International Journal of Electronics Communication and Computer Engineering
Volume 4, Issue (6) NCRTCST-2013, ISSN 2249–071X

National Conference on Recent Trends in Computer Science and Technology (NCRTCST)-2013

Effective Algo for the Frequent ItemSet in Data Mining

Madhuri Agrawal Gupta
Computer Science & Engineering Department
Geethanjali College of Engineering &Technology
Hyderabad, India
Email: madhuriagrawal2000@gmail.com

D. Praneeth
Computer Science & Engineering Department
Geethanjali College of Engineering &Technology
Hyderabad, India
Email: praneeth040@gmail.com

***Abstract* – A core data mining operation is Frequent itemset mining and has been extensively studied over last decade. This paper takes a new approach and implements a new algorithm for this problem and makes the two major contributions. First, it implements a one pass algorithm for frequent itemset mining which has the deterministic bound on the accuracy and does not require any out of core summary structure. Second, because this one pass algorithm does not produce any false negative it can be easily extended to a two pass accurate algorithm. The two pass algorithm is very memory efficient and allows the mining of datasets with large number of distinct items and/or very low support levels. The detailed experimental evaluation on datasets shows the following. First, this one pass algorithm very accurate in practice. Second, the algorithm requires significantly lower memory than the previous solutions given in [7]. The two pass algorithm outperforms apriori and FP-Tree when the number of distinct items is large and/or support levels are very low. In other cases it is quite competitive with the possible exception of cases where the average length of frequent item sets is quite high.**

***Keywords* – Itemset, Alphabet, DSMS, TreeHash, Support Level.**

I. INTRODUCTION

Algorithms for frequent itemset mining have typically been developed for datasets stored in the persistent storage and involved two or more passes over the datasets. Recently there has been much interest in the data arriving in the form of continuous and infinite data streams. In streaming environment a mining algorithm must take only a single pass over the data, such algorithm can only guarantee an approximate result.

Applications in which the data is modeled not as persistent relation but rather as transient *data streams*. Examples of such application include financial application, Network monitoring, Security, Telecommunication data management, Manufacturing, sensor network, and others. In the data-streaming model, Individual data items may be relational tuples, e.g. Network measurements, Call records, Web page visits, Sensor readings and so on .However their continuous arrival in multiple rapid time-varying, possibly unpredictable and unbounded streams appears to yield some fundamentally research problems.

In all the applications cited above it is not feasible simply load the arriving data into the traditional database management system (DBMS) and operate there on it. The traditional DBMS are not designed for rapid ad continuous loading of individual data items. And they do not directly supports the continuous queries that are typical of data stream application.[1] In this paper, we consider the fundamental models and issues in developing general purpose Data Stream Management Sysems (DSMS). Here, we are going to implement a new approach for frequent itemset mining. We present a single pass algorithm for frequent itemset mining in a streaming environment. This algorithm has the provable deterministic bound on accuracy. Unlike the only other work existing work in this area that we are familiar with, our new algorithm does not require any out-of-core summary structure. The key limitation on frequent itemset mining has been the high memory requirement when the numbers of distinct items are very large and or the support level required is very low. Our single pass algorithm has the property that it does not produce the false negatives i.e. all frequent itemset with the desired support level are reported. The false positive reported by this algorithm can be easily removed through a second pass on the dataset. This new two pass algorithm provides high memory efficiency while not compromising accuracy in any way.

II. DATA MINING

Data mining can be defined as a process of discovering new, interesting knowledge, such as patterns, associations, rules, changes, anomalies and significant structures from large amounts of data stored in data banks and other information repositories. It is currently regarded as the key element of a much more elaborate process called Knowledge Discovery in Databases (KDD). In general, a knowledge discovery process consists of an iterative sequence of the following steps (see *Figure 1*):

1. data selection, where data relevant to analysis task are retrieved from database;
2. Data cleaning, which handles noisy, erogenous, missing or irrelevant data.
3. Data integration (enrichment), where multiple heterogeneous data may be integrated into one;
4. Data transformation (coding), where data are transformed or consolidated into forms appropriate for different mining algorithms;
5. Data mining, which is an essential process where intelligent methods are applied in order to extract hidden and valuable knowledge from data;

6. Knowledge representation, where visualisation and knowledge representation techniques are used to present the mined knowledge to the user.

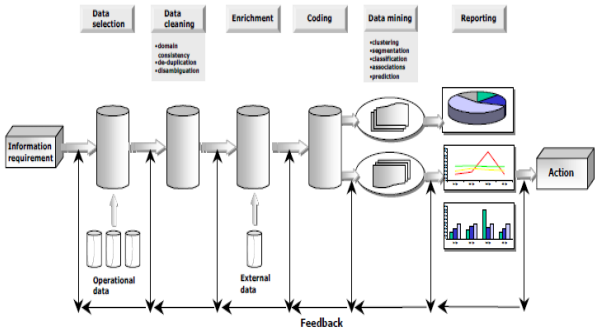

Figure 1. Position of Data Mining in the Knowledge Discovery process

Data mining is based on the results achieved in database systems, statistics, machine learning, statistical learning theory, chaos theory, pattern recognition, neural networks, probabilistic graph theory, fuzzy logic and genetic algorithms. A large set of data analysis methods have been developed in statistics over many years of studies. Machine learning and statistical learning theory have contributed significantly to classification and induction problems. Neural networks have shown their effectiveness in classification, prediction, and clustering analysis tasks. One can say that there is no one specific technique that characterizes data mining. Any technique that helps to extract more out of the data sets in an autonomous and intelligent way may be classified as a data mining technique. Therefore data mining techniques form a quite heterogeneous group.

III. DATA STREAM

The data stream concept has recently emerged in response to the continuous data problem. Algorithms for data streams can naturally cope with data sizes many times greater than memory and can extend to challenging real-time applications not previously tackled by machine learning or data mining. The assumption of data stream processing is that training examples can be briefly inspected a single time only, means they arrive in a high speed stream, then must be discarded to make room for subsequent examples. The algorithm has no control over the order of the examples seen, and must update its model incrementally as each example is inspected. An additional property, called anytime property, requires that the model is ready to be applied at any point between training examples. Measuring data stream classification performance is a three dimensional problem involving processing speed, memory and accuracy. It is not possible to enforce and simultaneously measure all three at the same time.

A classification algorithm must meet several requirements in order to work with the assumptions and be suitable for learning from data streams.[4]

Requirement 1: Process an example at a time, and inspect it only once (at most) –

The key characteristic of a data stream is that data 'flows' by one example after another. There is no allowance for random access of the data being supplied. Each example must be accepted as it arrives in the order that it arrives. Once inspected or ignored, an example is discarded with no ability to retrieve it again.

Requirement 2: Use a limited amount of memory -

The main motivation for employing the data stream model is that it allows processing of data that is many times larger than available working memory. The danger with processing such large amounts of data is that memory is easily exhausted if there is no intentional limit set on its use. Memory used by an algorithm can be divided into two categories: memory used to store running statistics, and memory used to store the current model.

Requirement 3: Work in a limited amount of time -

For an algorithm to scale comfortably to any number of examples, its runtime complexity must be linear in the number of examples. if an algorithm is to be capable of working in real-time, it must process the examples as fast if not faster than they arrive. Failure to do so inevitably means loss of data.

Requirement 4: Be ready to predict at any point -

An ideal algorithm should be capable of producing the best model it can from the data it has observed after seeing any number of examples.

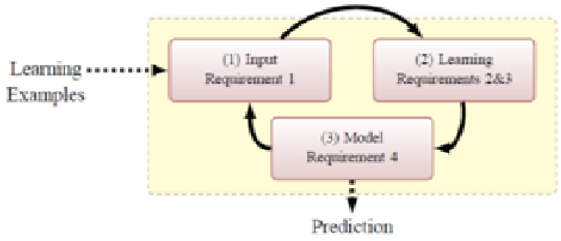

Fig.2. The Data Stream Classification Cycle

This illustrates the typical use of a data stream classification algorithm and how the requirements fit in a repeating cycle:

1. The algorithm is passed the next available example from the stream (Requirement 1).
2. The algorithm processes the example, updating its data structures. It does so without exceeding the memory bounds set on it (requirement 2), and as quickly as possible (Requirement 3).
3. The algorithm is ready to accept the next example. On request it is able to predict the class of unseen examples (Requirement 4).

IV. Frequent Item Set

Frequent sets play an essential role in many Data Mining tasks that try to find interesting patterns from databases, such as association rules, correlations, sequences, episodes, classifiers and clusters.

A set is called frequent if its support is no less than a given absolute minimal support threshold min_sup with 0>min_supabs>|D|. When working with frequencies of sets instead of their support, we use the relative minimal frequency threshold min_suprel, with 0>min_suprel>1. Obviously min_supabs = [min_suprel * |D|].[2]

Let D be a database of transactions over a set of items I, and min_sup is the minimal support threshold.

Together with the introduction of the frequent set mining problem. It uses the Apriori property to reduce the search space: All nonempty subsets of a frequent itemset must also be frequent. The apriori property is used in the Apriori algorithm. We can distinct two steps: join and prune.

Once the frequent itemsets from transactions in a database D have been found, it is straightforward to generate strong association rules from them, where strong association rules satisfy both minimum support and minimum confidence.

V. Improving the Efficiency of Apriori

1. Hash-based technique can be used to reduce the size of the candidate k-itemsets.
2. Transaction reduction – a transaction that does not contain any frequent k itemsets cannot contain any frequent k+1 itemsets.
3. Partitioning (partitioning the data to find candidate itemsets): A partitioning technique can be used that requires just two database scans to mine the frequent itemsets.
4. Sampling (mining on a subset of a given data): The basic idea of the sampling approach is to pick a random sample S of the given data D, and then search for frequent itemsets in S instead of D. In this way, we trade off some degree of accuracy against efficiency.
5. Dynamic itemset counting (we can add candidate itemsets at different points during the scan)

Mining the frequent itemset is an essential step in many data mining problems, such as mining the association rules, sequential patterns, closed patterns, maximal pattern, and many other important data mining tasks.[3] Recently the database and data mining communities have focused on new data models, where the data arrives in the form of the continuous streams. It is often referred to as *data streams* or *streaming data*. Many application generates large amount of data streams in real time, such as sensor data generated from sensor network, transaction flows in retail chains, web records and click streams in web applications, performance measurements in network monitoring and traffic management. call records in telecommunications, etc.

Hence the nature of the streaming data makes it essential to use online algorithm which requires only one scan over the data for knowledge discovery. Moreover it is not possible to store all the data in the main memory or even in the secondary storage.

VI. Memory Efficient Accurate Mining Algorithm

The problems explained above motivates the design for in-memory summary data structure with small memory footprints that can support both the one time and continuous queries. In other words, data streams mining algorithms have to sacrifice the correctness of its analysis result by allowing some *counting errors.[5]* Consequently, previous multiple pass data mining techniques studied for the traditional datasets can not be easily solved for the streaming data domain.

In this paper, we solve the problem of mining the frequent itemset in data stream. According to data stream processing model, [6] the research of mining frequent itemset in data streams can be divided into three fields:-

(i) Landmark window model
(ii) Sliding window model
(iii) Damped window model

In this paper, we implement a new approach for frequent itemset mining. Our work has two major contributions: -

In-core mining in streaming environment:

We present a single pass algorithm for frequent itemset mining in streaming environment. Our algorithm has provable deterministic bound on accuracy. Unlike the only other existing work in this area that we are familiar with, our new algorithm does not require any out-of-core summary structure. This is very desirable property since the stream mining algorithm need to be executed in small mobile devices which do not have attached disks for storing an out of core summary structure.

Memory efficient accurate mining:

Our two pass algorithm provides the high memory efficiency while not compromising accuracy in any way. Our work derives from recent work done by the Karp et. al. on determining the frequent items (or 1-itemsets).[2] They present a two pass algorithm for this purpose which requires only (1/θ) memory, where θ is the desired support level. Their first pass computes the superset of the superset of frequent items and the second pass eliminated all the false positives. Our work addresses three major challenges in applying their ideas for frequent itemset mining in a streaming environment. First, we have developed a method for finding frequent k-itemsets while still keeping the memory requirement limited. Second, we have developed a way to have a bound on the superset computed after a first pass. Third, we have developed a

new data structure and a number of other implementation optimizations to support efficient execution.

Our algorithm takes as input a parameter ε. Given the desired support level θ, our one pass algorithm reports all itemsets accruing with the frequency level θ and does not include any item with frequency level less than (1- ε) θ. In this process memory requirement increase proportional to 1/ε.

To efficiently implement this algorithm, we have designed a data structure *TreeHash*. This data structure implements a prefix tree using hash table. It has the compactness of prefix tree and allows easy deletion like hash table.[8]

VII. Application Domain

This Algorithms will be applicable in the applications which generates large amount of data streams in real time, Such as-

1. Sensor data generated from sensor networks.
2. Transaction flows in retail chains.
3. Web records and click streams in web applications
4. Performance monitoring and traffic management.
5. Call records in telecommunications. etc.

Also this algorithm will be applicable in firewall designing and executing online queries like measuring the network load at particular time.

VIII. Conclusion

This new algorithms is very accurate in practice, even when ε is 1, the accuracy is 94% or higher and in fact 100% in several cases.

Our algorithm is very memory efficient.

The algorithm can handle large number of distinct items and small support level using a reasonable amount of memory. For example with 100,000 distinct and support level of 0.05% could be handled with less than 200 MB main memory. A factor of 5 improvements over apriori.

By using this algorithm it is possible to compute more accurate result for online queries even if the less memory is available. So it is easier to use this algorithm for small hand held devised having the less amount of memory.[9]

References

[1] Ruoming Jin,Gagan Agrawal. An Algorithm for In-core frequent itemset miningon streaming Data

[2] Richard M.Karp, Christos H. papadimitrious, and scott shanker. A Simple algorithms for finding frequent elements in Streams and Bags.

[3] R.Agrawal and R. Shrikant Fast algorithm for mining association rules.In *proc 1994 Int. conf.* Very Large Databases(VLDB' 94) September 1994.

[4] B.Babcock S.Babu, M.Dattar, R.Motwani, J.Wisdom. Models and issues in Data Stream Systems. In *proceedings of the 2002 ACM Symposium on Principles of Database Systems (PODS 2002)* June-2002.

[5] E-H Han, G.Karypis and V. Kumar. Scalable parallel datamining for association rules. *IEEE transaction on Data and Knowledge Engineering* May-June 2000

[6] J.Han, J.Pei, and Y. Yin Mining frequent patterns without candidate generation. In proceeding of ACM SIGMOD Conference on Management of Data, 2000

[7] G.S. Manku and R. Motwani. Approximate frequency counts over Data Streams. In proceedings of Conference on Very Large Databases (VLDB), 2002

[8] Mohammed J. Zaki. Parallel and distributed association mining: A survey . IEEE Concurrency- 1999

[9] Z. Zheng, R.Kohavi, and L.Mason. Real World Performance of Association Rule Algorithm. In *Proceeding of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining* August-2001