Combined mining is an effective technique for assessing object relations and pattern relations, and for extracting and creating actionable complex knowledge (patterns or exceptions) in complex situations. The expectation of business person from complex data is fulfilled by combined mining to make decisions. Drawbacks in traditional data mining are overcome by a novel idea called combined mining. The concept of combined association rule is to find actionable knowledge from association rule. A traditional association mining often produces large number of association rules and that are difficult for users to understand such rules. Combined mining is a pillar for association rules. In this approach, first the association rule are filtered by varying support and confidence levels then using interestingness measure rules association rules are further extracted. This approach is done by using fuzzy logic.

3.1 BASE OF ASSOCIATION RULE MINING

The notion of mining association rules are as follows [27]. In the data mining, the association rule mining is introduced in [2] to detect hidden facts in large datasets and drawing inferences on how a subset of items influences the presence of another subset. Let S= \{S_1, S_2, S_3, \ldots, S_n\} be a universe of items and T= \{T_1, T_2, T_3, \ldots, T_n\} is a set of transactions. Then expression X \Rightarrow Y is an association rule where X and Y are itemsets and X \cap Y = \Phi. Here X and Y are called antecedent and consequent of the rule respectively. This rule holds support and confidence. Support is a set of transactions in set T that contain both X and Y and confidence is percentage of transactions in T containing X that also contain Y. An association rule is strong if it satisfies user-set minimum support (min_sup) and minimum confidence (min_conf) such as support \geq min_sup and confidence \geq min_conf. An
association rule is frequent if its support is such that support $\geq \minsup$. There are two types of association rules- positive association rules and negative association rules. The forms of rules $X \Rightarrow \neg Y$, $\neg X \Rightarrow Y$ and $\neg X \Rightarrow \neg Y$ are called negative association rules (NARs) [72]. In the previous research, NARs were discovered from both frequent and infrequent item sets.

As discussed above let $S$ be an universe of Items and $T$ be a set of transactions. It is assumed to simplify a problem that every item $x$ purchased in any given transaction $T$ is assigned a number for ex.- $Tid$. Now it is assumed that $X$ is an itemset then a transaction $t$ is assumed to $X$ iff $X \subseteq T$. Hence it is clear that an association rule is an implication of the form $X \Rightarrow Y$ where $X$ and $Y$ are subsets of $S$.

**Support**

The support is simply the number of transactions that include all items in the antecedent and consequent parts of the rule. (The support is sometimes expressed as a percentage of the total number of records in the database.)

$$Supp(X) = \frac{\text{No.of transactions containing } X}{\text{Total No. of Transactions}}$$

**Confidence**

Confidence is the ratio of the number of transactions that include all items in the consequent as well as the antecedent (namely, the support) to the number of transactions that include all items in the antecedent.

$$Conf(X \Rightarrow Y) = \frac{Supp(X \cup Y)}{Supp(X)}$$
Lift

Lift is nothing but the ratio of confidence to expected confidence. Lift is a value that gives us information about the increase in probability of the "then" (consequent) given the "if" (antecedent) part.

\[ \text{Lift}(X \Rightarrow Y) = \frac{\text{Supp}(X \cup Y)}{\text{Supp}(X) \times \text{Supp}(Y)} \]

3.2 ASSOCIATION RULE MINING ALGORITHMS

Several algorithms have been developed since the introduction of the Apriori algorithm [2]. Those algorithms are attempted to improve the efficiency of frequent pattern and/or association rule discovery. Most of the algorithms focus on either frequent itemset generation or discovering the association rules from the frequent itemsets. In contrary, Apriori provides solutions for both problems. This chapter will give a brief overview of some important mining algorithms. Exploring all the available algorithms for mining association rules will go beyond the scope of this thesis. Most of the algorithms have been developed for use with binary association rules, but they will equally work with the above described quantitative association rules.

There are two main strategies for developing an association rule mining algorithm. They are called breadth-first search (BFS) and depth-first search (DFS)[28].

In BFS, the support is first determined for all itemsets in a specific level of depth, whereas DFS recursively descends the structure through several depth levels. Thereby, association rule mining algorithms can be systematized as in Figure 3.1
The Apriori algorithm was the first attempt to mine association rules from a large dataset. It has been presented in [1] for the first time. The algorithm can be used for finding frequent patterns and also deriving association rules from them. Unlike in [2], rules having more than one element in the consequent are allowed. Such rules are called multi-consequent rules.

**Discovering Frequent Itemsets**

Generation of frequent itemsets, also called large sets here, makes use of the fact that any subset of a large itemset must as well be large. The number of items contained in an itemset is called by its size; say an itemset of size k is called a k-itemset. Within the itemset, the items are kept in lexicographic order. To represent the algorithm, the notation in Table 3.2 is used.
Table 3.1 Notation used in Apriori

<table>
<thead>
<tr>
<th>k-itemset</th>
<th>An Itemset having k items</th>
</tr>
</thead>
<tbody>
<tr>
<td>L_k</td>
<td>Set of large k – itemsets (those with minimum support). Each member of this set has two fields: i) itemset and ii) support count</td>
</tr>
<tr>
<td>C_k</td>
<td>Set of candidate k-itemsets (potentially large itemsets). Each member of this set has two fields: i) itemset and ii) support count</td>
</tr>
</tbody>
</table>

Each itemset has a count field associated with it, storing the support value. The pseudo code of the Apriori algorithm is given below. Firstly, the database is passed over in order to count the occurrences of single elements. If a single element has a support value that is below the defined minimum support, it does not have to be considered anymore because it can never be part of a large itemset. A subsequent pass k consists of two phases:

1. The discovered large itemsets of pass k -1, i.e. the sets L_{k-1}, are used to generate the candidate itemsets, C_k for the current pass.

2. The database is scanned once more in order to determine the support for the candidate itemsets C_k. If the support is above the minimum support, the candidates will be added to the large itemsets. Discovering the right candidates is crucial in order to prevent a long counting duration.

Apriori Algorithm

L_1 = \{ large 1-itemsets \};
For (k=2; L_{k-1} \neq \emptyset; \ k++) do begin

C_k Apriori-gen (L_{k-1}); // New candidates

forall transactions t \in D do begin

C_t = subset(C_k, t); // candidates contained in t

For all candidates c \in C_t do

c.count++;

end

L_k = \{ c \in C_k | c.count \geq \text{mins} \}

End

Answer = U_k L_k;

The Apriori-gen() function takes the large itemsets of the previous iteration as an input. These itemsets are joined together, forming itemsets with one more item than in the step before. After that, a prune step will remove any itemsets whose sub-combinations have not been part of the discovered sets in former iterations. The candidate sets are being stored in a hash-tree. This tree can either contain a list of itemsets, which is a leaf node, or a hash table, which is an interior node. The nodes contain the candidate itemset themselves. The subset function starts from the root node going towards the leaf nodes in order to find all the candidates contained in a transaction t. The itemsets starting with an item that is not contained in t will therefore be ignored by the function.

**Discovering Association Rules**

As stated before, association rules are allowed to have multiple elements in the antecedent as well as in the consequent. Only large itemsets are used to generate the association rules. The procedure starts with finding all possible subsets of the large itemset l. For each of those subsets, a rule is setup in the form a \rightarrow (1-a). If the confidence of the rule
is as least as big as the user-defined minimum confidence, the rule is considered to be interesting. All subsets are explored in order not to miss any possible dependencies. But, if a subset \( a \) of \( l \) does not generate an interesting rule, the subsets of \( a \) do not have to be explored. This will save computation power that would otherwise be wasted.

**Frequent Pattern Growth (FP-Growth)**

The FP-Growth algorithm allows generating frequent itemsets and tries to avoid generating a huge amount of candidates that is necessary for the Apriori algorithm. The heart of this algorithm is a compact representation of the original dataset without losing any information. This is achieved by organizing the data in a tree form, called the Frequent Pattern Tree, FP-Tree in short. The approach evolved out of the belief that the bottleneck of Apriori-like algorithms is the candidate-generation and -testing. The FP-Growth algorithm has been introduced in [29]. The algorithm first constructs the tree out of the original data set and then grows the frequent patterns. For a faster execution, the data should be preprocessed before applying the algorithm.

**Preprocessing the Data**

The FP-Growth algorithm needs the following preprocessing in order to be efficient: An initial scan over the dataset computes the support of the single items. As items that have themselves a support value below the minimum support can never be part of a frequent itemset, they can be discarded from the transactions [6]. The remaining items are recombined so that they appear in a decreasing order with respect to their support. The algorithm will work just fine without sorting the dataset, but it will perform much faster after doing so. With an ascending order, the algorithm performs even worse than using a random order Figure3.2
gives an example of how a transaction database will be preprocessed for the FP-Growth algorithm.

![Fig. 3.2 FP-Growth Preprocessing](chart.png)

### Constructing the FP-Tree

After having preprocessed the data, an FP-Tree can directly be constructed. A scan over the database has to be made, adding each itemset to the tree. The first itemset will be the first branch of the tree. In the transaction database of Figure 3.2, the first branch of the tree would be the items b, d and a. The second transaction shares a common prefix with the already existing set in the tree. In this case, the values along the path of the common prefix will be increased by one, and the remaining items will make new nodes for the tree. In our example, only one new node for e will be created. It is simply linked as a child of its ancestor. The tree corresponding to the transaction database of Fig. 3.2 is shown in Fig. 3.3. It represents the database in a compact format without the loss of any information.
Fig. 3.3 FP-Tree

Each node of the FP-Tree consists of three fields [29]:

- **item-name**: In this field, the name of the item that the node represents is stored.

- **Count**: The field count represents the accumulated support of the node within the current path.

- **node-link**: In order to build the structure of the tree, links have to be built between the nodes. The field node-link stores the ancestor of the current node, and null if there is none.

Having done, mining the database is not necessary anymore, now the FP-Tree is used for mining. The support of an itemset can easily be determined by following the path and using the minimum value of count from the nodes. For example, the support of itemset \{b, e\} would be 2, whereas the support of itemset \{b, e, a\} would only be 1.
**Mining the FP-Tree using FP-Growth**

The FP-Tree provides an efficient structure for mining, although the combinatorial problem of mining frequent patterns still has to be solved. For discovering all frequent itemsets, the FP-Growth algorithm takes a look at each level of depth of the tree starting from the bottom and generating all possible itemsets that include nodes in that specific level. After having mined the frequent patterns for every level, they are stored in the complete set of frequent patterns. The procedure of the algorithm is given below.

**FP-Growth Algorithm**

Procedure FP-Growth ( Tree α )

{
If Tree contains a single path P
Then for each combination (denoted as β)
of the nodes in the path P do
Generate pattern β U α with support =
Minimum support of nodes in β;
Else for each aᵢ in the header of Tree do {
   generate pattern β = aᵢUα with support=aᵢ.support;
   Construct β ‘s conditional pattern base and
   then β’s conditional FP-Tree Treeᵦ;
   If Treeᵦ≠Ø
   then call FP-Growth( Treeᵦ , β )
}
}
FP-Growth takes place at each of these levels. To find all the itemsets involving a level of depth, the tree is first checked for the number of paths it has. If it is a single path tree, all possible combinations of the items in it will be generated and added to the frequent itemsets if they meet the minimum support. If the tree contains more than one path, the conditional pattern base for the specific depth is constructed. Looking at depth a in the FP-Tree of Figure 3.4, the conditional pattern base will consist of the following itemsets: \( \{b, e : 1\}, \{b, d : 1\} \) and \( \{d, e : 1\} \). The itemset is obtained by simply following each path of an upwards. Table 3.2 shows the conditional pattern bases for all depth levels of the tree.

### Table 3.2 Conditional Pattern Bases

<table>
<thead>
<tr>
<th>Item</th>
<th>Conditional Pattern Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>( {b, e : 1}, {b, d : 1}, {d, e : 1} )</td>
</tr>
<tr>
<td>E</td>
<td>( {b : 2}, {b, d : 1}, {d : 1} )</td>
</tr>
<tr>
<td>D</td>
<td>( {b : 3} )</td>
</tr>
<tr>
<td>B</td>
<td>( \emptyset )</td>
</tr>
</tbody>
</table>

From the itemsets in the conditional pattern base, a so called conditional FP-Tree is constructed. This works in the same way as the construction of the initial tree, using the conditional pattern base as the transaction database. After constructing the conditional FP-Tree, the FP-Growth function is called again, making it a recursive function. The function is called until the tree contains only one single path or is empty. All the itemsets found in the
various conditional FP-Trees are stored and returned in the end as a list of all frequent itemsets in the FP-Tree and also in the database, respectively.

### 3.3 COMBINED ASSOCIATION RULE MINING

Let T be a dataset. In this dataset, each tuple is described by a schema $S = (S_{D_1}, \ldots, S_{D_m}, S_{A_1}, \ldots, S_{A_n}, S_C)$, in which $S_D = (S_{D_1}, S_{D_2}, \ldots, S_{D_m})$ are m non-actionable attributes, $S_A = (S_{A_1}, S_{A_2}, \ldots, S_{A_n})$ are n actionable attributes, and $S_C$ is a class attribute. Note that the data for combined association rule is not limited to one dataset. In fact, different kinds of attributes are often from multiple datasets [82].

Combined association rule mining is to discover the association among the ‘attribute-value’ pairs. For the convenience of description, call an ‘attribute-value’ pair an ‘item’. Suppose itemset $D \subseteq I_D$, $I_D$ is the itemset of any items with attributes $(S_{D_1}, S_{D_2}, \ldots, S_{D_m})$, itemset $A \subseteq I_A$, $I_A$ is the itemset of any items with attributes $(S_{A_1}, S_{A_2}, \ldots, S_{A_n})$, $C$ is 1-itemset of class attribute, a combined association rule set is represented as

$$
\begin{align*}
D + A_i &\Rightarrow C_{k_i} \\
& \vdots \\
D + A_i &\Rightarrow C_{k_i}
\end{align*}
$$

Here “+” means itemsets appearing simultaneously. Since one action may result in different classes while one class may correspond to different actions, $C_{k_1} \ldots C_{k_i}$ rather than $C_1 \ldots C_i$.

In order to make the combined association rules in a rule set containing the same non-actionable itemset, it is important to firstly discover frequent non-actionable itemsets. Once these itemsets are discovered, the relationships of frequent non-actionable itemsets with
target classes and actionable attributes are mined. In the rule generation step, the conditional support [83] is employed to tackle data imbalance problem.

For a single combined association rule \( D+A_i \Rightarrow C_{ki} \), the conventional interestingness measure are its confidence and lift. However, these two interestingness measures are not sufficient. For a discovered frequent pattern \( D+A_i \Rightarrow C_{ki} \) suppose \( \text{Conf}(D+A_i \Rightarrow C_{ki}) \) is 60% and the expected confidence of \( C_{ki} \) is 30%. So the lift of this frequent pattern is 2, which is high enough in most association rule mining algorithms. However the confidence of \( D \Rightarrow C_{ki} \) is 70% which means objects with non-actionable attribute \( D \) have 70% probability to be class \( C_{ki} \). On the other hand, if action \( A_i \) happens, objects with non-actionable attribute \( D \) only have 60% probability to be class \( C_{ki} \). Obviously action \( A_i \) is negatively correlated to class \( C_{ki} \) with respect to non-actionable itemset \( D \).

Hence a new lift named conditional lift is defined as follows to measure the interestingness of a combined association rule.

\[
\text{ConLift} = \frac{\text{Conf}(D+A_i \Rightarrow C_{ki})}{\text{Conf}(D \Rightarrow C_{ki})} = \frac{\text{Count}(D \cap A_i \cap C_{ki}) \cdot \text{Count}(D)}{\text{Count}(D \cap A_i) \cdot \text{Count}(D \cap C_{ki})}
\]

Where \( \text{ConLift} \) stands for the conditional lift of combined association rule \( D+A_i \Rightarrow C_{ki} \). \( \text{Count}(x) \) is the count of the tuples containing itemset “x”. Note that \( D, A_i \) and \( C_{ki} \) are all itemsets so that \( D \cap A_i \cap C_{ki} \) means \( D, A_i \) and \( C_{ki} \) occur simultaneously.

The combined association rule mining procedure consists of two steps. The first step is to find single rule composed of frequent itemsets. The second step is to extract interesting combined association rule sets. Since itemsets are treated as different groups, the time complexity of the algorithm is much lower than searching in the whole space of itemsets. In
order to calculate the interestingness measures, the support count of each frequent itemset is recorded in the frequent itemset generation step.

**Procedure for Collecting Patterns**

Discovering frequent non-actionable itemsets $I_D$ and the corresponding support counts $C_D$

For each frequent non-actionable itemsets $I_D$

Finding frequent itemsets including target class $I_{DC}$

Recording the support count $C_{DC}$ for each $I_{DC}$

Calculating conditional support $ConSup(DC)$

If $(ConSup(DC) > MinSup)$ for each $I_{DC}$

Finding candidate pattern of three kinds of itemsets $I_{DCA}$

Recording the support count $C_{DCA}$ for each $I_{DCA}$

Calculating conditional support : $ConSup(DA)$

Calculating $Conf$, $Lift$ and $ConLift$;

If $(Conf \geq min_c \& Lift \geq min_l \& ConLift \geq min_a)$

Adding the mined frequent itemsets to the rule set.

### 3.4 APPLICATIONS OF ASSOCIATION RULE MINING

The various applications of association rules for extracting useful information from the huge dataset are listed below:

**Medical diagnosis**

Applying association rules in medical diagnosis can be used for assisting physicians to cure patients. The general problem of the induction of reliable diagnostic rules is hard
because theoretically no induction process by itself can guarantee the correctness of induced hypotheses [56].

Practically, diagnosis is not an easy process as it involves unreliable diagnosis tests and the presence of noise in training examples. This may result in hypotheses with unsatisfactory prediction accuracy which is too unreliable for critical medical applications [23].

Serban [56] has proposed a technique based on relational association rules and supervised learning methods. It helps to identify the probability of illness in a certain disease. This interface can be simply extended by adding new symptoms types for the given disease, and by defining new relations between these symptoms.

**Protein sequences**

Proteins are important constituents of cellular machinery of any organism. Recombinant DNA technologies have provided tools for the rapid determination of DNA sequences and, by inference, the amino acid sequences of proteins from structural genes [5].

Proteins are sequences made up of 20 types of amino acids. Each protein has a unique 3-dimensional structure, which depends on amino-acid sequence; slight change in sequence may change the functioning of protein. The heavy dependence of protein functioning on its amino acid sequence has been a subject of great anxiety.

Lot of research has gone into understanding the composition and nature of proteins; still many things remain to be understood satisfactorily. It is now generally believed that amino acid sequences of proteins are not random.

Nitin Gupta, NitinMangal, Kamal Tiwari, and PabitraMitra[24] have deciphered the nature of associations between different amino acids that are present in a protein. Such
association rules are desirable for enhancing our understanding of protein composition and hold the potential to give clues regarding the global interactions amongst some particular sets of amino acids occurring in proteins. Knowledge of these association rules or constraints is highly desirable for synthesis of artificial proteins.

Census data

Censuses make a huge variety of general statistical information on society available to both researchers and the general public[45]. The information related to population and economic census can be forecasted in planning public services (education, health, transport, funds) as well as in public business (for setup new factories, shopping malls or banks and even marketing particular products).

The application of data mining techniques to census data and more generally to official data, has great potential in supporting good public policy and in underpinning the effective functioning of a democratic society[57]. On the other hand, it is not undemanding and requires exigent methodological study, which is still in the preliminary stages.

CRM of credit card business

Customer Relationship Management (CRM), through which, banks hope to identify the preference of different customer groups, products and services tailored to their liking to enhance the cohesion between credit card customers and the bank, has become a topic of great interest [10]. Shaw[58] mainly describes how to incorporate data mining into the framework of marketing knowledge management.

The collective application of association rule techniques reinforces the knowledge management process and allows marketing personnel to know their customers well to provide better quality services. Song[59] proposed a method to illustrate change of customer behavior
at different time snapshots from customer profiles and sales data. The basic idea is to
discover changes from two datasets and generate rules from each dataset to carry out rule
matching.