Chapter 2

Review

Today, association rule mining has been considered to be one of the important data mining techniques. It was introduced by Agarwal et al. in 1993. This is also referred to as market-basket problem because originally it was formulated for sales data. In simple term, association rule finds the measure of influence of one set of items on another set of items. The meaning of items varies from application to application. As for example, one association rule may be of the form “80% of the customers who buy bread also buy butter”. Here, the rule finds the influence of bread on butter. Association rules have got numerous applications such as decision support, telecommunication alarm diagnosis, prediction, catalogue design, add-on sales, store layout, customer segmentation based on buying pattern, etc.

The problem of association rule mining can be divided into two subproblems.

1. Find all the frequent (or large) itemsets in a given database.

2. Find the association rules using the large itemsets found in the first step.

Out of these two steps, the first step is important and difficult one. That’s why, most of the algorithms concentrate on finding the large itemsets from a large database in minimum possible time and using minimum resources. Once the large itemsets are known, finding association rules are straightforward. There are
many algorithms to find frequent itemsets. All these algorithms target different types of database.

Concept of data mining can be traced back to induction of classification approach [BFO+83, FWD93, HCC92]. The closest work in the machine learning literature is the \textit{KID3} algorithm presented in [Pia91]. Related work in the database literature is the work on inferring functional dependencies from data [Bit92, MR87]. Frequent itemset finding algorithms can be basically divided into two categories: for static databases and for dynamic databases. Next successive sections discuss some well known algorithms to find frequent itemsets in different domains.

2.1 Finding Frequent Itemsets in Static Databases

This section discusses some well known frequent itemsets finding algorithms for static databases: Static databases mean that databases do not grow. Its size, number of attributes remain static.

One of the earliest algorithms is \textit{SETM} [HS93], which was proposed by M Houtsma and A Swami. The algorithm uses SQL to compute large itemsets. It generates candidates on-the-fly based on transactions and remembers the TIDs for generating transactions with the candidate itemsets.

Another popular and robust frequent itemset finding algorithm is \textit{Apriori} [AMS+94], proposed by Agarwal et al. This algorithm is based on the fact that all the subsets of a large itemset are also large. The algorithm consists of multiple passes over the database. The first pass counts the number of occurrences of each item in the database to find the large 1-item sets. These 1-itemsets are used to generate the candidate 2-itemsets. Then, large 2-itemset are found by making a pass over the whole database. This process continues till there is at least one candidate itemset. This algorithm is simple and easy to implement. The algorithm is robust enough to find all the large itemsets in a database. The algorithm uses bottom-up and breadth-first approach. Another point to be observed is that number of database passes is equal to the length of the longest frequent itemset.
The main disadvantage of the algorithm is that it passes over the database several times which is responsible to increase the execution time for a large database. The same paper also proposed AprioriTid and Apriori Hybrid. AprioriTid is a little improvement over the Apriori. This algorithm uses TID, a unique number used to represent a transaction, and a different data structure. So, each transaction is identified by a TID. The main advantage of this algorithm is that the database is used to count the support of the candidate set only once - for the 1-itemsets. 2-itemsets onward, a different data structure is used to count the support of the itemsets. The size of this data structure gets reduced with the increase of number of iterations. Thus, it takes much less time than that of Apriori. Apriori Hybrid is just a combination of Apriori and AprioriTid. It uses Apriori in the initial passes and switches to AprioriTid for the remaining passes. Thus, it gets benefits from both the algorithms and can be found to be better than the other two in terms of execution time.

Another important algorithm is Pincer-Search [LK98]. This algorithm uses bi-directional approach i.e. top-down and bottom-up. It finds frequent itemsets in bottom-up manner and at the same time it maintains a list of maximal frequent itemsets. Maximum benefit is obtained when maximum frequent itemsets is found in the very early passes of the algorithm.

Park et al. proposed DHP (Direct Hashing and Pruning) [PCY95a]. It has two major features such as efficient generation of large itemsets using hashing technique and effective reduction on transaction database size. DHP is useful for generation of candidate large itemsets, particularly large 2-itemsets. However, this algorithm does not work properly for dense databases.

Some important algorithms can be found in [Zak00, ZPO+97, ZH99]. Among them, CHARM [ZH99] is an important algorithm. The algorithm introduced the concept of closed frequent itemsets, which is much smaller than the set of all frequent itemsets. With this concept, it is not necessary to generate all possible frequent itemsets and rules. The paper has shown that any rule is equivalent to some rules between closed frequent itemsets, resulting in reduction in redundant frequent itemsets and association rules.

Most of the algorithms mentioned above and other algorithms of its kind generate candidate sets and pass over the whole database to count the support of the
candidate sets. Generating the candidate sets and repeated passing over the database is a time consuming and tedious task. Moreover, it takes lot of space in the memory to store the candidate sets. To overcome these problems there are some algorithms which find the frequent itemsets without generating candidate sets. One such algorithm is \textit{FP-growth} algorithm \cite{Pujol, HPY00}. The algorithm consists of two phases. In the first phase, it constructs the FP-tree with respect to a given minimum support and in the second phase, it finds the frequent itemsets from the FP-tree. The algorithm first makes one pass over the database to find the frequent 1-itemsets. Then it removes the non-frequent items from the transactions and rearrange the items in the transactions in the descending order of their frequency. Then the algorithm makes one pass over the whole database to construct the FP-tree. In order to find the frequency of different combinations, the algorithm computes the conditional FP-tree. Obviously the \textit{FP-growth} algorithm has the advantage of not having to generate the candidate sets. The algorithm finds all the frequent itemsets and works very fast. However, this algorithm also has shortcomings \cite{Pujol, Bor} such as i) it takes lot of time to construct the FP-tree for high dimensional dense large databases ii) its performance degrades with increase of minimum support.

There have been some attempts to develop frequent itemsets finding algorithms using bitmap techniques \cite{AD95, Gra94, JD099, Joh98, MZ98, NG95}. The latest one being the \textit{BiLAssocRule} \cite{HLL03}. This algorithm uses bitmaps of the items and applies the basic bit operations like AND, OR, etc. to find the support of the candidates. So, the algorithm does not require to scan the database more than once and works much faster than the other algorithms mentioned above.

Other approaches such as sequential patterns \cite{AS95}, generalized association rules \cite{SA95}, multilevel association rules \cite{HF95}, quantitative association \cite{SA96} rules are worth mentioning.

\textit{Partitional, parallel and distributed} methods also have been studied to find the frequent itemsets. In the partitioning approach \cite{SON95}, database is partitioned and the rule mining is carried out for each partition. Finally, frequent itemsets computed for each partition are merged to generate the frequent itemsets for the whole database. The main shortcoming of the algorithm is the choosing of number of partitions. Two aspects are taken into consideration while choosing
the number of partitions—available buffer space and available memory. In this approach, the number of candidate sets and execution time are reduced to a great extent. Further, it provides scope for parallelization of the rule mining task.

2.1.1 Distributed and Parallel Algorithms

There have been some works on parallel and distributed algorithms. Main motivations behind parallel and distributed algorithms are as follows:

- Mining databases containing huge amount of data needs more processing power.
- Most of the databases are distributed in nature.
- The algorithms also can be used in centralized databases by partitioning the database and placing the portions in different sites.

[AMS+94] proposed two parallel versions of Apriori called Count Distribution (CD) and Data Distribution (DD). CD algorithm scales linearly and speedup of the algorithm is also good with respect to the number of transactions. The drawback of the algorithm is that it does not parallelize building of the hash tree. DD algorithm partitions the candidate sets and assigns each partition to a processor. However, the algorithm takes maximum time in data movement among the processors and most of the time processors remain idle due to poor interaction scheme among the processors. Parallel version of DHP algorithm, called PDM, was proposed in [PCY95b]. The main disadvantage of the algorithm is that $O(n^2)$ messages are required for support count exchange for each candidate set. Cheung et al. [CHN+96a] proposed one efficient distributed algorithm called Fast Distributed Mining of association rules (FDM). The algorithm is advantageous due to following reasons.

- It uses some relationships between locally large and globally large itemsets to reduce the candidate sets and in turn number of messages to be passed is reduced.
It uses local and global pruning techniques to prune away the candidates in the sites.

It requires only $O(n)$ messages for support count exchange, where $n$ is the number of sites.

[CHN+96a] introduced three versions of FDM i.e. FDM-LP, FDM-LUP and FDM-LPP. In [CNF+96], distributed version is proposed called DMA (Distributed Mining Association Rules). This algorithm also needs $O(n^2)$ messages for support count for each candidate set, where $n$ is the number of sites.

In [HKK00], two new parallel algorithms called Intelligent Data Distribution (IDD) and Hybrid Distribution (HD) can be found. IDD is improvement over DD. It reduces communication time and processor idle time. HD combines the advantages of CD and IDD. It groups the processors and partitions the candidate sets to maintain load balance.

### 2.1.2 Multilevel Association Rules Mining

In many real-life scenario, data items exist in the hierarchy of concept level. So, it is difficult to find strong association rules among data items at low levels of abstraction due to the sparsity of data in multi-dimensional space. Association rules at high concept level generally represent a common pattern. As for example, "bread and butter are bought together" may not be an interesting pattern, but "honey, bread and butter are bought together" may be an interesting pattern. Therefore, data mining systems should provide capabilities to mine association rules at multiple levels of abstraction (concept hierarchies) and traverse easily among different abstraction spaces. Concept hierarchies may be specified by the users familiar with the data or may be specified implicitly in the data itself. As for example, there may not exist any association rule between IBM laptop computer and Philips b/w printer, but there may exist one association rule between IBM computer and Philips printer. Rules generated from association rule mining with concept hierarchies are called multiple-level or multi-level association rules. [HF95, SA95] have discussed some issues of multiple-level association rule mining.
Different approaches may be used for multilevel association rule mining. In general a top-down approach is used. To find frequent itemsets at each level, general algorithms like Apriori can be used. One main difficulty in multilevel association rule mining with reduced support is applying the search strategy thorough the concept hierarchy. Some of the widely used search strategies are Level-by-level independent, Level-cross filtering by single item, Level-cross filtering by k-itemsets, etc. As far as algorithms are concerned, some algorithms to find association rules in multi-level databases can be found in [HF95, SA95].

2.1.3 Multidimensional Association Rule Mining

Multidimensional association rule mining refers to the mining of rules involving more than one predicate or dimension. One rule “bread ⇒ butter” can be written as “buys(X, bread) ⇒ buys(X, butter)”. Here, the rule consists of only one predicate buys. So, this is an example of single-dimensional or intra-dimensional association rule. In reality, the databases and warehouses store many other related information in addition to only transactional information. As for example, one database may store the sales transactions of a supermarket along with the customers' age, address, income, occupation, etc. So, in this case, it may be interesting to find the association rules containing more than one predicate/dimension such as age(X, "20...25") \land occupation (X, researcher) ⇒ buys(X, laptop). This association rule contains three predicates - age, occupation and buys. This kind of multidimensional association rule without any repetition of predicates is called inter-dimensional association rule, otherwise it is called intra-dimensional association rule.

Techniques for mining multidimensional association rules are categorized depending on the treatment of the quantitative attributes.

- The first category is called multidimensional association rules using static discretization of quantitative attributes, where a predefined concept hierarchy is used to replace the original numeric values of the quantitative attribute.

- In the second approach, quantitative attributes are discretized into beans based on the distribution of the data. These beans may be further merged
CHAPTER 2. REVIEW

28

during mining process. This process is dynamic and established so as to satisfy some mining criteria such as the maximizing the confidence of the rules mined. This method is also called quantitative association rules.

- The third approach discretize the quantitative attributes to capture the semantic meaning of the interval data. It considers the distance between two points. So, it is also called distance-based association rules.

Mining Multidimensional Association Rules Using Static Discretization of Quantitative Attributes

[KHC97] has given the detail account of the discretization of quantitative attributes and data cubes. In this technique, quantitative attributes are discretized prior to mining using predicate concept hierarchies and categorical attributes may also be generalized to the higher conceptual level. Classical algorithm like Apriori may be modified to find the frequent predicate sets instead of frequent itemsets. Other techniques such as sampling, hashing, partitioning may also be applied. Data cubes will be suitable for mining multidimensional association rule mining. Data cubes are the lattice of cuboids which are multidimensional. If the warehouse under study already contains some data cube, then it can be used to find the frequent predicates. Otherwise, some data cube will be required to be created.

Quantitative Association Rules Mining

This is the type of association rules in which the numeric attributes are dynamically discretized during mining process to satisfy some mining criteria. A quantitative association rule with \( n \) quantitative attributes in the antecedent is called \( n \)-dimensional quantitative association rule. As for example, the association rule $\text{income}(X, "10000...30000") \land \text{age}(X, "20...30") \Rightarrow \text{buys}(X, "\text{high resolution monitor}")$ is a 2-D quantitative association rules.

Reference [LCK98] has given an approach to find the quantitative association rules called ARCS (Association Rule Clustering System). This approach finds the association rules for two quantitative attributes - one in the antecedent and one
categorical attribute in the consequent. In this approach, the pair of quantitative attributes are mapped onto a 2-D grid for tuples satisfying a given categorical attribute condition. Then the grid is searched for clusters of points from which the association rules are generated.

Srikant and Agarwal [SA96] proposed a non-grid based technique for mining quantitative association rules which uses a measure of partial completeness. Other techniques such as mining quantitative rules based on rectilinear regions was proposed by Fukuda et al. [FMM+96] and Yoda et al. [YFM+97].

**Distance-Based Association Rules Mining**

If the quantitative attributes are discretized with the previous two methods, it may not capture the semantics of intervals since they do not consider relative distance between the points or intervals. As for example, equidepth partition may find interval 20000..50000, which is quite wide. The distance based partitioning seems to be most intuitive, since it groups values that are close together within the same interval. So, the distance based intervals produce more meaningful discretization. Intervals for each quantitative attribute can be established by clustering the values for the attributes. Another advantage of the distance-based association rule is the that it gives the scope of closeness or approximation in the predicates. As for example, in the association rule "is(X, cosmetics) \& make(X, foreign) ⇒ price(X, 300)", the price predicate is fixed at 300. The distance-based association rules allows the scope to give the range of values instead of a fixed value such as this.

The algorithms for distance-based association rule mining can employ two phase technique. The first phase finds the intervals or clusters using some clustering algorithm and the second phase finds the distance-based association rule by searching for groups of clusters that occur frequently together. [MY97] has proposed an approach to find the distance-based association rules by employing the above two phase techniques.
2.1.4 Spatial Association Rule Mining

With the wide application of remote sensing technology and automatic data collection tools, huge amount of spatial data have been collected in the large spatial databases. In other words, a spatial database stores a large amount of space-related data, such as maps, pre-processed remote sensing or medical imaging data. The extraction of the knowledge discovery in the large spatial pose great challenges to the currently available spatial database technologies. Spatial databases have many features that distinguishes them from relational databases. They carry the topological or distance information, usually organized by sophisticated, multidimensional spatial indexing structures that are accessed by spatial data access method and often require spatial reasoning, geometric computation and spatial knowledge representation techniques. Spatial data mining refers to the extraction of implicit knowledge, spatial relations or other patterns not explicitly stored in the spatial databases [KAH96].

Spatial data mining can be categorized based on the kinds of rules to be discovered in spatial databases. A spatial characteristic rule is a general description of a set of spatial-related data. For example, the description of a general weather pattern in set of geographic regions is a spatial characteristic rule. A spatial discriminant rule is the general description of the contrasting or discriminating features of class of spatial-related data from other classes. For example, the comparison of weather patterns in two geographic regions is spatial discriminant rule. There have been some interesting studies in spatial characteristic rules and spatial discriminant rules [NH94].

Statistical spatial analysis tools have been used extensively for analyzing spatial data [FR94]. Statistical tools are good for numerical data, but statistical techniques usually require the assumptions regarding to statistical independence and of spatially distributed data. Such assumptions do not apply in the real world situation because spatial objects are often influenced by the neighboring objects/regions. Again, predicate rules cannot be described using standard methods of statistical spatial analysis. It requires a lot of domain and statistical knowledge. So, only the persons who are experts in statistics can handle it. These arguments suggest that statistical techniques alone can not be used for spatial data mining. Another major approach in data mining is to apply gener-
alization techniques to spatial and non-spatial data to generalize detailed spatial data to certain level and study the general characteristics and data distribution at this level.

Although the concept of spatial association rule is same as that of association rules in a relational database, the definitions are required to be redefined to meet the requirement of spatial association rules. As for example, a spatial association rule may look like

\[
is_a(A, \text{large\_town}), \text{intersects}(A, B), \text{adjacent\_to}(A, C) \rightarrow is_a(B, \text{motorway}),
\]

\[
is_a(C, \text{sea}). \ (30\%, 80\%).
\]

This rule states that "30% of large towns intersects a motorway and are adjacent to the sea". This rule also states that "If a large towns intersects a spatial object B and is adjacent to C then B is a motorway and C is a sea in 80% cases".

Mining spatial association rules is a more complex task than mining transactional association rules. The degree of complexity are due to the implicit definition of association relations and the granularity of spatial objects. The spatial relations may be topological [Ege91] such as intersect, overlap, disjoint; distance such as close\_to, far\_way, etc. and direction such as left, right, etc. Therefore, complex data transformation processes are required to make spatial relations explicit.

Reference [KH95] has proposed a new algorithm for mining association rules in Geographic Information Databases. The algorithm specifies an SQL-like spatial data mining query interface, which is based on Spatial-SQL [EH94], for an experimental spatial data mining system protocol GeoMiner. Many variations of the above algorithm can be explored to enhance the power and performance of spatial association rule mining.

ILP methods also have been extensively used in spatial data mining. Most of the mining algorithm requires the reduction of multi-relational database to the single format. The strength of ILP method is the common background with deductive relational database (DDB) which can be exploited to implement the notion of inductive database [Man97] as pointed out by Flach [Fla98]. In recent times, a database(DB) approach to multi-relational data mining has been presented [KBJ+99]. It explodes the semantic information in the database schema to prune the search space and define the database primitives to ensure efficiency.
Some more works can be found in [Pop98]. The paper has presented a general purpose ILP system: *INGENS* [MEL+00], which is an inductive graphic information system with learning capabilities that currently support the classification task. There is another ILP system called *SPADA* (Spatial Pattern Discovery Algorithm) [ML01] which operates on DDB set up by an initial step of feature extraction from a spatial database. The basic idea in this ILP approach is that a spatial database can be boiled down to a DDB once that reference objects and task-relevant objects, their spatial properties and the spatial relationship among them have been extracted according to predefined semantics. As for topological relations, the algorithm has adopted the 9-intersection [EH94] model. The *SPADA* can tackle applications which cannot be handled by either Geo-Associator [HKS97] or WARMR [DT99].

### 2.1.5 Constraint-based Association Rule Mining

As the name suggests, constraint-based association rule mining allows [SMO+94] users to specify some constraints. Thus, association rules become more useful and interesting to the users. A simple way is to find all the association rules and then filter out the rules which do not satisfy the users’ constraints. However, it may generate a lot of redundant rules. So, it is required to incorporate the constraints into the steps of rule generations. Constraints may be of different types.

1. **Knowledge type constraints**: This refers to the type of knowledge to be mined such as association rules.
2. **Data constraints**: This specify set of task-relevant data.
3. Dimension/level constraint: This refers to the number of dimension and levels of concept hierarchy to be used.
4. **Interestingness constraints**: This refers to the interestingness measurement such as support, confidence, etc.
5. **Rule constraints**: This refer to the form of rules to be mined. As for example, user may specify number of predicates in the antecedent and consequents of the rules, attributes values, aggregate values, etc.
Ng. et al. [NLH+98] carried out some work on constraint-based association rule mining. They proposed \textit{CAP} algorithm for constraint-based association rule mining. Some works on meta-rule guided constraint-based mining can be found in [KHC97]. Meta-rules are generally based on users' experience, expectation, intuition, etc. Again, rule constraints can be classified into five categories with respect to frequent itemset mining.

1. \textit{Anti-monotone}: These constraints are generally applied to iterative algorithms like \textit{Apriori} so that number of iterations are reduced and at the same time preserves the completeness. One example may be “\text{sum}(\text{price}) \leq 500". So, any itemsets whose total price is greater than Rs. 500 can be rejected because all of its supersets will be having price more than Rs. 500.

2. \textit{Monotone}: These constraints are opposite to anti-monotone. One example is “\text{sum}(\text{price}) > 500". Here, superset of any itemset, which satisfies this constraint, will also satisfy the constraint. So, it is not required to check the constraint for the supersets.

3. \textit{Succinct}: By this constraint, one can find all the itemsets which are guaranteed to satisfy the constraint. One example may be “\text{max}(\text{price}) \geq 500". This constraint can be tested before the support count starts, which in turn reduces the execution time.

4. \textit{Convertible}: These constraints can be converted to anti-monotone or monotone by rearranging the items in the itemset and transactions. One example may be “\text{avg}(\text{prices}) \leq 1000". Here, if the items in a transaction are added to an itemset in the ascending order of process, superset of an itemset, which violates this constraint, will also violate the constraint.

5. \textit{Inconvertible}: These constraints are tough and cannot be converted to previous constraints. One example may be “\text{sum}(\text{itemset}) \leq 1000 and value of each item in the itemsets is any real number”.

Some useful concepts of the predicate constraints can be found in [AK93, LHC97, SVA97]. [AMS+94] also gives an efficient method for mining constrained correlated sets.
2.2 Finding Frequent Itemset in Dynamic Databases

One general assumption in all the above algorithms is that database is static. However, in practice, no database is static. The itemsets which are frequent may not be frequent when the database is updated and the itemsets which are not frequent may become frequent when the database is updated. So, some algorithms are required to update the set of frequent itemsets when the database is updated. Moreover, new database may contain some new interesting rules which were not present in the old database.

One obvious technique to find frequent itemsets in dynamic databases is re-running the algorithms for the updated database, which is not desirable. The main thrust is to use the already existing frequent itemsets. [CHN+96b] has given \textit{FUP} (Fast Update Algorithm). This algorithm has been found to be superior to re-running of the \textit{Apriori} algorithm over the updated database by a factor of 2 to 16. The algorithm works in the similar way as the \textit{Apriori}. It also generates the candidate-sets based-on the large itemsets in the previous pass. Following are the main features of the algorithm, which distinguishes it from \textit{Apriori}:

- In each iteration, the support of large itemsets are updated against the incremental database to filter out itemsets that are no longer large in the updated database. Only the incremental database is scanned to do the filtering.

- While scanning the increment, a set of candidate sets is extracted from the transactions in the incremental database, together with their supports. The support of these itemsets are then updated against the the old database to find the new large itemsets.

- Many itemsets are pruned by a simple check on their supports in the incremental database.

- the size of the updated database is reduced at each iteration by pruning some items from some transactions in the updated database.
Another version of \textit{FUP} is \textit{FUP}2 \cite{CLK97}, which addresses the maintenance problem for association rule mining. The algorithm has taken care of the deletion of transactions also. The algorithm works like \textit{Apriori}. The difference is that it divides the candidate itemsets into two subsets - one subsets keeps the candidate sets which were large in the old database and the other keeps the new candidate sets. The algorithm scans the old database, if there are some new itemsets in the updated database.

Thomas et al. \cite{TBA+97} has discussed one very efficient algorithm for the incremental updating of association rules. This algorithm has used the concept of negative border sets. The negative border consists of all itemsets that were candidates of level-wise method which did not have enough support i.e. an itemset which is not large, but all its subsets are large. This algorithm has been found to be superior to the \textit{FUP} algorithm both in terms of execution time and number of candidate generation.

Incremental algorithms were also considered in \cite{FAA+97}. The algorithm \textit{DELI} \cite{LCK98} has used a sampling technique to find the amount of changes of new association rules. It has used the concept of upper and lower bound to determine if the maintenance is required or not.

Feldman et al. \cite{FAL+99, PujOl} proposed one very efficient algorithm which uses the concept of \textit{border set} and \textit{promoted border set}. The algorithm is called \textit{Borders} algorithm. An itemset \(X\) is called a border set if \(X\) is not frequent, but all its subsets are frequent. An itemset that was a border set before update and has become frequent set after update is called a \textit{promoted border set}. The \textit{Borders} algorithm maintains support counts for all the frequent sets as well as for all the border sets. The main advantage of the algorithm is that it uses the existing frequent itemsets to find the new frequent itemsets in the updated database. However, the disadvantage of the algorithm is that it has to scan the whole database frequently if there is even one promoted border set.

The algorithm \textit{MAAP} \cite{ZE01} also efficiently generates the incremental association rules in the updated database by applying the \textit{Apriori} property. The algorithm first computes the high level large itemsets. Then it starts by generating all lower level large itemsets. This algorithm takes care of the small itemsets in the old database also. Incremental algorithms were also considered in \cite{FAA+97}.
Another algorithm can be found in [ES02], which has used *FP-tree* to update the association rules in an incremental database.

### 2.3 Interestingness of Association Rules

It is obvious that all the strong rules are not interesting. To support this idea some work on quantifying the *usefulness* and *interestingness* of the generated rules can be found in [PM94]. Several metrics such as confidence and support [AlS93], variance and chi-squared value [NM, Mor98], gain [FMM+96], entropy gain [MFM+98], gini [MFM+98], laplace [CB91, Web95], conviction [BMU+97], etc. are used to measure interestingness of a rule. There are several algorithms that efficiently find best rule according to some of these metrics [FMM+96, NM, RS02, BA99]. Among them, the technique given in [BA99] is worth mentioning. [BA99] has defined an optimized rule mining problem using partial order. It has also shown that solving the optimized problem with respect to a particular partial order is guaranteed to identify most interesting rule according to other interesting metrics mentioned above. Ultimately, it is the users who decide if a rule is interesting or not.

Sometimes, the rules of the form $X \Rightarrow Y$ may be misleading because algorithms may find strong rules among the items/attributes which are negatively correlated. So, some alternative framework is required to measure the interestingness of a rule. [BMS97] has given one such framework of correlation between the itemsets, which can be used to find if a rule is interesting or not. One strong rule can be interesting, if the itemsets in the rule is positively correlated. In addition to that, $\chi^2$ statistic can be used to see if the correlation is statistically significant or not.

### 2.4 Multi-Objective Rule Mining

Association rules are evaluated by metrics such as *support, comprehensibility, interestingness*, etc. These metrics can be thought of as different objectives of association rule mining. So, association rule mining is a multi-objective problem instead of single-objective problem. Multi-objective rule mining has been
discussed in [GN04], which has used Pareto based genetic algorithm to extract useful and interesting rules.

2.5 Feature Selection

Relevant feature selection is important in all kinds of databases including static and dynamic databases, because of the fact that all the attributes/features in a database are not important. So, it is required to find the relevant features/attributes in a database to better represent the domain. There have been some works in the field of relevant feature selection in a dataset. One of the earliest works is Branch and Bound [NF77]. The algorithm attempts to find the best set of features according to some monotonic function. Relief [KR92] is a weight-based algorithm. It uses random sample and is based on the concept of NearHit and NearMiss. The algorithm calculates weights of the features in each iteration and selects the features with highest weights. However, The algorithm is suitable for noisy, correlated features and binary classes. Another algorithm Focus [ADY1] selects features based on consistency measures. It works well with noise-free data. [LS96] also uses consistency measures to select subset of features. It first selects a random sample of features and then applies consistency measures on the sample. The algorithm uses one inconsistency threshold, which can be tuned according to requirement. The good things about the algorithm are that it is very simple to implement and guaranteed to find optimal subset. The algorithm MDLM [SDN90] is based on the concept that if the features in a subset $X$ can be expressed as a fixed non-class-dependent function of the features in another subset $Y$, then once the values in the features in the subset $X$ are known, the features in the subset $Y$ are useless. Minimum Description Length Criterion (MDLC) is used for this purpose. The algorithm exclusively searches all the possible subsets and returns the subset satisfying MDLC. This method can find all the useful features for Gaussian cases.
2.6 Discussion

As can be experienced from the related works reported so far that substantial works have been carried out in various dimensions of association rule mining techniques. Based on the survey, it can be observed that

1. Generation of unnecessary candidate itemsets by most of the algorithms such as Apriori is the main reason behind the degraded performance in terms of execution time. Even the performance of efficient algorithm with reduced candidate sets such as FP-growth also can be found to degrade with the increase of minimum support.

2. Partitioning is an effective approach of finding frequent itemsets over large databases. However, with the increase of dimensionality, performance of horizontal partitioning also degrades.

3. Borders algorithm suffers from the drawback of having to scan the entire database frequently.

4. There does not exist, to the best of our knowledge, any distributed version of Borders algorithm for distributed dynamic databases.

Feature selection plays an important role in machine learning problems. Based on the existing survey, it has been found that

- The existing algorithms cannot select relevant features in all kind of databases and they are very much time consuming.

- The drawbacks can be overcome by the use of frequent features.

Both static and dynamic association rule mining techniques, mostly fundamental association mining problems such as frequent itemsets generation problems, have been studied and analyzed thoroughly in this thesis.