Journal of Current Research in Engineering and Science
JCRES
Bi-Annual Online Journal (ISSN : 2581 - 611X)

Volume 4- Issue 2, Paper1 July 2021

AN INNOVATIVE MACHINE LEARNING MODEL FOR DYNAMIC FREQUENT PATTERN MINING

Yamuna devi. N
Associate Professor,
Department of Computing, Coimbatore Institute of Technology, Coimbatore, India
yamunaacit@gmail.com

Abstract— *Frequent patterns are most occurring forms present in a huge collection of data. Extracting these frequent patterns is always a challenging task. The extracted frequent patterns are evaluated for their usefulness using different measures in different techniques. Association rule mining techniques are very effective to mine frequent patterns. A new algorithm namely, Direct-Vertical algorithm proves its efficiency with minimum execution time. The complexity of this algorithm is measured in terms of number of transactions in the input database and the maximum length of transactions. Since its combinatorial approach does not depend on the unknown number of processing levels and does not generates unnecessary candidate sets and subsets, it gives better performance than the existing methods. Further, the efficiency of this algorithm is improved by parallel execution of the algorithm on number of clusters of database. The parallel version this algorithm is developed and implemented using parallel architecture based software. This paper explains the parallel Direct-Vertical algorithm and its execution in distributed environment.*

Keywords — *Frequent itemset mining, Parallel frequent pattern mining, parallel algorithm.*

I. INTRODUCTION

Data mining is a well defined research domain in which a large set of new algorithms and techniques have been proposed in last decades. Since the innovations and rapid development in various fields such as Online shopping, mobile applications, social networks and web usage cause a huge amount of data generation. This increases the need for advanced and innovative algorithms to mine frequent patterns in fast manner. Among many newly developed algorithms, the Direct-Vertical algorithm put forwards a different way of mining frequent patterns on the fly. The Direct-Vertical algorithm updates the result set of frequent patterns whenever a new set of transactions is appended into the input database. It generates all the k-itemset frequent sets for each and every transaction of the database in sequential order. The algorithm is very efficient in terms of execution time and not needed to be re-executed for each modification of input transaction database. Considering a further improvement, the Direct-Vertical algorithm is parallelized in efficient manner to be executed in a parallel environment. The implementation of parallel Direct-Vertical algorithm is achieved using Hadoop architecture and Map-Reduce Framework. The parallel algorithm is developed in such a way that it splits the input transaction database among many clusters of machines and the mining process is done in parallel manner.

II. RELATED WORK

Mining frequent patterns from huge data is achieved using mining algorithms. The Apriori algorithm is the first and foremost method to mine frequent patterns[2]. The limitations of Apriori algorithm are notified as the large number of scans and generation of huge quantity of candidate sets at each level. Alternatively, the Vertical data format method is suggested with only one database scan and transforms the input database into vertical form to proceed with mining process. Further, many methods and techniques have been suggested with improvements [19], [20]. Among them, frequent pattern tree

growth algorithm [6] eliminates large candidate itemset generation but the process of generating tree data structure and pruning process is considered as lacking part. Agrawal et al. developed AprioriTid algorithm [2] which uses the set $\bar{C}_{k-1}$ to prune candidate sets in C_k and produces $\bar{C}_k$. In this algorithm the database scan is reduced to one time but the huge candidate sets are generated as in Apriori method.

Bo Wu and Defu Zhang et al. presented Apriori-Growth Algorithm [3] that combines Apriori algorithm and FP-Growth to mine association rules. Goswami D.N. et al [4] proposed a new algorithm using record filter approach. Jaishree et al. explained transaction reduction method to improve efficiency [5]. Jnanamurthy HK et al discussed mining maximal frequent item sets using subset creation [8]. Sheila A. Abaya explained a method that minimizes candidate generation [18]. The probability concept in used in Apriori algorithm by Sunil Kumar et. Al. [21]. In all the above said and new developments, the thing which cannot be avoided is the generation of huge volume of candidate sets. The algorithm that avoids or reduces the generation of candidate sets will further improves the performance of frequent pattern mining.

A novel sequential algorithm, namely Direct-Vertical (DV), with exactly one scan of the transaction database was designed and developed to mine frequent patterns[26]. It is a good quality method that totally removes candidate itemset generation and pruning process. The DV algorithm generates the complete set of frequent itemsets efficiently in a minimum duration. This algorithm is capable of processing dynamic databases where the transactions are added at frequent intervals. The enhancement of DV algorithm is focused on handling a large volume of data efficiently in this paper. It is difficult for a sequential algorithm to process the large transaction databases on a single machine within a stipulated time. To overcome this drawback, it is proposed to develop algorithms that execute on a distributed environment.

The parallel version of Eclat was implemented by many researchers. Manalisha Hazarika and Mirzanur Rahman developed parallel version of Eclat using equivalence class concept in synchronous and asynchronous phases[12]. Sandy proposed Dist-Eclat, a distributed Eclat algorithm which partitions the input database among nodes and discussed with communication costs [16]. Another parallel method of Eclat algorithm is discussed by Jianwei Li et al using equivalence classes [7]. Apart from these, many parallel versions of Apriori and FP-Tree growth algorithms were developed and implemented [24].

Xianfeng Yang and Liming Lian proposed a new Fastman parallel algorithm based on map reduce structure to mine frequent patterns [25]. The authors compared the performance with clara algorithms. M.J.Zaki et al also proposed a parallel data mining algorithm for association rule generation[27]. Madhavi Vaidya discussed clustering concept in mining using map reduce model [11]. Jongwook Woo used map reduce structure for solving market basket problem [9]. Mohammed J Zaki et al elaborated a parallel algorithm for association rule mining. They used itemset clustering method and equivalence class concept in their proposal [13]. Ning Li et al implemented a parallel apriori algorithm based on map reduce [14]. Syed Khairuzzaman Tanbeer et al developed parallel frequent pattern tree mining algorithm which was implemented in distributed environment [22]. Phongphun Kijsanayothin et al used the mapreduce concept in Big data analytics[15].

In this paper, the Parallel Direct Vertical (PDV) model is designed and developed to mine the frequent patterns using Map-Reduce programming model. The PDV model uses static load balancing and data parallelism. It uses the hierarchical memory architecture in which both distributed and shared memory concepts are implemented. This model has two levels of map and a reduce functions. It executes DV algorithm as a part of level1 map function. This model uses a hash table to store the individual items along with their occurrences. The hash table is accessible by the map functions and is used for intersection process.

The equivalence class is efficiently used in PDV model to group the frequent itemsets based on their common prefix up to a certain length. These groups are handled by mapper to generate next level candidate itemsets and further frequent itemsets of high order. It uses distributed memory architecture. The equivalence class concept is effectively applied to reduce the execution time by making the mapper independent to generate the candidate itemsets.

III. DV ALGORITHM – A STUDY

The DV algorithm [26] was developed to mine the frequent patterns in totally different way using combinatorial method [17]. The direct-vertical algorithm follows the vertical data format and intersection process as in ECLAT. It generates the frequent patterns on the fly while reading the input transaction database D transaction-wise. The algorithm reads one transaction at a time and all possible k-itemset frequent sets corresponding to each transaction are instantly generated when it is read. Based on the minimum support, the frequent 1-itemsets are considered from the current transaction for making ordered combinations. These combinations are checked for minimum support threshold by intersection method. The passed combinations are stored in their corresponding k-itemset table.

This algorithm requires only one scan of the transaction database to generate the set of all frequent itemsets. It does not generate any candidate sets and subsets and hence there is no pruning process. All infrequent itemsets will be filtered automatically on the fly. The number of combination of items generated without repetition is calculated as n!/(n-k)! k! where n is the number of frequent 1-itemsets in the current transaction T_i, k = 2,3,…n (k-itemsets). Intersection is done for the itemset combination which is not a reproduction to calculate the support value. The intersection process generates TID-set for each combination. The combination which satisfies minimum support threshold is considered as frequent set. So, all possible k-itemset frequent sets are generated in a single stretch for the items in each transaction while it is read from the database. The absolute support count for each frequent itemset is the length of the TID_set of the corresponding itemset. The algorithm is given in Figure 1.

```
Algorithm Direct_Vertical (D, min_Support)
Initialize i=1, support=0;
while (i<= n) do        //for each transaction Ti in D & n is the total number of transactions in D
{
   read transaction Ti;
   for each item Ij ∈ Ti,
      include Ti in 1-itemset table against Ij and increment support of Ij;
   count Ij in Ti that have support >= minSupport and move to frequent list S;
   if count<2 then  {i=i+1; Break;}
   else {
         $C_k$ = Produce_Combinations (S) // Refer Section 4.1
```

```
        for each Cj $\in$ {$C_k$}
        {
            if Cj already exist, then append Ti against Cj;
            else {
                S1 = intersection of the TID set of each item in Cj;
                if count(S1) >= minSupport then
                    include S1 against Cj;
                }
        }
    }
i=i+1;
}
```

Fig 1 Direct Vertical Algorithm

IV. PARALLEL DV ALGORITHM

A. Need of Map/Reduce

The Map/Reduce programming model is suitable for algorithms or applications which are large-scale data intensive, requires the data in the format of <key, value> pair[10]. The input transaction database is in the form of <Tid, itemset> that is considered as key-value pair. It justifies the proposed parallel algorithm with Map/Reduce approach as the input database is highly compatible with the required format.

B. Use of Equivalence Clause

The equivalence clause is used in the proposed algorithm to implement join step that generates combinations of itemsets after finding 2-item frequent sets and further in all the following levels. In each level, the k-itemsets are partitioned into equivalence class records and are fed as input to each mapper for further processing. The results of these mappers are independent and there is no common itemsets as a consequence of equivalence class concept. This helps to have one time execution of the reduce function.

C. Proposed Method

In the proposed algorithm, there is two level map function and one reducer function. In first level map function**, t**he original input dataset is split into chunks and are stored in all mappers where each record is read as a string. Consequently, as a first step, this map function executes direct-vertical algorithm which generates a set of k-itemset frequent sets and infrequent sets. All the frequent k-itemsets, where k>1, are saved in a global output file namely L1_DB. As a second step, it generates 2-itemset candidate sets and filters frequent 2-itemsets to store them in the same file. The first level map function results in all 1-itemsets and infrequent k-itemsets, where k>1. The second level map function is responsible for generating all frequent k-itemsets, where k>2. The input phase in second map function reads the dataset from L1_DB and divides the k-itemsets into equivalence classes, The map function starts execute with reading 2-itemset equivalence classes and generates k+1-itemsets until there are no items in the output file. Each mapper executes individually without overlapping, as the input records are equivalence class records which are exclusive. The algorithm is given in Algorithm 1.

Algorithm 1: Map Function

```
Input Phase:
    Generate split files.
Level 1 Map Phase:
1.  Execute Direct-vertical algorithm and generate the data set
     Vi : <key_itemset_i, value_(list of transaction, count)_i>.
2.  Loop For each <key_itemset> (except 1-itemsets)
            If Vi satisfies minimum support, move it to L1_DB, else write Vi into output file.
    End Loop For
3.  Generate 2-itemsets and do intersection to generate the data set
    Ui : <key_2-itemset_i, value_(list of transaction, count)_i>.
4.  Loop For each <key_2-itemset>
            If count>=min_support, move <Ui> to L1_DB, else write Ui into output file.
    End Loop For
Level 2 Map Phase:
Input Phase:
5.  k=2;
6.  Loop while FREQ_TAB has k-itemsets
          Read all k-itemsets from FREQ_TAB and divide them into equivalence classes as split files.
Map Phase:
7.  Read input as equivalence class.
8.  Generate k+1-itemsets and do intersection to generate the data set
    Xi : <key_k+1-itemset_i, value_(list of transaction, count)_i>.
9.  Loop For each <key_k+1-itemset>
        If Xi satisfies minimum support, store it in L1_DB
    End Loop For
Increment k
    End While loop
```

The input of the reduce function is the data obtained from the map function of each host. In reduce step, all 1-itemsets and all infrequent itemsets are reduced by summing up the count and the reduce function filters all frequent 1-itemsets and 2-itemsets alone and stores them in L1_DB. In both map and reduce functions, the frequent itemsets alone are moved to the global output file for further process. The usage of equivalence class in generating combinations of itemset gives immense advantage of reducing the number of combinations into one fifth. Considering the generation of three itemset combinations, instead of 36 combinations, it produces only 7. Formally, $\sum_{i=0}^{n} \binom{|Ui|}{2}$ number of combinations is considered instead of $\binom{|Lk-1|}{2}$ number of combinations. The concept of equivalence partitioning was used by Zaki, et al [27] to parallelize the candidate generation step in CCPD. It was also used in *Candidate Distribution by* Agrawal and Shafer, [1] to partition the candidates into disjoint sets.

V. Experimental results

The PDV algorithm is a parallel algorithm that can execute on n-nodes which includes execution of direct-vertical algorithm in each node. The algorithm should be evaluated for the measures speed and scaling. For experimental evaluation, the datasets namely, Sales transaction dataset has been used from UCI Machine learning repository[23]. Experiments were carried on many times to obtain stable values for each data point. The speedup measure is evaluated in single node and the performance is compared with DV algorithm. Table 1 and Figure 2 show the comparison of speed between the algorithms for various support values on the dataset in tabulation and graph respectively.

Table 1: Speedup Measure

Support(%) / Algorithm	DV algorithm	PDV algorithm
17	2.57	1.43
19	2.5	1.3
20	2.41	0.97
21	2.36	0.81
23	2.29	0.67

Clearly, the proposed algorithm speeds well as it executes parallel on data nodes. The proposed algorithm executes direct-vertical algorithm to generate some of the frequent k-itemsets in advance and only the infrequent itemsets are used to make candidate itemsets. Direct-vertical algorithm consumes negligible time on small partition of database. Due to this reason, the proposed algorithm outperforms other algorithms.

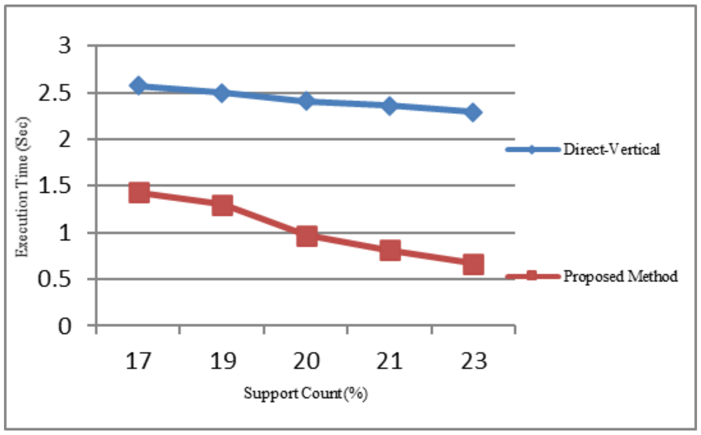

Fig. 2: Execution time comparison

The Scaleup measure denotes the ability of the algorithm against increase in both the number of clusters and the dataset size. Scaleup measure is taken for PDV algorithm to evaluate its efficiency. The execution time of PDV in different number of nodes is compared in Table 2. The corresponding graph analysis is shown in figure 3. It shows the scaleup efficiency of PDV algorithm. If the DV algorithm is not executed in each node, the time consumption of the proposed algorithm increases considerably.

Table 2: Scaleup Measure Of Pdv Algorithm

Support(%)/ Algorithm	PDV (2-Nodes)	PDV (4-Nodes)
17	0.57	0.36
19	0.52	0.32
20	0.40	0.27
21	0.31	0.21
23	0.22	0.12

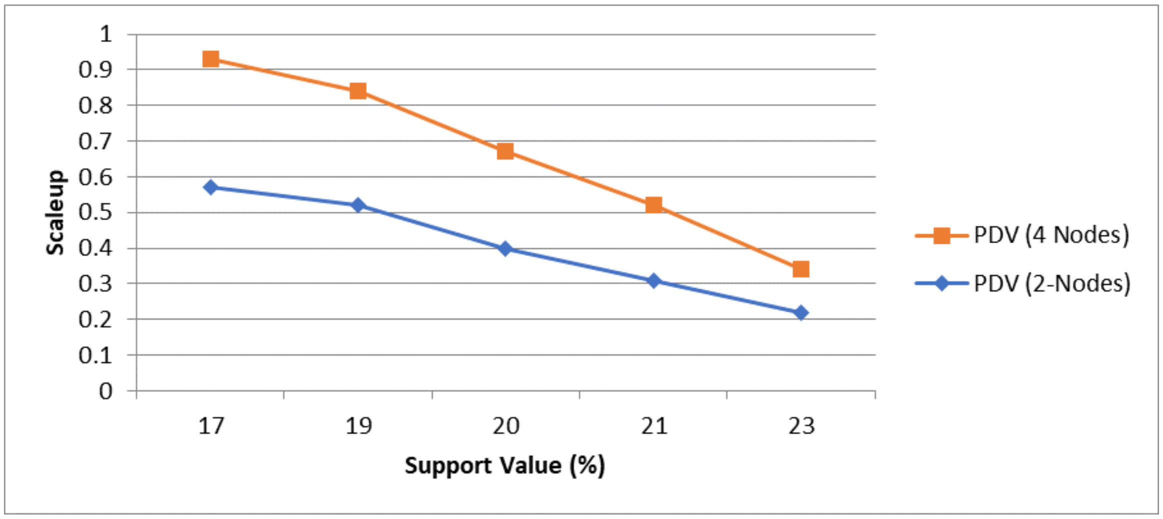

Fig. 3: Scaleup Performance of PDV algorithm

VI. Conclusions

Discovering frequent patterns in transactional databases is known as one of the most important data mining problems. This task requires a lot of computation power and memory. In this paper, it is proposed a new parallel algorithm which works on map reduce structure on Hadoop for mining frequent patterns. The algorithm uses direct-vertical algorithm and equivalence class concept for fast generation of frequent itemsets. The datasets are selected with the notion to measure speed and scaleup performance. The algorithm minimizes I/O overheads by scanning the local database portion only one time. Further, the algorithm uses simple intersection operation to compute frequent itemsets and doesn't have to maintain or search complex structures. The experimental results show that the proposed PDV algorithm outperforms than any other sequential algorithm and similar types of parallel algorithms. This assures that the algorithm can be implemented directly on general purpose habitual database systems as discussed by Holsheimer, et al. [5]. In future, scalability can be further improved by increasing number of nodes and the size of datasets.

REFERENCES

[1] Agrawal, R.; Shafer, J.: Parallel mining of association rules, *IEEE Trans. on Knowledge and Data Engineering.*, 8(6), 962–969 (1996).

[2] Agrawal Rakesh and Srikant Ramakrishnan (1994) Fast Algorithms for Mining Association Rules. Proceedings of twentieth International Conference on Very Large Data Bases, pp 487-499.

[3] Bo Wu, Defu Zhang, Qihua Lan, and Jiemin Zheng (Nov 2008) An Efficient Frequent Patterns Mining Algorithm based on Apriori Algorithm and the FP-Tree Structure. Third International Conference on Convergence and Hybrid Information Technology, Volume 1, pp 1099-1102.

[4] Goswami D.N., Chaturvedi Anshu, Raghuvanshi C.S (2010) An Algorithm for Frequent Pattern Mining Based on Apriori. International Journal on Computer Science and Engineering, Volume 02, No. 04, pp 942-947.

[5] Jaishree Singh, Hari Ram, Dr. J.S. Sodhi (Jan 2013) Improving Efficiency of Apriori Algorithm using Transaction Reduction. International journal of Scientific and Research Publications, Volume 3, Issue 1, pp 1-4.

[6] Jiawei Han, Micheline Kamber and Jian Pei (2012) Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers. USA.

[7] Jianwei Li; Ying Liu; Wei-keng Liao; Alok Choudhary:.Parallel data mining algorithms for association rules and clustering, CRC Press, (2006).

[8] Jnanamurthy HK, Vishesh HV, Vishruth Jain, Preetham Kumar, Radhika M. Pai (Jan 2013) Maximal Frequent Item sets using subset creation. International journal of Data Mining and Knowledge Management Process, Volume 3, No. 1, pp 27-38.

[9] Jongwook Woo.: Market basket analysis algorithm on Map/Reduce in AWSEC2, International Journal of Advanced Science and Technology, 46, (2012).

[10] Lin, J.; Dyer, C.: Data-Intensive Text Processing with MapReduce, Tutorial at the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics, (2010).

[11] Madhavi Vaidya.: Parallel processing of cluster by Map Reduce, International Journal of Distributed and Parallel Systems, 3(1), (2012).

[12] Manalisha Hazarika; Mirzanur Rahman.: Mapreduce based Eclat algorithm for Association rule mining in Data mining: MR_ECLAT, International journal of computer science and engineering, 3(1), 19-28 (2014).

[13] Mohammed J Zaki; Srinivasan Parthasarathy; Mitsunori Ogihara: Parallel Algorithms for discovery of Association rules, Data Mining and Knowledge Discovery, 1, 343–373 (1997).

[14] Ning Li; Li Zeng; Qing He; Zhongzhi Shi: Parallel Implementation of Apriori Algorithm Based on MapReduce, International Journal of Networked and Distributed Computing, 1(2), 89-96 (2013).

[15] Phongphun Kijsanayothin, Gantaphon Chalumporn and Rattikorn Hewett (November 2019) On using MapReduce to scale algorithms for Big Data analytics: a case study, Journal of Big Data, Volume 6, Article number: 105.

[16] Sandy Moens; Emin Aksehirli; Bart Goethals: Frequent Itemset Mining for Big Data, IEEE International Conference on Big Data- Workshop on Scalable Machine Learning: Theory and Applications, (2013).

[17] Shant Karakashian and Berthe Y. choueiry (November 2010) Tree-based algorithms for computing k-combinations and k-compositions. Constraint Systems Laboratory, Department of Computer and Engineering, University of Nebraska, Lincoln.

[18] Sheila A. Abaya (July 2012) Association rule mining based on Apriori algorithm in minimizing candidate generation. International Journal of Scientific and Engineering Research, Volume 3, Issue 7, pp 1-4.

[19] Sheng-Li Zhang (2012) A new mining algorithm of association rules and applications. Springer book on Bio-inspired computing and applications. Volume 6840, pp 123-128.

[20] Sunil Joshi, R.S. Jadon, R.C. Jain (Nov 2010) An Implementation of Frequent Pattern Mining Algorithm using Dynamic Function. International Journal on Computer Applications, Volume 9, No. 9, pp 37-41.

[21] Sunil Kumar S, Shyam Karanth S, Akshay K C, Ananth Prabhu, Bharathraj Kumar M (March 2012) Improved Apriori Algorithm based on bottom up approach using Probability and Matrix. International journal of Computer Science Issues, Volume 9, issue 2, No. 2, pp 242-246.

[22] Syed Khairuzzaman; Tanbeer Chowdhury; Farhan Ahmed; Byeong-Soo Jeong: Parallel and Distributed algorithms for

frequent pattern mining in large databases, IETE Technical Review, 26(1), 55-65 (2009).

[23] UCI Repository of Machine Learning Databases. Department of Information and Computer Science, University of California, Irvine, CA, USA, (1998).

[24] Vijayarani, S.; Sathya, P.: Mining Frequent item sets over Data Streams using Eclat Algorithm, International Conference on Research Trends in Computer Technology, 27-31 (2013).

[25] Xianfeng Yang; Liming Lian: A New Data Mining Algorithm based on MapReduce and Hadoop, International Journal of Signal Processing, Image Processing and Pattern Recognition, 7(2), 131-142 (2014).

[26] Yamuna devi N and Devi shree J (May 2014) A Combinatorial Tree Based Frequent Pattern Mining, Journal of Theoretical and Information Technology, Volume 63, No, 3, pp 781-789.

[27] Zaki, M.J.; Ogihara, M.; Parthasarathy, S.; Li, W.: Parallel data mining for association rules on shared-memory multi-processors, Technical report 618, University of Rochester, (1996).