Chapter 1

Introduction

Association rule mining is the process of uncovering the relationships or connections among a set of items. Association rule mining algorithms have been used in many diverse applications like Market Basket Analysis, Stock, Network, Neuroscience, etc. The development of new technologies has led to an exponential increase in the amount of data. A huge amount of data is generated in the digital world with applications such as social networks, stock markets, e-banking, economics, agriculture, education, medical [44], fraud detection [42], e-commerce, teaching[82], monitoring system [71]. Due to the increase in the size of the database, the importance of storing such valued information has become a necessity. Many businesses have adapted to digitisation generating gigantic amounts of data every nanosecond. There is a need to store the data for maintaining, processing, accessing, manipulating, and analyzing. A huge amount of storage is needed for storing such voluminous data [67].

Association rule mining algorithms are used to find all the interconnections or the relations of all the items in a dataset satisfying the constraints. The constraints are user-specified and it includes support and confidence metrics. The association rule mining problem is split into two subproblems. The first problem is to discover all the itemsets that satisfy a user-defined threshold in the dataset. These itemsets are called the frequent itemsets. The second one is to produce the rules from these frequent itemsets that satisfy the constraint, of minimum confidence. Such algorithms used for discovering and determining the patterns and trends are the Apriori Algorithm, FPTree algorithm, ECLAT algorithm, etc[1] [47].

1.1 Association Rule Mining (ARM)

Let T=$\{t_1, t_2, t_3..., t_n\}$ be the number of transactions where each transaction contains a set of itemsets I= $\{ i_1, i_2, i_3..., i_n\}$. Each itemset contains a group of items or elements. The association rule is a method of inferring the items in the form of $A \rightarrow B$, where A and $B \subseteq$ I. The itemset A, B contains a set of items. The item A is called the antecedent whereas the item B is called the consequent. The rule $A \rightarrow B$ means that in the same transaction if the items A occur then the items B also appear.

The user-defined constraints; support and confidence are required for the process of generating the association rules. The database is very large and users are often concerned with only those itemsets that are frequent. Thus, the user defines the threshold for support and confidence so that the uninteresting or irrelevant rules may be dropped. The patterns are extracted from the database as rules. A rule is considered interesting if it satisfies both the minimum threshold of support and confidence.

1.2 Background Literature

The problem of mining the association rule is quite intensive. Thus, it is necessary to implement efficient algorithms [38]. Algorithms such as SETM Algorithm use only simple database operations like merge-scan join and sorting [41]. Other fast algorithms used for mining association rules proposed include AIS algorithm [1], Apriori algorithm [2], DHP algorithm [62], OCD algorithm [73], Incremental Temporal Association Rules Mining (ITARM) Algorithm [31].
Association rule mining was introduced by Agarwal et al.,[1]. Various researchers tried to attempted to solve the problem of association rule mining in different ways. The approach focuses on the frequent itemset generated with candidate generation. The Apriori algorithm is used for the generation of frequent itemset with candidate generation. Depending on the value of the support threshold, the candidate itemset whose support count satisfies the constraint is considered as the frequent itemset. This process continues until there are no frequent itemsets left. The itemsets can be used for demonstrating the relationship of the items that are in the form of rules, IF ELSE based on condfidence[1]. In the paper[8], Bodon et. al uses a trie data structure for Apriori Algorithm. A trie is a hash tree that has a pointer that points to the node and the root is at depth 0. This trie uses array

and vector for implementation. The nodes in the hash tree are in sorted order. For the process of transversing, a linear search is used to find the searched item. The process of candidate generation becomes faster if the nodes are sorted since it is easier to perform the union operation and search for the itemset. Using a hash table, the searching technique is also improved. For computing a large dataset, an algorithm called Carma is introduced. The algorithm generates the large itemsets and provides the user the flexibility of choosing the support count only at the first scan. It outperforms the Apriori and DIC algorithms at lower thresholds [40].

GA-PPARM uses the algorithm Frequent Pattern method to mine the rules. In the first phase, the FP-Growth algorithm generates the association rules by using the metrics confidence. The generated association rules are then used for processing using the GA to produce the preserved privacy association rule keeping in view the sensitive rules [54]. IFTARMFGT is an algorithm proposed that combines the pros of the boolean matrix and incremental mining. In the beginning, the time series is transformed to discrete which includes the time intervals with fuzzy membership. Then, the boolean matrices concept was incorporated into the fuzzy membership for generating a fuzzy grid table to mine the itemsets. In the end, fuzzy temporal association rules are mined. The updation is done without scanning repeatedly the database and inherits the previous mining information [77].

An improved version of the previous Crow Search Algorithm (CSA) known as the NDCSA-CAR algorithm for mining the Class Association Rules from the dataset that is collected from gathered from the survey [39]. A fast algorithm was discovered for mining the association rules for temporal data based on the support of anti-monotonicity. The algorithm is divided into steps, discovering the frequent association rules and reliability of the rules. The algorithm used the two strategies, joining and pruning method [26].

An algorithm known as TM (Transaction Mapping) algorithm compresses the transaction *ids* of each itemset to transaction intervals [70]. The itemset are mapped and compressed to continuous transaction intervals using a tree. The intersection time is saved by this compression. However, this algorithm does not perform well as compared to FP growth. BIN - Tree is one of the data structures introduced by Rai et al. that represents the dataset in a compact form. Each transaction is stored as a node in the tree. This data structure is used to discover all the rare frequent itemsets in the main memory[65]. A novel algorithm was introduced for mining itemsets to solve the problem of scalability. The itemset tree compresses all the closed itemsets and a proposed search technique is used

to prune the tree. A fuzzy set-based frequent itemset mining algorithm (FSFIM) was introduced for mining the frequent itemsets and generation of the association rules expressed in the form of items. The itemsets in the dataset are categorized into three levels:- low, medium, and high depending on the item's quantity. Then, the item is classified into a category where each category is depicted as a fuzzy set. The number of generated patterns is less as compared to conventional methods [6]. SufRec is an algorithm that uses a suffix tree for the process of construction of a tree. The process of mining is divided into a series of tasks. The first part is called the SufRecDep algorithm which employs the outcome of the previous and the second part i.e., the SufRecInd algorithm which accomplishes and completes the task independently. The algorithm recursively works on updating the process of mining when new items are excluded or added [58].

A novel algorithm called the Weighted Binary Count Tree (WBIN) is implemented in CUDA to find the rules with the combination of frequent consequent and rare antecedent by using the parallel approach. However, the performance of the algorithm is low when the dataset with very large size or dimensions. The WBIN tree is a tree data structure that stores the data in memory. The node in the tree represents each transaction and every item is represented using a bit. However, the efficiency of the algorithm is limited by bounded by the dimension of the GPU during the process of mining [66]. A novel prediction model for generating itemsets based on the different objectives: non-defective class rules, defective class rules, and association relationships using features. During the process of mining the itemsets, the itemsets combine the selection of features to find the feature combination using mutual information. The correlation coefficient is further employed to portray the relation between the defect classes and the features [79].

1.3 With Candidate Generation

In association rule mining, there are two ways of generating frequent item sets: with candidate generation and without candidate generation. In candidate generation, the candidate itemsets are used for the generation of frequent itemsets. Some of the issues faced in the candidate generation phase are as follows:-

- **Consumption of memory space**

  When the database is large, a huge number of candidate itemsets are generated. Suppose 1-itemset generated by Apriori algorithm is 10^4 1-itemsets,

then it will generate 10^7 2-candidate itemsets. Furthermore, to produce a frequent pattern of itemset size 100, it will generate at most $2^{100} \approx 10^{30}$ candidates in total. Thus, storing a large number of candidate itemsets becomes an issue if memory is limited [35].

- **Time consumption**

  The performance of the algorithm depends on time consumption for running the algorithm. Association rule mining algorithm usually consumes a significant amount of time for its execution as it involves scanning the large database for computing the frequent itemsets[1].

- **Identifying the threshold value**

  When the database size increases, the threshold for support count needs to be adjusted. There is no appropriate index to define the threshold value. The main issue for setting the threshold is that it needs expert knowledge to set the parameter. The current mining techniques require the user to specify the parameters before executing the method. The user has to wait until the execution to adjust the parameters [72]. Sometimes, if the wrong threshold is given, the algorithm has to re-run again from the start. Since the algorithm cannot stop while running, this will take a long time for the algorithm to execute especially when the database is very large[23][1][72].

1.4 Without Candidate Generation Phase

Without candidate generation phase is a phase where the frequent itemsets are generated without producing the candidate itemsets. Some of the challenges faced during the generation of frequent itemsets without candidate itemsets generation.

- **Construction of tree**

  Creating an FP tree is unrealistic if the database size is very large. Traversing the FP tree is easy if the trees can fit in the main memory but it is impractical if it is not available in the memory as additional costs will be incurred for swapping operation [37].

- **Selection of support threshold**

  Choosing a good support threshold is dependent on the user's query. If 60% of the query has a support threshold greater than 50%, threshold=50 can be assigned. For the remaining 40%, a new FP will be constructed. This

creates a problem because every time FP tree will be constructed depending on the user's query.
Moreover, if the support count is too low, the FP Tree becomes infeasible. There is an overhead for pointers and links which increases the overhead of the FP tree. In such cases, the performance of FP Tree will be the same as Apriori [37].

- **Rule redundancy**
  If the support threshold is too low and the database size is large, many itemsets will be generated. From these itemsets, rules will be created. If there are millions of itemsets, then there will be redundancy in rule generation [37].

1.5 Problem Statement

The problem statement is to design a scheme that is concise and efficient for representing an itemset in the memory. If the memory for a single itemset can be minimized then the total memory needed for the remaining itemsets will also be minimized. If the representatin using bitmap can represent any set which is a subset of a universal set with the cardinality 400, only 400 bits are needed. These 38 bytes can be treated as a sequence of 50 bit longer integer array. These 400 bit long integer is treated as an array of 50 characters or 10 system given integers taking a total memory of 50 bytes. Therefore, to represent the set of 400 elements, linked list consume (4+4 * 400) 3200 bytes, array representation takes (4 * 400) 1600 bytes and bit representation will used 50 bytes. The rule mining algorithms attempts to find the frequent itemsets which is subset of the items in the dataset. Many set operations are utilised during the process of mining such as union, set difference, membership checking, intersection, etc. A 50 bytes long integer can be designed to handle the set operations, then representation will benefit the process of mining.

1.6 Motivation

Most of the existing work depends on support and experts for all situations to generate minimum possible itemsets [14], [23],[71], [80]. The threshold value for the support count has to be specified by the user in advance that sometimes require

some experts knowledge on certain domain. By using the proposed algorithm, it helps filtering the result depending on the user choice that may help the user to use association rule mining without too much detail of the domain. However, if we try to maintain all possible itemsets, huge amount of memory will be needed. out to be large. If the database consists of more than a million transactions, then storing itemsets becomes an issue because it requires more memory. Another issue of the association rule mining algorithm is mining the datasets without accessing the secondary memory. The dataset must fit into the main memory to enable frequent scanning of the dataset. This saves the repetitive loading of the itemsets from the disk and the I/O costs. The existing algorithms use either the linked list or array representation to represent the itemsets. While representing as a linked list, the space requirement may be 5 bytes for each element assuming that each element item is represented as a symbol but using this scheme, sets of more than 256 cannot be represented. Moreover, to represent one element, a total of 8 bytes will be used by the computer (though only 5 bytes are required by the system) due to word alignment. However, same 1600 bytes can be used to represent a set having 200 elements but from a universal set with cardinality 2^{32}-1, by simply assuming every element to be an integer. This representation may be useful for set with smaller cardinality. Array representation can be used only for finite set (universal set has fixed number of elements). If the universal set size ≥ 256 to represent any random set coming from this, a minimum 256 bytes is needed (1 byte for each element) but this representation will fail to represent sets with cardinality ≥ 256. To handle this situation, every element must be considered as an integer. In this case, domain size may be extended upto 2^{32}. If a set of maximum cardinality 300 is to be represented, then 1200 bytes is needed irrespective of cardinality of the set that is stored at this moment. This representation will use same amount of memory to represent the set irrespective of the elements present in the set.

1.7 Objective

Objective of the research

The objective of the thesis is to introduce a representation of the itemset that minimizes memory consumption. This representation of the itemset is incorporated into the Apriori Algorithm. To achieve the above-said objective, the following goals need to be fulfilled:-

- To define an itemset representation suitable for Association Rule Mining.

- Improving Apriori Algorithm by using a proposed representation of itemsets.
- To develop a graph-based algorithm for maintaining the frequent item sets.
- To generate rules from the frequent itemsets that are represented using the proposed itemset representation.

1.8 Organization of the Thesis

The thesis is organized as follows:

- Chapter 1 presents the research motivation in the thesis, the objective of the research, and the organization of the thesis.
- Chapter 2 provides the literature survey of the different set representation techniques, and generation of itemsets with candidate generation and without candidate generation.
- Chapter 3 presents the proposed representation for Apriori Algorithm
  - Incorporating the itemset representation for the Apriori
  - Incorporating the searching techniques for the itemsets in the Apriori Algorithm
- Chapter 4 presents the graph-based itemset algorithm. Motivation of the proposed algorithm, proposed graph-based algorithm, and evaluation of the proposed algorithm.
- Chapter 5 presents the different rule generation algorithms with the existing representation and the rule generation algorithms with the proposed representation.
- Chapter 6 concludes by summarizing the discoveries, illustrating the findings, concluding the thesis, and enunciating the future directions of the work.