Chapter – 2

REVIEW OF LITERATURE
CHAPTER 2

REVIEW OF LITERATURE

In this chapter, the research work related to data mining functionalities, classification and association mining is presented.

2.1 CLASSIFICATION

Classification is the basis to perform research in any field. There are many classification methods developed by various researchers which are classified into three groups viz., statistical methods, machine learning methods and neural net methods [2, 13, 16, 85]. Classification techniques are used in Business, Banking and Finance [8], Remote sensing, Packet classification in Wireless networking [28], Image classification [68], Speech recognition [89], Instruction detection [31], Text classification [69, 70, 77] and Digital information classification [55].

The process of classification begins by identifying one of the attributes of the tuples as the class label. The data set used to build the model is called training data set. The tuples in the training data has a class label in supervised learning method. Then the model is evaluated using test data whose class label is unknown. These models classify the test data and the result is compared to the class label of training data. If the performance of the model is good, it will be used to classify new tuples with unknown label.
Among the various methods, Decision tree is one of the most popular approaches used by various researchers [53, 59] from different disciplines such as statistics, machine learning, pattern recognition and data mining.

2.1.1 DECISION TREE INDUCERS

The decision tree consists of nodes that form a rooted tree, which means that it is a directed tree with a node called "root" which has no incoming edges [91] whereas other nodes have exactly one incoming edge. A node with outgoing edge is called an "internal" or test node and all other nodes are called "Leaves". Each leaf node is assigned to one class representing the target value. Decision tree consists of both nominal and numeric attributes. The tree complexity has a crucial effect on the accuracy performance and is measured by various metrics namely total number of nodes, total number of leaves, tree depth and number of attributes used [91]. As the dataset increases, the resulting decision tree will also increase rapidly, which makes it more difficult for people to understand the decision.

Decision trees for classification have been first developed by Hunt et al. [49], in the concept learning system (CLS). Based on this algorithm, Quinlan [84] proposed ID3 decision tree algorithm which is the starting point for induction of decision trees. This algorithm uses information gain as splitting criteria. The attribute with the highest information gain is selected as the splitting
attribute. The main drawback of ID3 is that the measure gain used tends to favor attributes with a large number of distinct values [21]. The algorithm C 4.5 [83] overcomes this drawback and is an extension of ID3 in which the gain ratio is the splitting criteria. Further, ID3 does not handle numeric attributes whereas the algorithm C 4.5 is capable to handle.

The accuracy of decision tree model design depends on the scale of training data set [25]. When the number of training data is too small, the necessary information may be missed and thus the model can not explain the classification rules of data. To handle large databases new algorithms are designed viz., SLIQ [71], SPRINT [94], PUBLIC [87] and Rainforest [35] which use new data structures to provide tree construction.

The algorithm SLIQ [71] is a decision tree classifier that uses a pre sorting technique in the tree-growth phase. Gini Index is used as a split measure. This sorting procedure is integrated with a breadth first tree to enable classification of disk-resident data sets. This algorithm does not require loading the entire dataset into main memory, instead it uses secondary memory. It creates a single decision tree from the entire dataset and provides upper limit for the largest dataset that can be processed as it uses a data structure that scales with the dataset size and this data structure is required to be resident in main memory all the time. SLIQ also uses a new-tree pruning algorithm based on Minimum Description Length (MDL) principle, which is efficient and produces compact
trees. Shafer et al. [94] have designed SPRINT algorithm that induces decision
trees relatively quickly and removes all the memory restrictions from decision
tree induction. SPRINT uses hash table proportional in size to the training set.
The hash table becomes expensive as the training set size grows in SPRINT.

Chandra et al. [22] proposed Elegant Decision Tree Algorithm (EDTA) to
improve the performance of SLIQ wherein Gini index is computed not for every
successive pair of values of an attribute like in SLIQ but over different ranges of
attribute values. The number of split points at which Gini is computed is less
compared to that of SLIQ and also the classification accuracy is better than that
of SLIQ. The number of split points evaluated in EDTA is \( n / k \) (where \( n \) is the
total number of different values the attribute can take and \( k \) is the total number of
intervals or group size). EDTA has been improved as “Proposed variation” in
which the Gini index is evaluated for each attribute at displacements of standard
deviation from the minimum value and or maximum value (in the direction of
decreasing value) of the attribute [21]. The other variations include evaluating
Gini at additional split point at a displacement of two standard deviations from
the minimum and maximum value.

Algorithm PUBLIC [87] is same as SPRINT except that entropy is used
for checking the goodness of a split. Pruning is performed with the tree building
and integrates the second pruning phase with the initial building phase. A node is
not expanded in PUBLIC during the building phase, if it is determined that it will
be pruned during the subsequent pruning phase. In order to make this determination for a node, before it is expanded, PUBLIC computes a lower bound on the minimum cost subtree rooted at the node. This estimate is then used by PUBLIC to identify the nodes that are certain to be pruned and for such nodes effort is not expended on splitting them.

Some of the various decision tree algorithms available in literature are given in Table 2.1. A review and comparison of the various algorithms has been made [59].

Table 2.1 Other Decision Tree Inducers

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAL5</td>
<td>designed specifically for numerical-valued attributes</td>
<td>[75]</td>
</tr>
<tr>
<td>LMDT</td>
<td>constructs a decision tree based on multivariate tests that are linear combination of the attributes</td>
<td>[18]</td>
</tr>
<tr>
<td>T1</td>
<td>a one level decision tree that classifies instances using only one attribute. Missing values are treated as a special value. It supports both continuous and nominal attributes</td>
<td>[47]</td>
</tr>
</tbody>
</table>

2.1.2 DECISION TREE ALGORITHMS FOR VARIOUS APPLICATIONS

The growth in the amount of data collected by information systems necessitates the need for decision trees that can handle large datasets. Recent research in this area is reviewed below.

Thabtah et al.[101] proposed multi-class multi-label classification algorithm which classify more than one unknown label that contain rules with multiple labels which integrate frequent itemset discovery and rule generation. Kwan Yi [55] has explained the library classification scheme to classify the text
which is a task of classifying documents to a predefined set of classes without human assistance.

Alhammady and Ramamohanrao [7] proposed a new algorithm to construct weighted decision trees. They used ‘emergency patterns’ method to find out the weights for the training data sets and compared the itemsets whose supports in one class are higher than in other classes. Weights are assigned to different training instances related to different classes and entropy measure is applied on weights of instances to build decision trees.

Yang et al. [112] proposed an algorithm for extracting actionable knowledge from decision trees. This algorithm is applied to customer relationship management (CRM) to find out the customers who are attritors and who are loyal. This method integrates data mining and decision making together. For improving customer relationship, the enterprise must know the actions to be taken to change customers from an undesired status to a desired one. This information is represented using decision trees.

Nguyen et al. [76] proposed a new algorithm for visualization of decision tree using a technique called T 2.5 D (2.5 dimensions tree). This model is compared with CARBO which is an interactive tree visualizer and their method shows the large tree in multiple views.
Violetta *et al.* [104] created own decision tree learning system—Generalization by Inductive Symbolic Method (GIMS)—which was applied to generate decision trees for classification from a banking domain. Czerwinski coefficient of attributes association is applied to generate a decision tree. This coefficient measures degree of dependence or independence which exits between two variables. The maximum value of the coefficient decides on the choice of the attributes to the following nodes of the decision tree.

Decision trees are also being used for image classification. Maree *et al.* [68] proposed a method for image classification based on decision tree. The method directly operates on pixel values and it combines a simple local sub-window extraction technique with the induction of ensembles of extremely randomized decision trees. The technique is applied to classify handwritten digits, faces, 3D objects and textures.

Lei Tu and Yao Chung [57] developed a new decision tree classification algorithm called "Intelligent decision tree" in which they designed a optimal decision tree which can be defined as the one with the minimum total external path length or the minimum total weighted external path length. It considers the global dependency structure among variables and looks ahead by selecting the good attributes which lead to a better classification decision tree.
Wang et al. [107] implemented the decision tree using hierarchical decomposition technique to deal with multi-class problem by developing multi-level decision trees. The idea of hierarchical decomposition is to transfer multi-class into two classes. At first, it classifies the training set into two groups i.e., 'Yes' or 'No' by choosing the most related attribute value. Using ID3 algorithm, hierarchical decomposition produces the decision tree of the first rank. Then it continues to classify 'Yes' or 'No' class in the first step into 'Yes' or 'No' subclass to produce the second rank decision tree. The method will go on iterating the above mentioned process until it classifies all the classes clearly. Lastly the hierarchical decomposition will turn the decision tree of each rank into a set of rules.

2.1.3 ADVANTAGES OF DECISION TREES [91]

- Decision trees are self-explanatory and they can be easily converted to a set of rules.
- Decision trees are capable to handle both nominal and numeric input attributes.
- Decision trees are capable of handling datasets that may have errors and missing values.

2.1.4 DISADVANTAGES

- Most of the algorithms (ID3 and C4.5) require the target attribute, which have only discrete values.
- As decision trees use “divide and conquer” method, they tend to perform well if a few highly relevant attributes exist, but not if many complex interactions are present.
There is a relation between classification and association. In classification, there is only one target to predict, i.e., the class, while association rule can predict any attribute in the data. A recent study on classification that integrates association named as associative classification has been proposed [53].

2.2 ASSOCIATION RULES

Association rule mining is introduced by Agarwal et al. [6] in 1993 and the frequent itemset mining has become a focused area of research. Most of the researchers have developed number of algorithms for mining all frequent itemsets, closed frequent itemsets and maximal frequent itemsets. Various search strategies have been developed, such as depth-first search vs breadth-first search, vertical formats (CHARM, MAFIA, TM) vs. horizontal formats (A-close, CLOSET, CLOSET+), tree-structure vs. other data structures, top down vs. bottom-up traversal, pseudo projection vs. physical projection of conditional database etc. Every algorithm tries to run faster by taking two things into the consideration viz., the number of times to scan the database and reduction in the size of the database in each scan. The selected frequent set discovery algorithms have been discussed in chronological order

AIS and SETM Algorithms: Candidate itemsets are generated and counted on the fly as the database is scanned in AIS [48]. After reading a transaction, it determines which of the itemsets that are found to be large in the
previous pass are contained in this transaction. New candidate itemsets are generated by extending these large itemsets with other items in the transaction. A large itemset ‘L’ is extended with only those items that are large and occur later in the lexicographic ordering of items than any of items in ‘L’. The candidates generated from a transaction are added to the set of candidate itemsets maintained for the pass, or the counts of the corresponding entries are increased if they are created by an earlier transaction. The SETM algorithm [48] is designed to use only standard SQL commands to find the frequent set. The drawback with AIS is that it generates too many candidates that later turn out to be small, causing wastage of effort. The Apriori algorithm performs better than AIS and SETM.

**Apriori algorithm** [2] is the most popular algorithm for finding frequent itemsets. It is an iterative level-wise approach, which uses frequent K-itemsets to generate frequent K+1 candidate itemsets with join and prune operations. Initially, the set of frequent 1-itemsets, L₁ is found. L₁ is used to find the set of frequent 2-itemsets L₂, which is used to find L₃ and so on. This algorithm terminates when no further frequent K-itemsets Lₖ is found. The OCD algorithm [66] uses the same closure property to eliminate candidates. The difference between OCD algorithm and Apriori is in the preliminary candidate generation. The algorithm OCD might produce a superset of the preliminary candidates produced in the join procedure of Apriori algorithm.
The AprioriTID [23] algorithm does not use the database for counting the support of candidate itemsets after the first pass. An encoding of the candidate itemsets used in the previous pass is employed for this purpose. The size of this encoding can become much smaller in later passes than the database, thus saving much reading effort.

The Partition algorithm [92] has been proposed to divide the database into equal sized partitions. Each partition is processed independently to produce a local frequent set for the partitions. After all local frequent sets are discovered, their union, the global candidate set, forms a superset of the actual frequent set. The database is then read again to produce the actual support for the global candidate set. The entire process takes only two passes.

The Sampling algorithm [102] has been proposed to consider first small samples of the database and discover an approximate frequent set by using a standard bottom-up approach algorithm. The approximate frequent set is then verified against the entire database. False frequent itemsets need to be removed and missing frequent itemsets need to be recovered.

The DHP algorithm [17] has extended the Apriori algorithm by introducing a hash filter for counting the upper-bound of the support of candidates in the next pass. Some candidates can be pruned before reading the database in the next pass. The drawback of DHP algorithm is that the hash table
is in competition for memory space with the hash tree used to hold the counts for the itemsets.

Gopalan and Sucahyo [36] proposed a compressed data representation method for mining frequent itemsets. A compressed prefix tree is used in this method to group the transactions and a method to compress the prefix tree for efficient use of memory is used. The drawback of this method is that the compressed transaction tree for huge databases will not fit into memory.

The two algorithms, Tree Projection and FP-Growth [3, 44] proposed new data structures to store compressed information about frequent itemsets. Tree projection uses a hierarchical lexicographic tree to project transactions at each node of the tree and uses a matrix counting technique on the projected transactions for counting support of candidate itemsets. The frequent pattern tree is a compressed data structure used in the FP-growth algorithm. FP-tree is constructed after two scans of the database and a divide and conquer method is used to generate the complete frequent itemsets.

Recently Baralis et al. [10] proposed Index based algorithm which is based on indexing support for mining in a RDBMS. It is named as I-tree index, which allows the extraction of frequent itemsets from relational database. The I-tree is a covering index, which encodes the complete data set. In I-tree index paths are partitioned into three layers based on the frequency in accessing nodes
namely ‘Top layer’ (for most frequently accessed items), ‘Middle layer’ (frequently accessed items) and ‘Bottom layer’ (rarely accessed frequent items). It is based on FP-tree but with indexing mechanism and its physical organization allows a reduced overhead in accessing the index blocks during the extraction task.

Grahne and Zhu [38] proposed a new algorithm for frequent itemset mining, using FP-trees in which FP-array technique is used to reduce the traverses in FP-tree which works well for sparse data sets. Three new methods for mining all (FPgrowth*), maximal (FPmax*) and closed frequent itemsets (FPclose) have been proposed. The drawback with these methods is in that they consume lots of memory when the data sets are very sparse.

To reduce the number of scans over the source database than Apriori, Hash-Mine algorithm [111] has been developed wherein the created hash tables derived from the original database are used for pruning candidate itemsets in some of the iterations. The method involves two database scans only. The first scan creates hash tables while the second one performs final pruning.

Song and Rajasekaran [98] proposed a Transaction Mapping (TM) algorithm using vertical database representation for frequent itemset mining. Transaction ids of each itemset are mapped in this method and compressed to continuous transaction intervals in a difference space and the counting of
itemsets is performed by intersecting these interval lists in a depth-first order along the lexicographic tree. When the compression coefficient becomes smaller than the average number of comparisons for interval intersection at a certain level, the algorithm switches to transaction id intersection.

2.2.1 MAXIMAL FREQUENT ITEMSET (MFI)

Maximal frequent itemsets have been introduced by Mannila et al. [67, 68]. ‘MAFIA’ [19] is one of the maximal frequent itemset algorithm which uses linked list representation to organize the all frequent itemsets. This algorithm stores the transactional database as a series of vertical bitmaps, where each bitmap represents an itemset in the database and a bit in each bitmap represents whether or not a given customer has the corresponding itemset.

In the algorithm Pincer-Search [60], maximum frequent set can be found using bottom-up and top-down searchers. Initially, the search starts in bottom-up, but a restricted search is also conducted in the top-down direction. This search is used only for maintaining and updating a new data structure, the maximum frequent candidate set (MFCS). It is used to prune early candidates that would be encountered in the bottom-up search. The key component of this algorithm is, use of information gathered in the search in one direction to prune more candidates during the search in the other direction. If some maximal frequent itemset is found in the top-down direction, then this itemset can be used to eliminate candidates in the bottom up direction. The subsets of this frequent
itemset can be pruned because they are frequent. If an infrequent itemset is found in the bottom-up direction, then it can be used to eliminate some candidates in the top-down direction.

Borgelt [15] introduced **Max Miner** algorithm to mine only long patterns. This algorithm partitions the candidate set into groups with the same prefix. Max-miner looks ahead at longest itemsets that can be constructed from every group. A frequency heuristic is used to reorder the items so that the most frequent items appear in the most candidate groups. It searches only for the maximal frequent itemsets, thus the search space can be reduced. This algorithm still needs several passes of the database to find the maximal frequent itemsets.

### 2.2.2 FREQUENT CLOSED ITEMSETS

Various algorithms are proposed for mining frequent closed itemsets including CHARM, LCM, CLOSET+ and AFOPT.

The algorithm Linear Time Closed Itemset Miner (LCM) [103] is based on prefix preserving closure extension, which is an extension from a closed itemset to another closed itemset, thereby completely enumerate closed itemsets without duplications. The time complexity of this algorithm is bounded by a linear function in the number of frequent closed itemsets. This algorithm is implemented with only arrays, so it is fast and outperforms other algorithms for some sparse datasets. The drawback is that, it is not good for dense datasets with
large minimum supports which involve many unnecessary items and transactions.

The algorithm CLOSET [81] deals with the FP-tree structure and some optimizations for reducing the search space are proposed. It is an extension of FP-growth algorithm which constructs a frequent pattern tree FP-tree and recursively builds conditional FP-tree in a bottom-up tree search method. CLOSET performance suffers in sparse datasets or when the support threshold is low. CLOSET has been modified to CLOSET+ [106] which uses data structures and data traversal strategies that depend on the characteristics of the dataset to be mined. They used top-down pseudo tree-projection and upward subset-checking for sparse datasets, whereas for dense datasets, the bottom-up physical tree-projection and a compressed result-tree have been adopted. The algorithm AFOPT [64] also uses FP-tree structure in which item search order, intermediate result representation and tree traversal strategy are considered dynamically, making the algorithm adaptive to general situations. In AFOPT, three different structures have been used to represent conditional databases viz., arrays for sparse conditional databases, AFOPT-tree for dense conditional databases and buckets for counting frequent itemsets, containing only top-k frequent items, where k is a parameter to control the number of buckets used. Several parameters are introduced to control when to use arrays or AFOPT-tree. It adopts the dynamic ascending frequency order. Later Ning Li et al. [78] proposed a
new technique for fast frequent closed itemset mining. The algorithm employs the concept of prefix-based equivalence class to partition the database in such a way that the possible frequent itemsets are uniformly distributed in the partitions. The advantage of this algorithm is that only frequent closed itemsets are generated and they are generated only once. It also avoids the closed-set examination in the entire mining processing. This method is suitable for parallel mining of frequent closed itemsets also.

An algorithm for finding Top K-Frequent Closed Itemset (TFP) [105] was developed recently. This algorithm mine the top-K frequent closed itemsets of minimum length $\text{min}_k$, where $k$ is a user desired number of frequent closed itemsets to be mined, top-k refers to the k most frequent closed itemsets and $\text{min}_1$ is the minimal length of each itemset without minimum support.

2.3 MULTILEVEL ASSOCIATION RULES

With the widespread of computerization in business, science and administration, enormous data are warehoused and it is necessary to study the data at multiple levels of abstraction spaces. With the recent development of data warehousing and OLAP technology, arranging data at multiple levels of abstraction has been a common practice. In market basket analysis, it is useful to analyze past transaction data to discover customer behavior, thereby the quality of business decision can be improved. The strategy of mining
association rules focus on discovering large itemsets which are groups of items appearing together in a sufficient number of transactions. There are several possible directions to explore efficient mining of multiple-level association rules. Han and Fu [40] developed a top-down progressive deepening technique based on Apriori algorithm for mining of multilevel association rules from large databases. The method uses an encoded transaction table, instead of the original transaction table and uses different support thresholds for different levels of abstraction. Based on the initial algorithm, three algorithms have been designed which are variants of the earlier. Rajkumar et al. [86] proposed a new algorithm 'AprioriNewMulti' in which minimum support will vary for different length of the itemset. The algorithm does not depend on the number of levels in concept hierarchy i.e., it does not scan the database for each level of abstraction for finding association rules.

2.3.1 MULTILEVEL ASSOCIATION ALGORITHMS FOR VARIOUS APPLICATIONS

Jane Yen [51] proposed a graph based approach to generate generalized multiple level association rules from a large database of customer transactions, which describes the associations among items in any concept level. This approach scans the database once to construct an association graph and then traverse the graph to generate large itemsets.
Sharma et al. [95] proposed a new approach of mining spatial data using multilevel association rules. The multiple level spatial mining methods are applied to extract interesting patterns in spatial and or non-spatial predicates. Data and spatial predicates are organized as set hierarchies to mine them level-by-level as required for multilevel spatial positive and negative association rules. A pruning strategy is used to reduce the search space in this algorithm.

The review of the reported literature on data mining pertaining to Classification and Association helps to identify the following limitations.

1. Most of the algorithms for classification of transaction database employ decision trees. For large databases, the use of decision trees approach needs more time and effort.

2. Most of the algorithms developed for finding frequent itemsets are based on Apriori algorithm that lead to several scans and generation of several candidate keys.

3. The algorithm reported in literature for finding multilevel association uses Apriori algorithm. It is necessary to generate frequent itemsets for all levels from 1 to N-1 for finding frequent K-itemset at level N.

In the present study, an attempt is made to propose algorithms that overcome the above limitations.