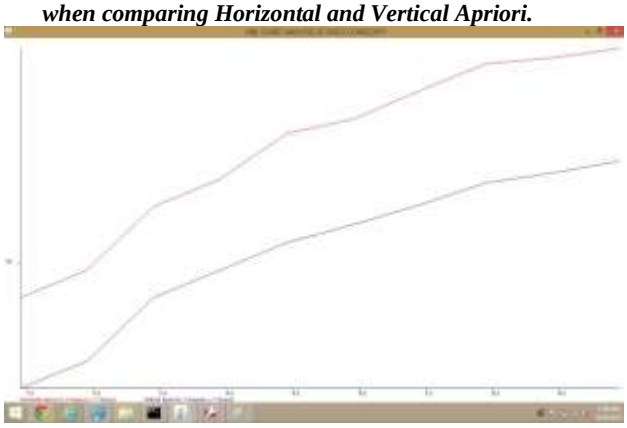

Fig 6.3 Analysis of Time complexity for Horizontal and Vertical Apriori.

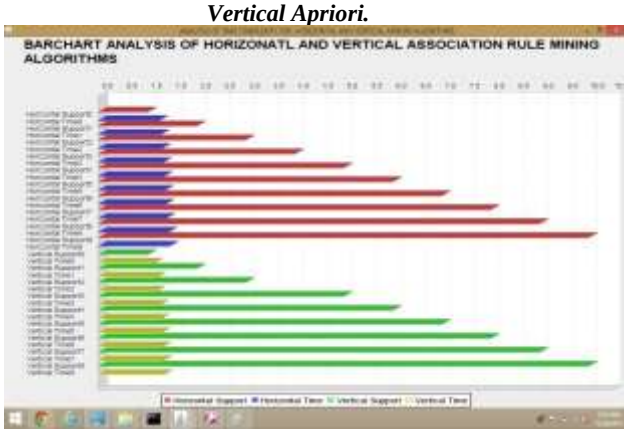

Fig 6.4 Analysis of Time complexity for Horizontal and Vertical Apriori.

The following shows the Analysis of Space complexity when comparing Horizontal and Vertical Apriori.

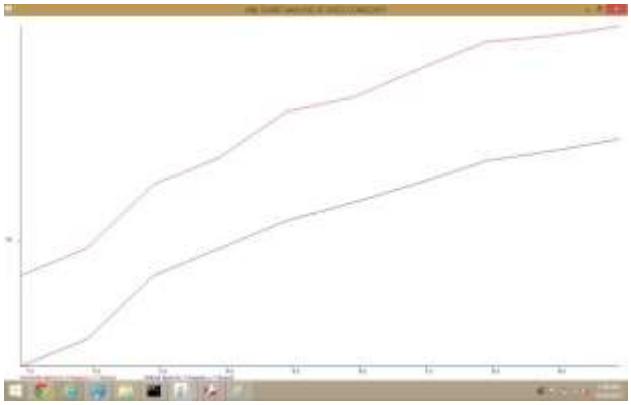

Fig 6.5 Analysis of Space Complexity for Horizontal and Vertical Apriori

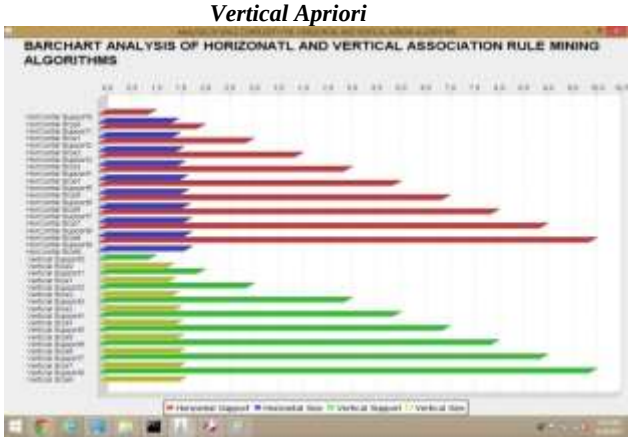

Fig 6.6 Analysis of Space Complexity for Horizontal and Vertical Apriori

Conclusions

In this paper, we proposed a new technique for mining Frequent Item Sets that uses vertical Association rule mining with Bit Map Matrix. This Technique, we compare with Horizontal Apriori Our approach reduces the total process and also takes less time , less memory.

References

1. "Horizontal Format Data Mining with Extended Bitmaps", Buddhika De Alwis1, Supun Malinga2, Kathiravelu Pradeeban3, Denis Weerasiri4, Shehan Perera, International Journal of Computer Information Systems and Industrial Management Applications. ISSN 2150-7988 Volume 4 (2012) pp. 514–521
2. J. Han and M. Kamber. "Data mining: Concepts and Techniques", Morgan Kaufman, San Francisco, CA,2001.
3. Arun K Pujari "Data Mining Techniques" UniversityPress (India) Pvt. Ltd 2001

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

4. "Java.The.Complete.Reference", Herbert Schildt, McGrawHill, 7th.Edition.Dec.2006
5. R. Agrawal, and R. Srikant. "Fast algorithms for mining association rules," in Proceedings of. 1994 Int. Conf. Very Large Databases (VLDB'94), Sep. 1994.
6. H. Mannila, H. Toivonen, and A. Verkamo. Efficient algorithm for discovering association rules. AAAI Workshop on Knowledge Discovery in Databases, pp 181-192, Jul. 1994.
7. J.S. Park, M.S. Chen, and PS Yu. "An effective hash-based algorithm for mining association rules". In Proceedings of ACM SIGMOD International Conference on Management of Data, pp 175-186, May 1995.
8. J.S. Park, M.S. Chen, and P.S. Yu. "Efficient parallel data mining of association rules," 4th International Conference on Information and Knowledge Management, Baltimore, Maryland, Nov. 1995.
9. R. Agrawal, and R. Srikant. "Fast algorithms for mining association rules," in Proceedings of. 1994 Int. Conf. Very Large Databases (VLDB'94), Sep. 1994.
10. J. Han and Y. Fu. "Discovery of multiple-level association rules from large databases," in Proceedings of. Int. Conf. Very Large Databases (VLDB'95), pp 402-431, Sep. 1995.
11. A. Savasere, E. Omiecinski, and S. Navathe. "An efficient algorithm for mining association rules in large databases," in Proceedings of the 21st International Conference on Very Large Database, pp 432-443, Sep. 1995.
12. H. Toivonen. "Sampling large databases for association rules," in Proceedings of the 22nd International Conference on Very Large Database, Bombay, India, pp 134-145, Sep. 1996.