Lazy Associative Classification Based On Maximal Feature Selection

by

Akiz Uddin Ahmed

Exam Roll No: Curzon-801, Session:2007-08

Nahim Adnan

Exam Roll No: Curzon-811, Session:2006-07

Supervised by

Dr. Chowdhury Farhan Ahmed

Associate Professor

Department of Computer Science and Engineering

University of Dhaka

A thesis submitted in partial fulfilment of the requirements for the degree of

Department of Computer Science and Engineering

at the University of Dhaka

June 2012

Contents

Contents **i**

List of Tables **iii**

List of Figures **iv**

1 Introduction **1**
- 1.1 Associative Classification 1
- 1.2 Lazy Associative Classification 4
- 1.3 Motivation 4
- 1.4 Contribution of the Research 5
- 1.5 Thesis Organization 5

2 Literature Survey **7**
- 2.1 Associative Classification Techniques 9
  - 2.1.1 CBA Algorithm 9
  - 2.1.2 CMAR Algorithm 14
  - 2.1.3 CPAR Algorithm 22
- 2.2 Lazy Associative Classification Techniques 26
  - 2.2.1 LAC 26
- 2.3 Summary 28

3 Our Proposed Approach **29**
- 3.1 Prelimineries 30
- 3.2 LACF: Our Proposed Algorithm 31
  - 3.2.1 Descriptive Example 31
- 3.3 Algorithm 33
  - 3.3.1 Steps Of Algorithm 34

4 Performance Evaluation **39**
4.1 Performance Variation With Different Dataset Parameters . 39
4.2 Comparison of LACF and Other Associative Classifiers . . . 41
4.3 Performance for Different Values of l 43
4.4 Cost Analysis 45
4.5 Summary 46

5 Conclusion **47**
5.1 Research Summary 47
5.2 Scope of Future Studies 48

Bibliography **49**

List of Tables

2.1 A Traning Dataset 15
2.2 Rules found in a traning dataset 17
2.3 The observed contingency of rule R1 21
2.4 The observed contingency of rule R2 21
2.5 The expected contingency of rule R1 21
2.6 The expexted contingency of rule R2 22
2.7 Training Dataset 27
2.8 Projected Training Dataset D_A 27

3.1 Class labeled customer database from AllElectronics 31
3.2 Reffered class label and values for previous rules 33

4.1 Accuracy: C4.5, RIPPER, CBA, CMAR, CPAR, LACF. 42
4.2 Accuracy with number of records and attributes 42
4.3 Accuracy vs $\boldsymbol{l}$-value relation for different datasets 44

List of Figures

2.1 FP-tree for Example 2.1 16
2.2 CR-tree for Example 2.2 18
2.3 Rule generation process by CPAR 25

4.1 Number of Attributes and Accuracy Relation 40
4.2 Number of Records and Accuracy Relation 41
4.3 Average accuracy of classification techniques 43
4.4 Accuracy vs $\boldsymbol{l}$ value relation for two dataset 44
4.5 $\boldsymbol{l}$ value over threshold for vote dataset 45

Chapter 1

Introduction

Data mining refers to the process of discovering knowledge and determine patterns and their relationships by analyzing large database. A primary reason for using data mining is to assist in the analysis of collections of observations of behavior. Many business decisions are greatly benefited by data mining. It is a powerful technology with great potential to help industries to focus on the most important information. Data mining is applicable in biological and chemical research, bank, business companies, government decisions, computer security, education, finance, customer relationship management and many other areas. Many industries use data mining to increase sales and guess future investments. There are various feilds of data mining. *Frequent pattern mining, classification, prediction, clustering, graph mining, web mining* are some notable fields of data mining.

1.1 Associative Classification

In the field of data mining classification plays an important role. Classification is one of the most important topic in the field of data mining. With a given dataset, the task of classification is to build a model, which will classify new data object to certain class labels. There are sevaral classification techniques have been proposed by researchers so far in the field of pattern recognition, statistics, machine learning, data mining depending on the application nature. Classification techniques have been used in real life applications, including fraud detection, target marketing, performance prediction, manufacturing, medical diagonosis etc.

Frequent pattern mining[6, 14, 15, 25] is one of the basic task in data mining, which extracts interesting patterns, correlations among the set of attributes in transactional database. From these set of patterns interesting associations can be extracted. Despite of many classification algorithms from the different fields including machine learning, pattern recognition, statistics, a new classification algorithm has been proposed known as *Associative Classification*[7, 28, 31, 11, 32, 9, 26, 5, 4, 29], which is completely based on *association rule mining*[19, 18, 3, 21].

The process of building the associative classifier is basically composed of two phases. First, the class association rules or CARs are extracted from the training dataset. Second, select subset of high-quality rules from CARs applying *pruning techniques*[22, 23] to build the classification model for the training dataset. Basically a large set of rules are extracted which helps to build the classifier accurate. There exists a lot of redundant and unimportant rule in extracted CARs. So pruning techniques are applied to select high-quality rules. Various approaches are used in different associative classification algorithms. Basic approaches as pruning based on *support threshold, confidence threshold, database coverage, correlation measurement* are applied. Then the classifier is built with the selected CARs. A unknown class of an object is then predicted by the classifier. The class label of the object is then determined same as a CAR or a small set of CARs whose attributes are highly matched with the object. The selection of the appropriate CAR or CARs is done in different approaches in different algorithms. *Database coverage, χ^2 measurement, laplace accuracy* are applied in classification techniques [7, 28, 31] respectively.

CBA[7] is a naive approach, which is based on the *Apriori*[6] algorithm. Apriori algorithm is a frequent pattern mining algorithm, which is used in CBA in a modified fashion. A huge number of CARs are extracted from the training dataset. First, sorting the rules based on support and confidence threshold. Then CBA implemented database coverage technique to evaluate high quality rules. CBA classifies test object by a single rule which is best in the classifier for the object.

CMAR[28] is a multi class associative classification algorithm. CMAR uses a small subset of CARs to classify rather than only a single CAR. CMAR achieves higher accuracy than CBA for involving a set of rules to classify for class imbalanced dataset. It uses modified *FP-growth*[14] frequent pattern generation algorithm to extract the set of CARs. FP-growth is mines frequent pattern from the dataset very efficiently and with a low memory requirement. CMAR adopts a tree for storing as the CR-tree ideas based on *FP-tree*[14]. By the CR-tree, CMAR adopts effiency in using the CARs in classification phase. For the high quality CARs to classify, CMAR prunes rules based on support, confidence, correlation and database coverage. To build the classifier with high quality CARs, CMAR involves correlation between object attributes and class label. During classification phase, CMAR uses a subset of rules from CR-tree and determining class label based on χ^2-*measurement*[28].

For mining CARs, CPAR[31] follows a completely different algorithm than frequent mining algorithm. CPAR uses rule generation from the dataset directly as FOIL[17] for mining CARs. Rules are generated by joining attributes based on satisfying positive examples in training dataset. This technique is applied for each class. In this way, CPAR extracts the most appropriate set of CARs. For classification, CPAR measures the appropriate CARs using *Laplace expected error estimation*[31].

All the associative classification techniques are based on frequent pattern mining. But sometimes it is useful to use *negative association rule* in associative classification for improving accuracy. Some associative classification algorithms[20] have been proposed using mining negative association rules[3, 21]. *Emerging patterns*[8] is another form as itemset, which is used for retrieving useful trends in dataset. Some associative classification techniques[16, 11, 8] have been proposed based on emerging patterns.

1.2 Lazy Associative Classification

Some new form of associative classifications have been proposed based on the lazy learning technique. In lazy learning technique, learner simply stores the training dataset and starts processing when a test tuple is given. The idea is employed in lazy associative classification as pruning the CARs when classifier starts classifying the test tuples. If a subset of CARs misclassify test tuples then based on some conditions they are pruned. The basic idea of lazy learning approach is to minimize the number of pruning rules and keep the most preferable high quality classfication rules. LAC[5], LACI[26], L^3[9] are some of the lazy associative classification algorithm that have been proposed.

1.3 Motivation

Renowned associative classifiers like CBA, CMAR, CPAR generates rule from training tuples to classify unseen test tuples. These rules are ignorant about the future test tuple properties and thus suits for some test tuples and useless for others. They fail to explore and take advantages of the properties revealed later by the test tuples. They also emphasis only on support and confidence. Thus small rules gets more priority as it is likely that small rules will occur more times than long rules in database. This technique presents feature selection bias and hampers performance. They do not consider rule length to evaluate a rule's importance . We can estimate that a rule with more participating attributes is more reliable.

Some drawbacks of the renowned associative classifiers

1. Associative classifiers generate rules from training tuples which are unable to cope with varying nature of all the upcoming test tuples. This affects performance.It is better to generate rules from test tuples.
2. For minimum support and confidence value includes some unnecessary small rules and excludes rather potential long rules.
3. They present feature selection bias as the same dataset is used for both support, confidence calculation and rule generation.
4. Generate redundant rules by not including a minimum number of participating attributes in a rule.

5. Affects classification by not including the number of participating attributes in to rule evaluation method.

1.4 Contribution of the Research

Some improvements over the renowned associative classifiers are achieved by our algorithm, LACF(Lazy Associative Classification Based On Maximal Feature Selection), considering new classification properties. Contributions of the research are as follows

1. LACF generates only the potential rules for a specific test tuple. Thus it is better to classify each tuple with a set of dedicated rules.
2. LACF defines a minimum rule length for potential rules which is best for not losing necessary information and classifying each tuple.
3. Through LACF, we have formulated the relation between rule length and rule evaluation method for better classification.

1.5 Thesis Organization

The research work is carried out in order to achieve the objectives of the research. A brief overview of the contents of the chapters is provided as follows:

Chapter 2 describes the literature review and background study regarding this field.

Chapter 3 contains elaborate description of our proposed algorithm and some analytical discussion about the algorithm.

Chapter 4 focuses on the experimental results showing the performance of our proposed algorithm and comparison among our algorithm(LACF) and some renowned associative classification algorithms.

Chapter 5 includes some ideas about future scope of improvement over LACF algorithm.

Chapter 2

Literature Survey

In this chapter, some renowned and very basic associative classification algorithms with different approaches will be explained briefly. In associative classification algorithms, it uses a set of association rules to formulate a classifier. Then the classifier classifies the unknown object classes or attributes. There are a lot of approaches for determining the association rules. One of the basic techniques for generating association rules is based on frequent pattern mining. Another approach is using of rule based generation. Some other associative classification approaches have been introduced integrating with lazy learning technique. We will explain two associative classification algorithms based on frequent pattern mining named as CBA, CMAR. Another technique based on rule generation technique named as CPAR, which is based on FOIL. At last, we will explain the LAC (Lazy Associative Classification). We will focus to the LAC algorithm because our proposed algorithm will be using Lazy learning approach as LAC. We will explain the four algorithms with respect to association rule generation, selection of association rules for the classifier and classification approach through the classifier.

CBA (Classification Based on Association)[7] algorithm, which is based on frequent pattern mining. The approach which is used for generating frequent pattern is the Apriori algorithm[6]. Apriori algorithm is a naive approach and very easy to understand. It has some problems as :

- It uses a user defined minimum support threshold.

- It needs to scan the database many times to generate the frequent patterns. So, it will be slower, if the database is large.

A set of highly predicted association rules are selected from the set of frequent patterns generated by Apriori algorithm is used to form the classifier based on the support and confidence threshold. Then the classifier is used to classify the unknown object class.

Another classification technique known as CMAR (Classification Using Multiple Association Rules)[28] is based on multiple association rules. This technique is basically mined class association rules through a CR-tree, which is based on the renowned frequent pattern generation algorithm known as FP-growth (Frequent Pattern Growth)[14]. It's basic functionalities are as follows:

- Through the CR-tree structure, CMAR adopts highly efficient methods for storage and retrieval of association rules.
- It also uses minimum support threshold, confidence, correlation and database coverage to prune ineffective and redundant association rules from the generated association rules.
- Classification task is based on *weighted-χ^2 analysis* using multiple association rules.

CPAR (Classification based on Predictive Association Rules)[31], which uses advantages of the traditional rule based classification and associative classification algorithm. For faster rule generation, rule based techniques are more preferable but they tend to less accurate than associative classification. It's advantages are as follows:

- Association rules are produced from the training dataset directly.
- It uses expected accuracy to measure rules and uses k-rules to classify.

LAC (Lazy Associative Classifier)[5], which performs classification task on demand driven basis. It generates classification rules after getting the unknown object attributes. It considers only the attributes appeared on the unknown object. It forms the association rules through local focus on the training dataset using the attributes of the unknown object. It uses simple caching mechanism to overcome the processing problem.

2.1 Associative Classification Techniques

Databases are rich with hidden information that can be used for intelligent decision making. Classification is a form of data analysis that can be used to extract models describing important data classes. For example, we can build a classification model to categorize bank loan applications as either safe or risky.

Frequent patterns and their corresponding association or correlation rules characterize interesting relationships between attribute conditions and class labels, and thus have been recently used for effective classification. Association rules show strong associations between attribute-value pairs (or items) that occur frequently in a given data set[12]. From these association rules, a small set of rules is selected through different rule measurements. A classification model is then built with the set and used to classify the unknown object class label. In the following subsections, we will discuss three associative classification algorithms.

2.1.1 CBA Algorithm

This subsection discusses the CBA[7] algorithm briefly. CBA algorithm is the simplest associative classification algorithm. It generates a set of CARs(class association rules) by the well known Apriori algorithm. Apriori algorithm needs to have a predefined minimum support threshold value. Then it prunes CARs based on another predefined confidence threshold. For classifying a new class label of an object it chooses the best matched rules from the set.

Definition(CAR(Class Association Rule)) 2.1.1. *A special subset of association rules generated from frequent mining algorithm, whose right-hand-side are restricted only to the unknown class label, is defined as CAR.*

In this framework, the dataset will be used is basically relational table, which consists of N rows with l distinct attributes and classified into q known class labels. Attributes can be *discrete* or *continuous.* If *continuous*, then it will be discretized into intervals using discritization algorithm[10, 1, 13] and then mapping these intervals into consecutive integers as categorical attribute. A row is treated as a set of (*attribute, integer-value)* pairs and a class label. Each (attribute, integer-value) is called an *item.*A *ruleitem* is of the form :

$$< condset, y >$$

Where *condset* is a set of *item*s and y is the class label from q. Each *ruleitem* basically represents a association rule of the form as follows:

$$condset \rightarrow y$$

Number of appearance of the *condset* in the dataset is called the *support-Count* of the *condset*(*condsupCount*). Number of appearance both of the *condset* and class label y in the dataset is known as the *supportCount* of the *ruleitem*(*rulesupCount*). If a *ruleitem* satisfies minimum support threshold, then it is called *frequent ruleitem*, otherwise *infrequent.* The support of a *ruleitem* is as follows:

$$support(ruleitem) = \frac{rulesupCount}{|D|} * 100\%$$

where, |D| is the number of tuples in the dataset.
Now, the *confidence* of a *ruleitem* is as follows:

$$confidence(ruleitem) = \frac{rulesupCount}{condsupcount} * 100\%$$

For all the *ruleitems* that have the same *condset*, the *ruleitem* with the highest *confidence* is chosen as the possible rule (PR) representing this set of *ruleitem*s. If there are more than one *ruleitem* with the same highest *confidence*, we randomly select one *ruleitem.*
For example[7], the is two *ruleitem*s with same *condset*,

1. <(A,1),(B,1),(class:1)>
2. <(A,1),(B,1),(class:2)>

Assume the support count of the *condset* is 3. The *supportcount* of the first *ruleitem* is 2, and the second *ruleitem* is 1. Then, the *confidence* of *ruleitem* 1 is 66.7%, while the *confidence* of *ruleitem* 2 is 33.3%. With these

two *ruleitems*, we only produce one PR (assume |D| = 10):
<(A,1),(B,1)→(class,1)> [support=20%, confidence=66.7%]

If the confidence is greater than minconf , we say the rule is accurate. The set of class association rules (CARs) thus consists of all the PRs that are both frequent and accurate. The whole process consists of two steps:

1. To generate a complete set of CARs whose support and confidence will be greater than predefined minimum threshold.
2. Using the set to build the classifier.

CBA-RG Algorithm

CBA-RG algorithm extracts all *frequent ruleitems* from the dataset. It scans dataset multiple times during the process. First, it extracts individual *frequent ruleitems* and using these *ruleitems* to produce *candidate ruleitems* for possibly new *frequent ruleitems.* Then, it scans the dataset to find out the *frequent ruleitems.* In this way, it generates all the CARs.

Let a *ruleitem* whose *condset* has k items is called *k-ruleitem.* Let F_k denote the set of *frequent k-ruleitems.* Each element of this set is of the following form:

<(*condset*, *condsupCount*), (y , *rulesupCount*)>

Let C_k be the set of candidate k-ruleitems. The CBA-RG is given in Algorithm 2.1.

Algorithm 2.1. CBA-RG Algorithm[7]

Input : Dataset **D**, minimum support threshold **minsup**

Output : All frequent CARs

```
procedure CBA-RG($D, minsup$)
    $F_1 \leftarrow \{large - 1 - ruleitems\}$;
    $CAR_1 \leftarrow genRules(F_1)$;
    $prCAR_1 \leftarrow pruneRules(CAR_1)$;
    for ($k = 2; F_{k-1} \neq \emptyset; k++$) do
```

```
        $C_k \leftarrow candidateGen(F_{k-1};$
        for each data case d $\in$ D do
            $C_d \leftarrow ruleSubset(C_k, d)$
            for each candidate c $\in C_d$ do
                $c.condsupCount++;$
                if d.class = c.class then
                    c.rulesupCount++;
                end if
            end for
        end for
        $F_k \leftarrow c \in C_k | c.rulesupCount \geq minsup;$
        $CAR_k \leftarrow genRule(F_k);$
        $prCAR_k \leftarrow pruneRules(CAR_k)$
    end for
    $CARs \leftarrow \cup_k CAR_k;$
    $prCARs \leftarrow \cup_k prCAR_k;$
end procedure
```

1-frequent ruleitems are discovered from the first pass within the line 1-3. In line 2, *1-ruleitems* are generated as CAR_1 and pruned in line 3. In line 4-18, the algorithm performs major operations. For generating C_k, it first generates *frequent ruleitems* F_{k-1}. Then, it scans the dataset and discovers new CAR in line 15. After pruning on the discovered CAR on line 16, it generates all the CAR for *k-ruleitems.* The complete set of CARs are saved as line 18 after pruning. For discovering all the CARs, the algorithm needs to perform multiple passes over the dataset.

Building The Classifier

Now CBA-CB algorithm will be presented, which uses the complete set of CARs to formulate the classifier. Oredering of the rules needs to be defined.

Rule Ordering

Two rules as r_i and r_j, then r_i has higher precedence than r_j or $r_i \succ r_j$ if and only if

1. r_i has higher confidence than r_j.
2. If confedence is equal but support of r_i if greater than r_j.
3. If both the confedence and support are equal but r_i is generated before r_j.

Let, R be set of generated rules(i.e.,CARs) and D be the training dataset. The idea behind to build the classifier as a set high precedence rules in R by covering D. Format of the classifier is as follows:

$$< r_1, r_2, ..., r_k, default - class >$$

For classifying an unknown instance, first rule that satisfies the instance will classify. If no matches are found then it will be classified into the dafault-class. The algorithm for building the classifier is given in Algorithm 2.2. It is a naive approach and it involves three steps:

Step 1. Sort the generated CARs or R according to the relation "$\succ$".

Step 2. For each rule r from R, it goes through D and count the number of of rows it covered. If r covers at least one row, then it is a potential rule for the classifier. Those cases are covered by rule r is then removed from D. From the remaining trining dataset, default-class is then selected by the majority class in the dataset. The computation of the error made by the classifier C and default-class are computed. when there is no rule or no training dataset, the rule selection process is then completed.

Step 3. Discard the rules that are not going to improve accuracy.

Algorithm 2.2 CBA-CB. CBA Classifier[7]

Input : Discovered CARs as R and Dataset D

Output : CBA Classifier

```
procedure CBA-RG($R, D$)
    $R \leftarrow sort(R)$;
    for each rule r $\in$ R in sequence do
        $temp \leftarrow \emptyset$;
        for each data case d $\in$ D do
```

```
            if d satisfies the condition of R then
                store d.id in temp and mark r if it correctly classifies d;
            end if
        end for
        if r is marked then
            insert r at the end of C;
            delete all the cases with the ids in temp from D;
            select a default class in C;
            compute the total number of errors;
        end if
    end for
    Find the first rule p in C with the lowest total number of errors and
  drop all the rules afterp in C
    Add the default class associated with p to end of C, and return C (our
  classifier).
end procedure
```

It is very simple appraoch. Some efficiency problem occurs when the overall dataset is larger than the main memory. Some improvements are proposed to overcome the memory issue. But it was the first main proposed algorithm in this domain.

2.1.2 CMAR Algorithm

CBA and CBA type associative classification algorithms basically tend to predict unknown class label of an object through predefined confidence threshold. But it is not guaranteed that it will always predict the right class label. Another drawback of these algorithms is that they produce a lot of CARs and suffer from performance degradation. From these observations, CMAR[28] adopts a CR-tree[28] data structure to generate, store and retrieve CARs, which provides better performance from previous algorithms. It also determines a class label by a set of rules instead of just one rule.

Classification Technique

Let, a data object as obj=(a_i,....,a_n) follows the schema (A_1,A_2,.....A_n), where A_i is the attributes and n is the number of attributes in the schema. Let, C=(c_1,...,c_m) be the finite set of distinct class labels. A trining dataset is a dataset, in which for all object obj there is a class label c $\in$ C. The task of the classifier will be for an unknown object, it will return a class label c $\in$ C.

A *pattern* $P=(a_{i_1},....,a_{i_k})$ is a set of (attribute-value) pair such that for ($1 \leq$ j $\leq$ k), $a_{i_j} \in A_{i_j}$, and $i_j \neq i_{j'}$ for j'. A data object *obj* is said to be matched with *pattern* $P=(a_{i_1},....,a_{i_k})$ if and oly if for ($1 \leq$ j $\leq$ k), *obj* has a value a_{i_j} in attribute A_{i_j}.

CMAR consists of two phases : *rule generation* and *classification.*

- In the *rule generation* phase, CMAR mines the complete set of CARs as R:P$\rightarrow$c, where P is from the training dataset and c is class label. From the set, it discards unnecessary and low quality rules employing the support and threshold.
- In the *classification* phase, CMAR selects a subset of highly predictable rules to classify an unknown object.

Mining CARs

For generating complete set of CARs, CMAR employs a frequent mining algorithm. For faster mining and retrieving of CARs, CMAR adopts a variant of FP-grorth[14] algorithm, which is faster than the Apriori[6] like methods. The general idea of mining CARs is explained in the following example.

Example 2.1 : Mining CARs[28]

A training dataset is given in Table 2.1. Let, the support threshold be 2 and confidence is 50%. Then, CMAR mines CARs as follows:

Row-id	A	B	C	D	Class-Label
1	a_1	b_1	c_1	d_1	A
2	a_1	b_2	c_1	d_2	B
3	a_2	b_3	c_2	d_3	A
4	a_1	b_2	c_3	d_3	C
5	a_1	b_2	c_1	d_3	C

Table 2.1: A Traning Dataset

First, CMAR scans the dataset and finds the set of attributes that passes the support threshold. Other attributes are deleted. So, the set is F=$\{a_1,b_2,c_1,d_3\}$ because they all appeared twice in the dataset and F is frequent itemset.

Then, CMAR sort the F-list in support decreasing order as F-list= $\{a_1$-b_2-c_1-$d_3\}$. CMAR scans the dataset again and constructs a FP-tree as shown in Figure 2.1.

At first, each tuple is scanned from the dataset and sorted according to F-list. Then, it is inserted into the root of the FP-tree and class label is attached to the last node. For a tuple, if it's prefix matched with any of the previous branch, then they share the common prefix.
All nodes with same attribute value are linked together as a queue started from the header table.

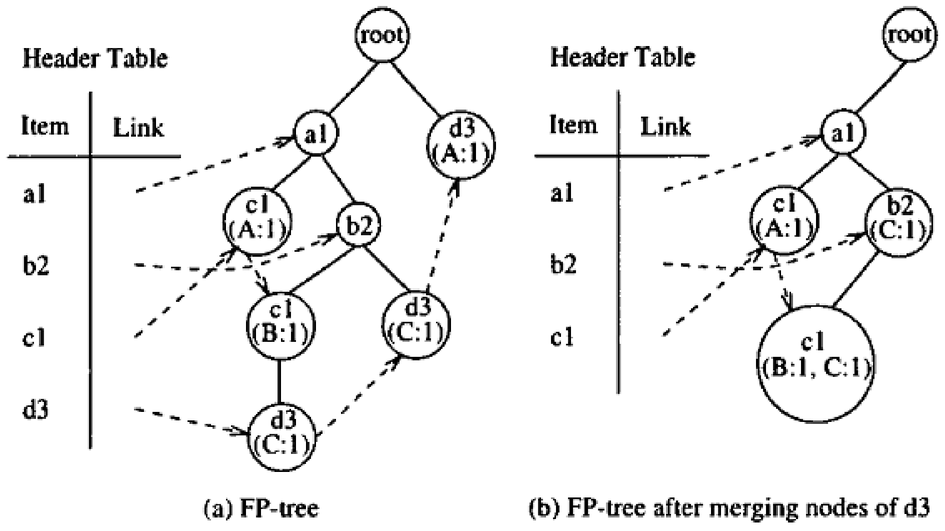

Figure 2.1: FP-tree for Example 2.1

The set of CARs can be divided into four subsets without overlapping : (1) the ones having d_3; (2) the ones having c_1 but no d_3; (3) the ones having b_2 but no d_3 nor c_1; and (4) the ones having only a_l. CMAR finds these subsets one by one.

For finding the subset of rules having d_3, CMAR traverses nodes having attribute value d_3 and look upward to collect a d_3-projected database, which contains three tuples: (a_1, b_2, c_1, d_3) : C, (a_l, b_2, d_3) : C and d_3 :A. It contains all the tuples having d_3. The problem of finding all frequent patterns having d_3 in the whole training set can be reduced to mine frequent patterns in d_3-projected database.

In d_3-projected database, a_1 and b_2 are the frequent attribute because they passed the minimum support threshold. d_3 will be ignored locally because it can not passed the threshold. In this process FP-tree can be mined recursively. For more details please see[14].

In d_3-projected database, a_1 and b_2 appears together frequently, thus a_1b_1 is a frequent pattern. But a_1 and b_2 are two subpatterns and have same support count. So, to avoid redundancy both a_1 and b_2 can be pruned. Now, only $a_1b_2d_3$ will be adopted. So, $a_1b_2d_3 \rightarrow$ C will be generated with support 2 and confidence 100% based on the class distribution.

After search for rules having d_3, all nodes of d_3 are merged into their parent nodes, respectively. That is, the class label information registered in d_3 node is registered in its parent node. The FP-tree is shrunk as shown in Figure 2.1(b). Please note that this tree-shrinking operation is done at the same scan of collecting the d3-projected database.

The remaining subsets of rules can be mined similarly.

CR-tree Stucture For Storing CARs

Once the set of CARs is generated they are stored into CR-tree, which is a prefix tree structure. In the following example, the procedure will be explained.

Example 2.2 : (CR-tree)[28]

After mining a training dataset, four rules are found as shown in Table 2.2.

Rule-id	Rule	Support	Confidence
1	abc→A	80	80%
2	abcd→A	63	90%
3	abc→B	36	60%
4	bcd→C	210	70%

Table 2.2: Rules found in a traning dataset

A CR-tree is built for the above CARs and shown in Figure 2.2.

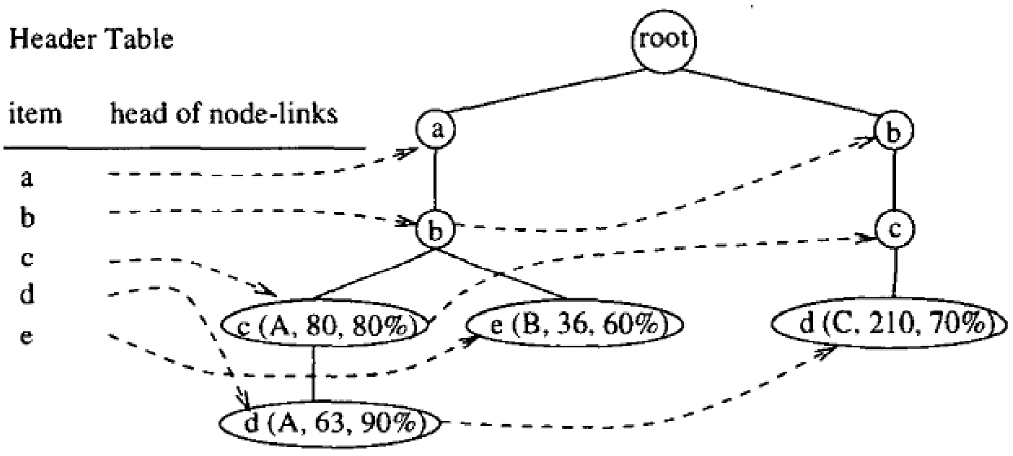

Figure 2.2: CR-tree for Example 2.2

CR-tree has a root node. Left hand side of the rules are sorted according to their frequency and most frequent rule is appeared at the left of the tree.

First rule {abc→A} appeared in the tree as a path from the root node. In the last node, class label, support, confidence appeared as (A, 80, 80%).

The second rule {abcd→A} has a common prefix of abc with the first rule so, it shares the prefix and extends it's path to the last node with d along with the class label, support, confidence.

In this process, all the rules are inserted into the CR-tree. From the Figure 2.2, it can be noticed that the rules with highest support and confidence are appeared in the left side of the CR-tree.

Rule Pruning

For effective and efficient classification, redundant and noisy rules need to be pruned. For pruning, a simple ranking methodology has been used. Let, R1 and R2 are two rules and R1 has *higher rank* than R2 if and only if

1. confidence(R1)   confidence(R2).
2. confidence(R1) = confidence(R2) and support(R1)   support(R2).

3. confidence(R1) = confidence(R2) and support(R1) = support(R2) but, R1 has less attributes in it's left side than R2.

In addition, a rule R1 : P→c is said to be a *general rule* w.r.t. R2 : P'→c' if and only if P is a subset of P'.

First, the basic motivation of rule pruning involves keeping general rules with high confidence and pruning specific rules with low confidence. Let, R1 and R2 are two rules and R1 is a *general rule* w.r.t. R2. Then, CMAR prunes R2 if R1 has *higher rank* than R2.

Second pruning phase starts by measuring the correlation between P with c. Let, R : P→c is a rule where the pattern P is positively correlated with the class label c, then CMAR uses this rule for classification. Negatively correlated rules are thus pruned. CMAR performs this method of pruning by classifying with the strong implications and removing negatively correlated rules as removing noise. Correlation measurement is done by χ^2 *testing.*

Third phase of rule pruning is based on the *database coverage.* CMAR uses a *coverage threshold*[28] to select database coverage as the **Algorithm 2.3**. The process is as CBA but some changes has been made. In CBA a data object is removed after covering by any rule but in CMAR, object is removed after covering by at least δ rules, where δ is threshold value for removing a data object. Basic motivation is that when classifying a new data object, it may have more rules to consult and may be classified more accurately.

Algorithm 2.3: Selecting rules based on database coverage[28]

Input : a set of rules as R and a coverage threshold value δ

Output : a subset of classification rules

```
procedure RulePruning($R, \delta$)
    1. Sortrules in the rank descending order;

    2. For each data object in the training data set, set its cover-count to
0;
```

3. While both the training data set and rule set are not empty, for each rule R in rank descending order, find all data objects matching rule R. If R can correctly classify at least one object then select Rand increase the cover-count of those objects matching R by 1.A data object is removed if its cover-count passes coverage threshold δ;

end procedure

Classification based on Multiple Rules

After generating the set of rules now CMAR is ready for classifying new data object. For a new data object, CMAR collects the subset of rules which are matched with the object. In the following discussion, the process of determining the class label from the subset of rules will be explained.

Simply, if all the rules have the same class label, CMAR just assigns the class label to the object.

If the rules are not consistent, then CMAR divides the rules into some groups based on the distinct class labels. Each group has a distinct class label. CMAR computes the combined group effect and assigns the class label of the most strong group to the new object.

To measure the combined effect of a group, the correlation of the rules from a group can be evaluated by employing χ^2 testing. But simply choosing of highest χ^2 value can be biased towards the miniority classes, which is undesired. The problem is illustrated in the Example 2.3.

Example 2.3[28]

In a credit card approval case, there are two rules as follows:

R1 : {job=no→ rejected }[support=60, confidence=60%],
R2 : {education=university→ approved }[support=200, confidence=99.5%]

The expected and observed contingencies are shown in the Table 2.3, Table 2.4, Table 2.5, Table 2.6.[28]

R1	approved	rejected	total
job=yes	438	32	470
job=no	12	18	30
total	450	50	500

Table 2.3: The observed contingency of rule R1

R2	approved	rejected	total
edu=univ	199	1	200
edu≠univ	251	49	300
total	450	50	500

Table 2.4: The observed contingency of rule R2

R1	approved	rejected	total
job=yes	423	47	470
job=no	27	3	30
total	450	50	500

Table 2.5: The expected contingency of rule R1

Based on the observed and expected values, the χ^2 values for R1 and R2 are 88.4 and 33.6, respectively. For a customer having no-job and with university education, it is possible to predict his/her application would be rejected using rule R1 ,if the choice of rules is based on only χ^2 values. However, rule R2 is intuitively much better than R1 since R1 has much higher support and confidence.

For classifying without biasing towards miniority class, CMAR adopts a new approach of evaluating groups through χ^2 value as follows:

For each rule R : P $\rightarrow$ c, let sup(c) be number of objects that have class label c. |T| be the number of data objets in the training dataset. CMAR defines maxχ^2 for rule R as follows :

$$max\chi^2 = (minsup(P), sup(c) - \frac{sup(P)sup(c)}{|T|})^2|T|e$$

R2	approved	rejected	total
edu=univ	180	20	200
edu≠univ	270	30	300
total	450	50	500

Table 2.6: The expexted contingency of rule R2

where,

$$e = \frac{1}{sup(P)sup(c)} + \frac{1}{sup(P)(|T| - sup(c))} + \frac{1}{(|T| - sup(P))sup(c)} + \frac{1}{(|T| - sup(P))(|T| - sup(c))} \quad (2.1)$$

Upper bound of the rule is computed through the maxχ^2 value. The weighted χ^2 measure is for the group is defied as follows:

$$\sum \frac{\chi^2 \chi^2}{max\chi^2}$$

For overcoming the bias of χ^2 value favoring miniority class, CMAR uses the ratio χ^2 value against it's upper bound.

CMAR handles the problem of huge number of association rule generation very efficiently. It also achieves efficiency by CR-tree. CMAR overcomes the problematic issues with main memory from CBA by using the FP-growth approach. It classifies new class labels more accurately than CBA. CMAR discovered the fact of classification using a set of rules rather than a single best rule. It achieved novelty through classification using multiple rules. Though many other efficient algorithms have been proposed so far, CMAR was a novel algorithm at that time.

2.1.3 CPAR Algorithm

A huge set of CARs are generated through the frequent pattern mining algorithms. So, associative classification algorithms which are based on frequent pattern mining algorithms tend to have huge processing overhead. Traditional rule based classification algorithms such C4.5[24], FOIL[17] are faster than associative classifiers though they tend to be less accurate.

CPAR(Classification Based On Predictive Rule Mining)[31] employs both the advantages of traditional rule based classifier and associative classifier. CPAR generates complete set of CARs through the dataset directly which is much much faster than the rule generation process through frequent pattern mining algorithm. It employs dynamic programming to avoid repeated calculation in rule generation.

Predictive Rule Mining(PRM) In CPAR

Rule generation process of CPAR is basically a modified version of FOIL[17]. For calculating the gain of a literal(same as item in CBA) CPAR uses a PNArray.

PNArray Structure

A PNArray stores the following information of a rule r as follows:

- P and N : the number of positive and negative examples satisfying r's body.
- P(p) and N(p) : for each possible literal p, the numbers of positive and negative examples satisfying the body of rule r', the rule constructed by appending p to r.

Rule generation process of CPAR is given as sudocode in Algorithm 2.4.

Algorithm 2.4 : Predictive Rule Mining(PRM)[31]

Input : Training set D = P ∪ N.(P and N are all positive nad negative examples respectively.)

Output : A set of rules for predicting class labels.

```
procedure PRM($D$)
    set the weight of every example to 1
    ruleset $R \leftarrow \emptyset$
    $totalWeight \leftarrow TotalWeight(P)$
    A ← Compute PNArray from D database
    while TotalWeight(P) $\geq \delta$.totalweight do
        N'← N, P'← P, A'← A
        rule r ← emptyrule
        while true do
            find best literal p according to A'
            if $gain(p) < min\_gain$ then
                break
            end if
            append p to r
            for $each example t in p' \cup N' not satisfying r's body$ do
                remove t from P' or N'
                change A' according to the removal of t
            end for
        end while
        $R \leftarrow R \cup r$
        for $each example t in P satisfying r's body$ do
            t.weight ← $\alpha * .weight$
            change A according to the weight decreased
        end for
    end while
    return R
end procedure
```

Now, we will discuss the rule generation process of CPAR in detail. Assume, in the process of building a rule after finding the best literal p, another literal q is found which has gain same as p. Now, technically p will append to rule r and q is also appended to the rule r and inserted into the queue. Whenever a new rule is to be built, the queue is first checked. Rule generation is explained in the following example.

Example 2.4. Rule Generation[31]

Let, $(A_1 = 2)$ is the first literal selected and other two literals $(A_2 = 1)$, $(A_3 = 1)$ are found. Literal $(A_2 = 1)$ is selected and rule is generated along this direction. Now, $(A_1 = 2, A_3 = 1)$ is taken as the current rule. Again, two literals with similar gain $(A_4 = 2)$ and $(A_2 = 1)$ are selected and two rules are generated along both the directions as depicted in Figure 2.3:

Rule Evaluation

For a rule r= "$p_1 \wedge p_2 \wedge .... \wedge p_l \rightarrow c$", CPAR calculates it's expected accuracy as the probability that an example satisfying r's body satisfying class label c or Prob(t$\in c|t$ satisfies r's body).

$$(A_1 = 2,\ A_2 = 1,\ A_4 = 1).$$
$$(A_1 = 2,\ A_3 = 1,\ A_4 = 2,\ A_2 = 3).$$
$$(A_1 = 2,\ A_3 = 1,\ A_2 = 1).$$

$$\begin{tikzcd}
A1=2 \arrow[r] \arrow[dr] & A2=1 \arrow[r] & A4=1 & \\
 & A3=1 \arrow[r] \arrow[dr] & A4=2 \arrow[r] & A2=3 \\
 & & A2=1 &
\end{tikzcd}$$

Figure 2.3: Rule generation process by CPAR

CPAR also uses the Laplace Expected Error Estimate[31] to estimate the accuracy of the rule which is defined as follows:

$$LaplaceAccuracy = \frac{n_c + 1}{n_{tot} + k}$$

where k is the number of classes, n_{tot} is the number of examples satisfying the rules body, among which n_c examples belong to c[31].

Classification Process

CPAR has distinct group of rules for each class label. For classifying an object, CPAR selects all the rules whose body is matched with the object. From the selected rules, select the best k rules for each class and calculate average expected accuracy of best k rules of each class. Then choose the class with highest expected accuracy as the predicted class label.

2.2 Lazy Associative Classification Techniques

Associative classification algorithms are a form of eager laerner. Eager learners, when given a set of training tuples, will construct a generalization model before recieving new tuples to classify. We can think of the learned model as being ready and eager to classify previously unseen tuples.

Imagine a contrasting lazy approach, in which the learner instead waits until the last minute before doing any model construction in order to classify a given test tuple. That is, when given a training tuple, a lazy learner simply stores it (or does only a little minor processing) and waits until it is given a test tuple. Only when it sees the test tuple does it perform generalization in order to classify the tuple based on its similarity to the stored training tuples. Unlike eager learning methods, lazy learners do less work when a training tuple is presented and more work when making a classification or prediction. Because lazy learners store the training tuples or "instances," they are also referred to as instance-based learners, even though all learning is essentially based on instances[12].

Associative classifiers prune CARs based on support or confidence or correlation measurement. Many effective CARs may be pruned during this process. CARs that will misclassify unseen tuples only need to be pruned. For exploiting this idea, lazy approach of pruning is introduced. In classification phase, after seeing the test tuples lazy pruning approach determines the CARs that will misclassify and pruned. Some other approaches have also been introduced. We will explain LAC[5] algorithm in detail in the next subsection.

2.2.1 LAC

LAC produces CARs to each specific test instances. We will describe the LAC using an example.

Example 2.5

In the Table 2.7, a training dataset is given. LAC classification process will start when a test tuple is given. Using test tuple LAC projects the dataset D into D_A only on those features appeared in the test instance A. Let consider the last row of the Table 2.7 as the test instance A. Now, the projected database D_A of A is given in the Table 2.8.

No.	Outlook	Temperature	Humidity	Windy	Play
1.	rainy	cool	normal	false	yes
2.	rainy	cool	normal	true	no
3.	overcast	hot	high	false	yes
4.	sunny	mild	high	false	no
5.	rainy	cool	normal	false	yes
6.	sunny	cool	normal	false	yes
7.	rainy	cool	normal	false	yes
8.	sunny	hot	normal	false	yes
9.	overcast	mild	high	true	yes
10.	sunny	mild	high	true	no
Test	overcast	hot	low	true	???

Table 2.7: Training Dataset

No.	Outlook	Temperature	Humidity	Windy	Play
1.	-	-	-	true	no
2.	overcast	hot	-	-	yes
3.	-	hot	-	-	yes
4.	overcast	-	-	true	yes
5.	-	-	-	true	no

Table 2.8: Projected Training Dataset D_A

Suppose minimum support is set to 40% for D. Projected dataset has less instances only consisting of the appeared attributes in the test instance. Therefore, CARs not frequent in D may be frequent in D_A. A CAR for the dataset D_A need to have support as half of 40%, because the number of instances of D_A is half than D. LAC founds two CARs inD_A as :

1. {outlook=overcast→play=yes}
2. {temperature=hot→play=yes}

LAC not only predict the correct class but also it is very simple in implementation than eager associative classifiers. The sudocode of the LAC is given in Algorithm 2.5.

Algorithm 2.5 : Lazy Associative Classification Algorithm[5]

Input : let D be the set of all n training instances

Output : let T be the set of all m test instances

```
procedure LAC($D, T$)
    for each $t_i \in T$ do
        let $D_{t_i}$ be the projection of D on features only from $t_i$
        let $C^l_{t_i}$ be the set of rules X $\rightarrow$ c mined from $D_{t_i}$
        sort $C^l_{t_i}$ according to information gain
        pick the first rule X $\rightarrow$ c $\in C^l_{t_i}$, and predict class c
    end for
end procedure
```

Lazy approach in associative classification is becoming important research area. Efficient mining of high quality CARs is too important task to predict class label more accurately. Thus different techniques are involved in lazy associative classification.

2.3 Summary

In this chapter, we discussed the basics of associative classification algorithm. Some very renowned associative classification algorithms were discussed briefly for understanding of the classification process. Different parts of an associative classification were explained with examples. We got familiar with some of the common terminologies for associative classification. Finally, an associative classification using the lazy learning approach were described briefly.

Chapter 3

Our Proposed Approach

Although associative classifiers started a new way to the classification techniques but they have some significant drawbacks. Some lazy associative classiers tried to overcome these drawbacks but they still use the traditional classification parameters taken from previous associative classifiers. A new thought in classification parameters is required to overcome the drawbacks. These drawbacks are listed below

1. Associative classifiers generate rules from training tuples which are unable to cope with varying nature of all the upcoming test tuples. This affects performance.It is better to generate rules from test tuples.
2. For minimum support and confidence value includes some unnecessary small rules and excludes rather potential long rules.
3. They present feature selection bias as the same dataset is used for both support, confidence calculation and rule generation.
4. Generate redundant rules by not including a minimum number of participating attributes in a rule.
5. Affects classification by not including the number of participating attributes in to rule evaluation method.

Above limitations of the traditional classifiers motivated us to propose a new approach that deals with these problems.

3.1 Prelimineries

Before describing our proposal some necessary terms need to be defined.

Definition(literal) 3.1.1. *A literal is an attribute-value pair (A_i,v) , in which A_i is the attribute and v is the value. A tuple t(t_1,t_2,t_3,...,t_n) satisfies literal p(A_i,v) if and only if value of t_i=v and attribute of t_i=A_i.*

For example, from Table 3.2.1 (age,young) is a literal.

Definition(rule) 3.1.2. *A rule R is a collection of literals with an associated class label*
"$p_1 \wedge p_2 \wedge p_3 \wedge ... \wedge p_m \rightarrow c$" here p_i is literal and c is the class label. Tuple t(t_1,t_2,t_3,...,t_n) positively satisfies rule R if and only if t satisfies every literal p_i and class label c. Tuple t(t_1,t_2,t_3,...,t_n) negatively satisfies rule R if and only if t satisfies every literal p_i but not class label c.

For example, from Table 3.2.1 (age,youth) → (buys-computer,no) is a rule.

Definition(positive coverage) 3.1.3. *Let D be the training dataset . Positive coverage of a rule*
R=($p_1 \wedge p_2 \wedge p_3 \wedge ... \wedge p_m \rightarrow c$) is the maximum cardinality $|A|$ where $A \subset D$ and for each tuple $A_i \in A, A_i$ positively satisfies R.

For example, from Table 3.2.1 rule (age,youth) → (buys-computer,no) has three satisfying items RID(1), RID(2), RID(11). So its positive coverage is three.

Definition(negative coverage) 3.1.4. *Let D be the training dataset . Negative coverage of a rule*
R=($p_1 \wedge p_2 \wedge p_3 \wedge ... \wedge p_m \rightarrow c$) is the maximum cardinality $|A|$ where $A \subset D$ and for each tuple $A_i \in A, A_i$ negatively satisfies R.

For example, from Table 3.2.1 rule (age,youth) → (buys-computer,no) has one non-satisfying items RID(9). So its negative coverage is one.

3.2 LACF: Our Proposed Algorithm

We are proposing a new algorithm LACF(**L**azy **A**ssociative **C**lassifier based on maximal **F**eature Selection) which is a lazy learning approach in association rule mining that will generatate rules from test tuple and classify using back computation. We are also proposing a new rule evalution function to choose from generated rules.

3.2.1 Descriptive Example

For explaining LACF, we will describe it's functionality based on the AllElectronics customer database given in Table 3.1. Let RID(14) is our test

RID	age	income	student	credit-rating	Class:buys-computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle-aged	high	no	fair	yes
4	senior	medium	no	fair	no
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle-aged	low	yes	excellent	yes
8	senior	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	excellent	yes
11	youth	medium	no	excellent	no
12	middle-aged	medium	no	excellent	yes
13	senior	high	no	excellent	yes
14	senior	medium	no	excellent	no

Table 3.1: Class labeled customer database from AllElectronics

tuple and rest are training tuples.We will describe this example in three basic parts namely rule generation , rule pruning and classification.

3.2.1.1 Rule generation

Let consider the following tuple as test case
T={(age=senior),(income=medium),(student=no),(credit-rating=excellent)}

Potential rules to classify T should come from $\mathcal{P}\{T\}$. So there will be $2^4 = 16$ members . Our class label number is 2 . So in total 16×2 or 32 rules will be generated. These rules are

1. {age=senior → buys-computer=yes}
2. {age=senior → buys-computer=no}
3. {age=senior,income=medium → buys-computer=yes }
4. {age=senior,income=medium → buys-computer=no}
5. {age=senior,income=medium,student=no → buys-computer=yes}
6. {age=senior,income=medium,student=no → buys-computer=no}
   .... and more

3.2.1.2 Rule pruning

From the above process we can have $2^4 \times 2$ or 32 rules. All of these are not necessary for classification . Rule with more literals has less feature subset selection bias[27] than rule with less literals. We choose $l=\frac{3}{4}$ and discard the rules whose number of literals is less than $l \times |T|$ or 3.Choosing optimal l is based on experiments. After pruning we will have only $(\binom{4}{3} + \binom{4}{4}) \times 2$ or 10 rules.These are the set LACFRules we need

1. {age=senior,income=medium,student=no → buys-computer=yes }
2. {age=senior,income=medium,student=no → buys-computer=no}
3. {age=senior,income=medium,credit-rating=excellent → buys-computer=yes}
4. {age=senior,income=medium,credit-rating=excellent → buys-computer=no}
5. {age=senior,student=no,credit-rating=excellent → buys-computer=yes}
6. {age=senior,income=no,credit-rating=excellent → buys-computer=no}
7. {student=no,income=medium,credit-rating=excellent→ buys-computer=yes}
8. {student=no,income=medium,credit-rating=excellent → buys-computer=no}
9. {age=senior,income=medium,student=no,credit-rating=excellent → buys-computer=yes}
10. {age=senior,income=medium,student=no,credit-rating=excellent → buys-computer=no}

3.2.1.3 Classification

Now those 10 rules will go through our rule evaluation function and rule with the best value will be chosen to classify.The Rule evaluation function is a function consists of positive coverage,negative coverage and number of literals. It is defined as

$$\textbf{LACFEval}(\text{R}) = \frac{positiveR}{positiveR+negativeR} \times positiveR \times R.length$$

where,
$positiveR$ = positive coverage of R
$negativeR$ = negative coverage of R
For example rule number (2) will have ,

$$\textbf{LACFEval}(\text{R2}) = \frac{2}{2+0} \times 2 \times 3{=}6$$

Thus according to rule evaluation function 10 rules above have values given in Table 3.2. Table 3.2 shows that rule number 2 has the highest value .

Rule Number	Value	Class:buys-computer
1	0	yes
2	6	no
3	3	yes
4	0	no
5	3	yes
6	0	no
7	1.5	yes
8	1.5	no
9	0	yes
10	0	no

Table 3.2: Reffered class label and values for previous rules

It refers to Class-label:(buys-computer=no) so (buys-computer=no) is the desired result.

3.3 Algorithm

Input
A rule R
Output
A real number value *val*

```
function LACFEval(R)
    $positiveR \leftarrow$ positive coverage of R
    $negativeR \leftarrow$ negative coverage of R
    if $positiveR + negativeR$=0 then
        $val \leftarrow 0$
    else
        $val \leftarrow \frac{positiveR}{positiveR+negativeR} \times positiveR \times |R|$
    end if
    return $val$
end function
```

Input
Test tuple T=$(p_1, p_2, ..., p_k)$,class labels C=$C_1, C_2, ..., C_m$ threshold value l
Output
A class label C_i

```
function LACF($T$,$l$,$C$)
    $max \leftarrow 0$
    $c \leftarrow default - classlabel$
    $ruleset \leftarrow \mathcal{P}(T)$
    remove from ruleset all members whose length$<l \times |T|$
    for each $C_i$ in C do
        for each $Q$ in ruleset do
            rule R $\leftarrow (Q, C_i)$
            $temp \leftarrow LACFEval(R)$
            if $temp$>$max$ then
                $max \leftarrow temp$
                $c \leftarrow C_i$
            end if
        end for
    end for
    return $c$
end function
```

3.3.1 Steps Of Algorithm

3.3.1.1 Rule Generation

Let T=$(p_1, p_2, p_3, ..., p_k)$ be a test tuple and class labels are C=$(c_1, c_2, .., c_m)$. Tuple T should be classified better with the rule R=(Q$\rightarrow C_i$) if Q$\subset$T,Q is a set of literals. So classifying rules should be chosen from power set $\mathcal{P}(T)$

thus always Q$\subset$T and rules are potential. Generating rules from training tuples lacks this important property. So we generate rules from $\mathcal{P}\,(T)$.

For example, from Table 3.2.1 say a tuple
T={ (age=middle-aged), (income=medium), (student=yes), (credit-rating=excellent)}
needs to be classified. Like traditional associative classifiers two candidates for class association rules could be.

1. {(student=no) $\rightarrow$ (buys-computer=yes)} [support= 9]
2. {(student=yes) $\rightarrow$ (buys-computer=yes) } [support= 5]

if minimum support= 6 then rule number two will be discarded from class association rules. Rule number one can be useful for any other tuple but it is totally irrelevant for tuple T. This same scenerio is possible for minimum confidence value. So it is better to generate rules from each test tuple.

3.3.1.2 Rule Pruning

Power set $\mathcal{P}\,(T)$ can generate huge number of rules mathematically in exponential order most of which are less potential. We prune the rules upon literal number . Because it gurantees less feature selection bias.

Feature Subset Selection Bias[27]

Using class conditional probability our goal can be described as P(T|C_i).Expected value of P(T|C_i) is E[P(T|C_i)]. If we choose a subset from T like Q and classify T using Q then the expected value is E[P(T|C_i)|selected subset Q].The estimated probability values tend to get biased. This bias is conditioned upon feature subset Q. Thus
bias=E[P(T|C_i)|selected subset Q] - E[P(T|C_i)]
bias is zero if Q=T and increases with decreasing literal or attribute numbers.

In feature selection, we select features that will help improve classification accuracy. Subsequently, when the same dataset is used for estimating the classifier parameters, probability estimates tend to be biased.

Similarly, when we select a rule that has a subset of test tuple attributes, we select such a rule that help improve classification accuracy. Subsequently, when we evaluate or give a priority value to this rule from the same dataset, the class label estimation tend to be biased. So, the less attributes are discarded from test tuple to form a rule that can classify the tuple best, the less bias exists. An ideal non-biased situation will be to classify a tuple using all of its attribute values.

For example, from Table 3.2.1 say a tuple
T={ (age=middle-aged), (income=medium), (student=no), (credit-rating=excellent)} needs to be classified. Let us consider three potential rules

1. {(student=no) → (buys-computer=no)} [support= 9 and confidence= 0.67]
2. {(student=no) → (buys-computer=yes)}[support= 9 and confidence= 0.33]
3. {(student=no),(age=middle-aged)} → (buys-computer=yes)}[support= 2]

if minimum support= 3 then rule number three will be discarded. Among the first two rules, rule number one has higher confidence value. So by traditional approach desired class label is (buys-computer=no). Now a very important property that is missing here is the contributing features. If we did not discarded rule number three, we can see that it has 100% confidence towards class label (buys-computer=yes). Rule number three has more contributing features and exhibits more information about the tuple. Rule number one is strongly biased. Rule number three has less bias then rule number one because it has more contributing features. If rule number one is the best rule for this tuple then it has to be very high in support and confidence value than rule number three to prove that the other feature is not so important, which is the task of algorithm before preparing training sets. To design a classification algorithm, we should assume that all the features are important. This example shows that number of features should also be considered and a minimum contributing feature number should be introduced.

Selection of optimal number of literals

Now, it is not likely that all the training datasets will contain an ideal rule for each test tuple. Training datasets with more items are better to find such an ideal rule than smaller datasets. To cope with different datasets, we need to find such a rule against each tuple that minimizes the *bias* for that particular dataset. We need a minimum allowable rule length close to the ideal rule. To find this value, we propose a variable l that represents the fractional value with ideal rule length. Using l our classification rules will look like,

LACFRules = $\{$ X$\rightarrow C_i |$ X$\in \mathcal{P}(T) and C_i \in Cand$ $|X| > l \times |X|$ $\}$

l is a fractional value ,$0 < l \leq 1$. Selection of optimal value for l is experimental and it can vary for different types of datasets.

3.3.1.3 Classification

After selecting the most potential rules, we give each rule a priority value. This rule evaluation method considers rule length to overcome feature selection bias. Among the potential rules, longer ones get some extra priority. If a rule has to outperform a longer rule, it requires to be very strong in its positive coverage and percentage of positive coverage value. By allowing this oppurtunity, we are giving a chance to relatively smaller but very potential rules. This is useful when there are some noisy attributes in the dataset or comparatively irrelavent attribute values exist. Thus our rule evaluation method is formulated as,

Rule Evaluation Method

Let,
R=(X$\rightarrow C_i$) $\in$ LACFRules
positiveR=positive coverage of rule R
negativeR=negative coverage of rule R
Rule evaluation method ,

$$\textbf{LACFEval}(\text{R}) = \frac{positiveR}{positiveR+negativeR} \times positiveR \times |X|$$

For each R=(X$\rightarrow C_i$) $\in$ LACFRules is evaluated using LACFEval and R=(X$\rightarrow C_i$) with the highest value is chosen for classification where C_i is the desired class label.

Chapter 4

Performance Evaluation

In this chapter , we will present the performance of our proposed algorithm.We will focus on several scenarios to compare our proposed algorithm(LACF) with C4.5,RIPPER,CBA,CMAR,CPAR.We will also consider the size of the training data sets with classification accuracy.Finally, we will discuss about some tuning of our used constants to improve performance. All the datasets used here are collected from UCI Machine Learning Repository[2]. Memory space concern and cost analysis will also be briefly discussed at the end of this chapter.

4.1 Performance Variation With Different Dataset Parameters

In this section, we will see accuracy variation with different dataset parameters like dataset record number and attribute number. We will take five different datasets with different number of attributes and records.

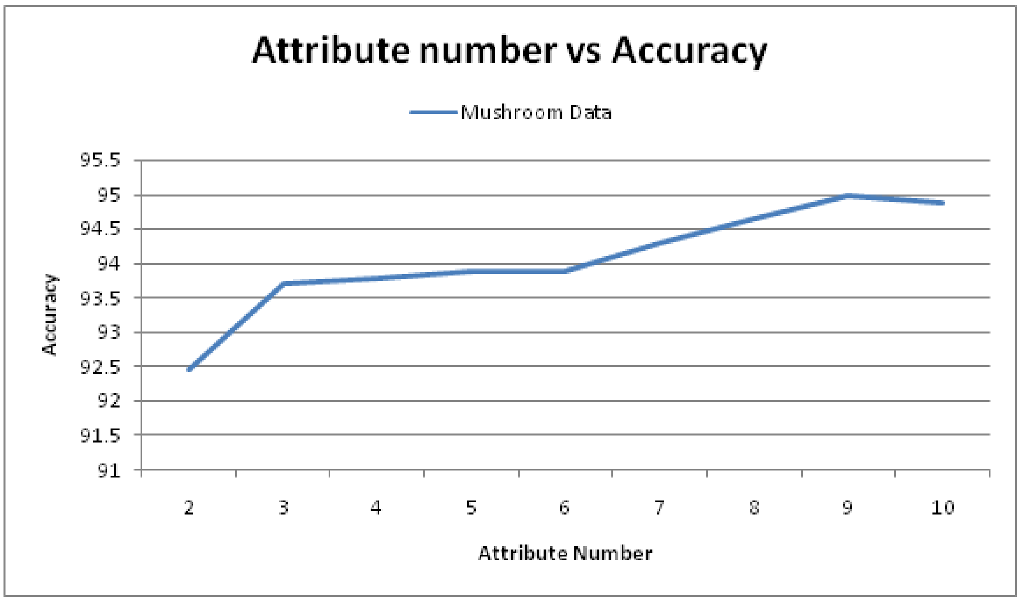

Figure 4.1: Number of Attributes and Accuracy Relation

Figure 4.1 shows that accuracy increases with increasing number of attributes. It also varies with number of records , so few accuracy points are below their expected value. Otherwise overall proportional relation holds.

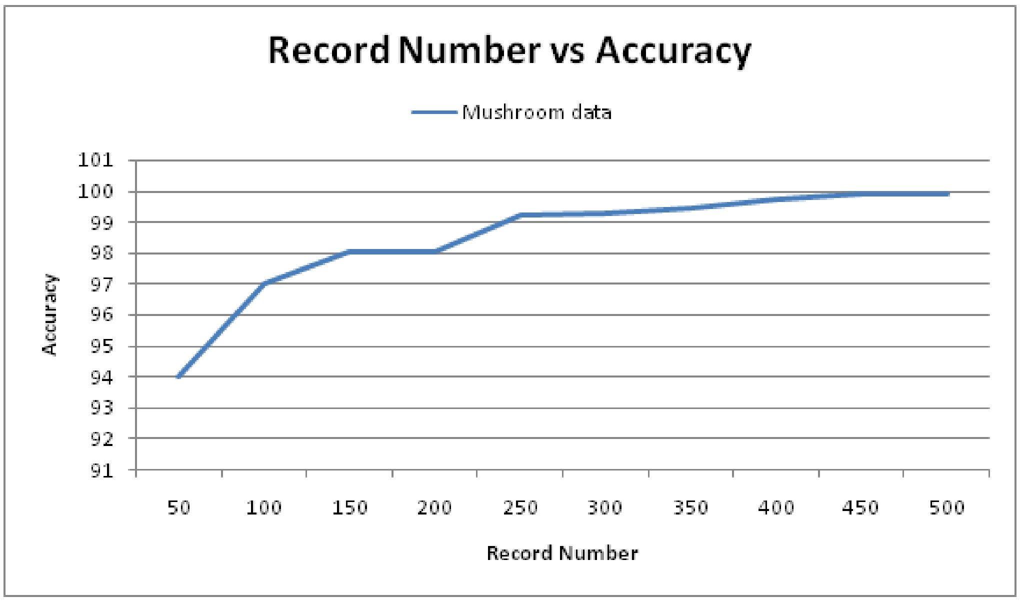

Figure 4.2: Number of Records and Accuracy Relation

Figure 4.2 shows the mushroom dataset with different datset size.Here, accuracy increases with increasing number of records. It also varies with number of attributes , so few accuracy points are below their expected value. Otherwise overall proportional relation holds.

4.2 Comparison of LACF and Other Associative Classifiers

In this section, we will try to show a number of scenarios and present some experimental results using charts to show the comparison among the LACF algorithm ,recent associative classifiers and other classic classifiers namely C4.5[24], RIPPER(RIP.)[30], CBA[7], CMAR[28], CPAR[31].We used some popular datasets found in machine learning repository which were used to evaluate previous classifiers.This will make it easy to compare between LACF and other classifiers.We used 10 fold cross validation method to measure accuracy.In 10 fold cross validation process , dataset is partitioned in to 10 pieces where member of each piece is chosen randomly and exclusively.Then result of all training and test dataset pairs are averaged for final result.

Dataset	C4.5	RIPPER	CBA	CMAR	CPAR	LACF
tic-tac	99.4	98	99.6	99.2	98.6	98.74
led7	73.5	69.7	71.9	72.5	73.6	72.63
zoo	92.2	88.1	96.8	97.1	95.1	95
vote	88.27	87.35	87.39	-	-	90.47
mushroom	99.96	99.96	98.92	-	98.8	100
Average	90.66	88.62	90.9	89.6	91.5	91.4

Table 4.1: Accuracy: C4.5, RIPPER, CBA, CMAR, CPAR, LACF.

Table 4.1 shows an experimental result. Here some popular datasets are given to show the accuracy percentage for different classifiers.It shows that some classifiers are well below LACF and some gives better performance for particular dataset.Only one classifiier has better average accuracy than LACF.But LACF has significant accuracy in every dataset.

Dataset	#att	#rec	C4.5	RIP.	CBA	CMAR	CPAR	LACF
tic-tac	9	958	99.4	98	99.6	99.2	98.6	98.74
led7	7	3200	73.5	69.7	71.9	72.5	73.6	72.63
zoo	18	101	92.2	88.1	96.8	97.1	95.1	95
vote	16	435	88.27	87.35	87.39	-	-	90.47
mushroom	22	8124	99.96	99.96	98.92	-	98.8	100
Average	-	-	90.66	88.62	90.9	89.6	91.5	91.4

Table 4.2: Accuracy with number of records and attributes

Table 4.2 shows an experimental result . Here it is noticable that attribute number and dataset size effects LACF accuracy.For better result there should be sufficient attributes and training data.It is very much overwhelming that LACF has hundred percent accuracy for mushroom dataset.It is an outstanding result which outperforms every other classifiers.Mushroom data set has twenty two attributes and over eight thousand data items . So LACF performs much better as it was expected for sufficient attributes and data items.

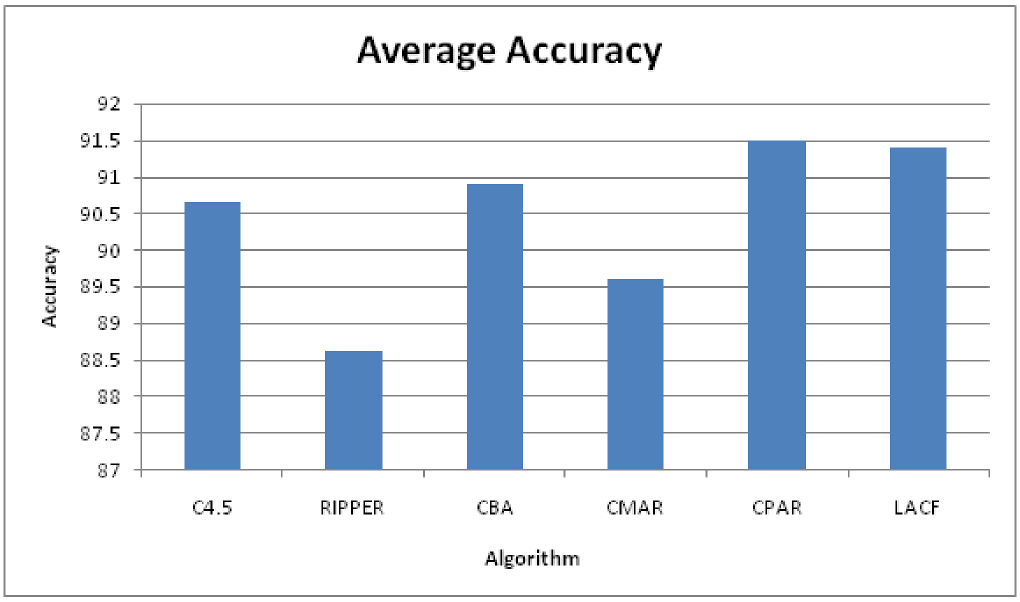

Figure 4.3: Average accuracy of classification techniques

Figure 4.3 shows a graphical view for different classifier's average accuracy . LACF has only one classifier which has better average accuracy.It is expected that fine tuning of l will improve its average accuracy.

4.3 Performance for Different Values of l

l defines the minimum rule length required to take part in classifying a test tuple.So l plays an important role in classification accuracy.Theoretically performance should improve with increasing l value and it is seen also in experiments. The experiment data below proves its significance Table 4.3 shows that accuracy has an important relation with constant l . Two datasets tic-tac-toe and car evaluation were evaluated for different l values

-	accuracy for different l-value							
dataset	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
tic-tac	63.16	63.16	73.7	84.2	93.7	97.9	98.7	-
car	72.67	72.67	72.67	73.26	80.82	80.82	88.95	88.95

Table 4.3: Accuracy vs ***l***-value relation for different datasets

between 0 and 1. Accuracy increases gradually with increasing l value for both the datasets. Finally l reaches a saturation point from which further improvement is not possible.This point is the optimal value for l and it varies from dataset to dataset.

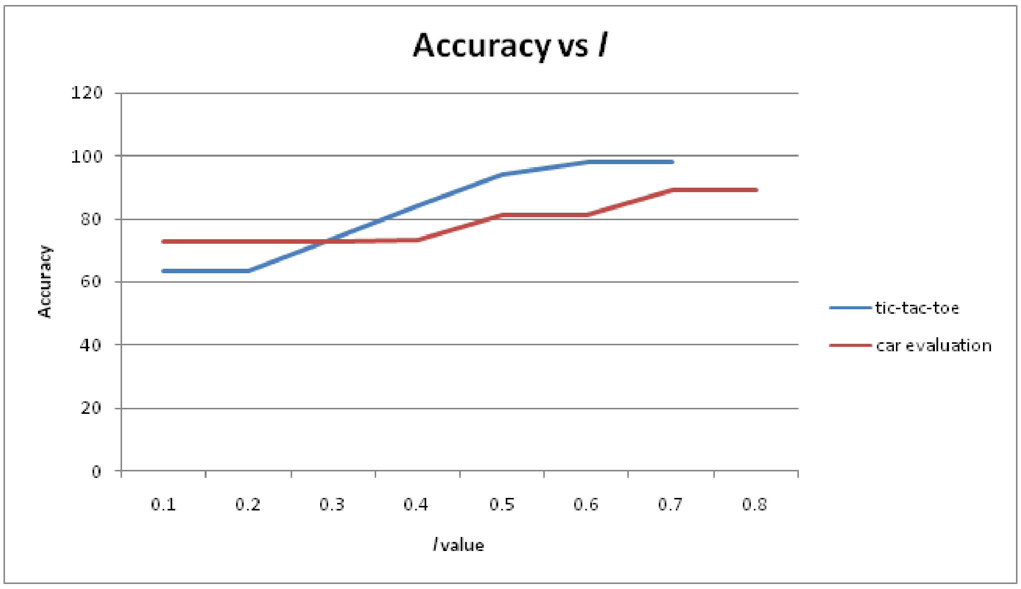

Figure 4.4: Accuracy vs ***l*** value relation for two dataset

Figure 4.4 shows the graphical view of accuracy vs l relation.A gradually increasing slope is noticed in the graph.There are periods when accuracy remains unchanged for a whille but it continues to increase after some short intervals until the l value reaches the saturation point.

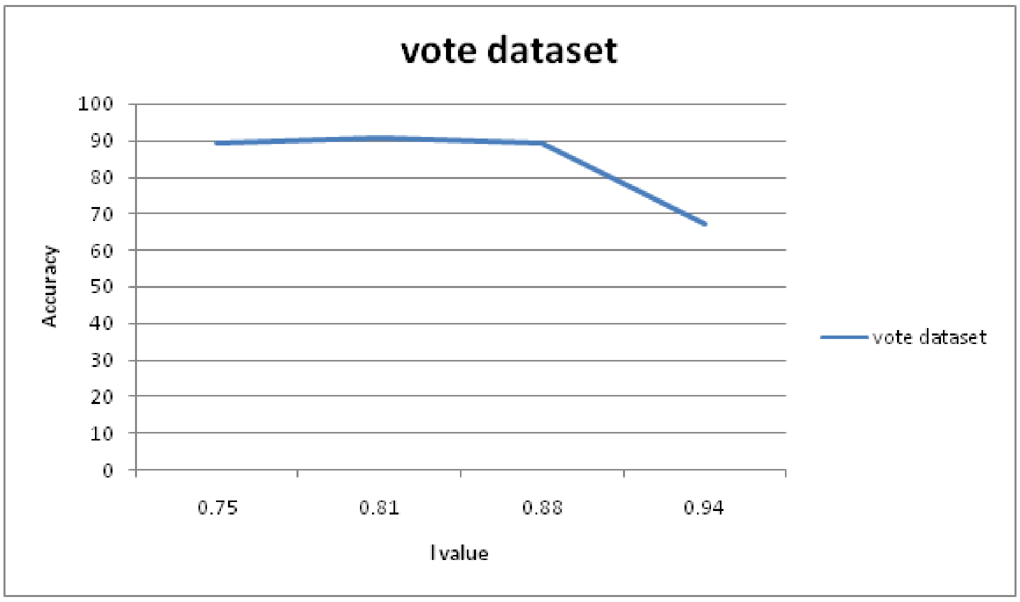

Figure 4.5: ***l*** value over threshold for vote dataset

Figure 4.5 shows the graphical view of l value over threshold for house vote dataset. l value remains steady after reaching optimal value for a while and after that accuracy decreases drastically instead of increasing.

4.4 Cost Analysis

LACF is a lazy learning approach . So, it requires less memory than other classifiers.It does not need any pregenerated rules to store.

We need to generate classifying rules from test tuples. So, before testing no time calculation is needed. To classify a tuple, a subset of the power set of tuple attributes is generated as classification rules. This may increase the classification time. Classification time also increases with dataset size. This may be reduced by using some precalculation and efficient use of memory.Choosing an optimal l value improves classification time. Because an optimal l value discards most of the members from eligible power set .

4.5 Summary

Our algorithm performs better for sufficient attributes and data items. Moreover it indicates a very important property of classification techniques, the role of partcipating attributes.It also outperforms most of the recent associative classifiers. At the end, we can say that with the right choice of l, LACF is one of the most accurate classifier in classification field.

Chapter 5

Conclusion

In this research work, we have developed an idea to improve the existing associative classifiers using lazy learning approach. Existing associative classifiers use pregenerated association rules to classify an unlabeled tuple and do not consider the number of participating attributes in classification rules. More participating attributes are likely to predict correct class labels than less participating attributes. Association rules generated upon training dataset only, may include useless rules as well as exclude potential rules for a test tuple. To overcome these problems we implemented an advanced idea over the existing associative classifiers.

5.1 Research Summary

In our algorithm, we generated rules from test tuple. This approach ensures to include all the potential rules and exclude any redundant rule. Though this may look like that a huge number of rules will be generated for every test tuple but rule pruning method of this algorithm keeps this number within reach.

In the rule pruning section, we used a constant named l to limit the classificaiton rules. The value of l can be between 0 and 1. This value defines the fraction of minimum rule length. Finding an optimal l value will ensure a short classification rule set and best accuracy.

We also proposed a new rule evaluation method. This method emphasis on rule length or participating attributes besides its confidence and positive coverage. This makes it more powerful than other rule ranking methods. Because, here a small rule must be very strong in confidence and positive coverage to outperform a long rule. This ensures less overall bias of some attributes and increases reliablity.

5.2 Scope of Future Studies

Some points can be made to improve our algorithm. First, our algorihm generates power set of the test tuple attributes and scans database for every test tuple. By effecient memory usage techniques we can reduce database scan to once and make a less time consuming power set generation algorithm.

Second, to find the optimal l value our algorithm does not propose any efficient technique. It requires continuous experiments to reach the optimal l value. It should be possible to find a formula to obtain an optimal l value.

Bibliography

[1] A. An and N. Cercone. Discretization of Continuous Attributes for Learning Classification Rules. In *Proceedings of the Third Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining*, PAKDD '99, pages 509–514, 1999. [cited at p. 10]

[2] A. Frank and A. Asuncion. UCI machine learning repository, 2010. [cited at p. 39]

[3] A. Savasere, E. Omiecinski and S. Navathe. Mining for Strong Negative Associations in a Large Database of Customer Transactions. In *Proceedings of the Fourteenth International Conference on Data Engineering*, ICDE '98, pages 494–502, 1998. [cited at p. 2, 3]

[4] A. Veloso, W. Meira Jr. and M. Zaki. Calibrated lazy associative classification. In *Proceedings of the 23rd Brazilian symposium on Databases*, SBBD '08, pages 135–149, 2008. [cited at p. 2]

[5] A. Veloso, W. Meira Jr. and Mohammed J. Zaki. Lazy Associative Classification. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM, pages 645–654, 2006. [cited at p. 2, 4, 8, 26, 27]

[6] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, Chile, September 1994. [cited at p. 2, 7, 15]

[7] W. Hsu B. Liu and Y. Ma. Integrating classification and association rule mining. In *In Knowledge Discovery and Data Mining*, pages 80–86, 1998. [cited at p. 2, 7, 9, 10, 11, 13, 41]

[8] G. Dong and J. Li. Effciient Mining of Emerging Patterns: Discovering Trends and Differences. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge discovery and Data Mining*, pages 43–52, 1999. [cited at p. 3]

[9] E. Baralis, S. Chiusano and P. Garza. A lazy approach to associative classification. *IEEE Transactions on Knowledge and Data Engineering*, vol. 20(no. 2). [cited at p. 2, 4]

[10] U. Fayyad and K. Irani. Multi interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1022–1027, 1993. [cited at p. 10]

[11] L. Wong G. Dong, X. Zhang and J. Li. CAEP: Classifcation by aggregating emerging patterns. In *Proceedings International Conference on Discovery Science*, pages 30–49, 1999. [cited at p. 2, 3]

[12] Jiwei Han and Micheline Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers, San Francisco, 2006. [cited at p. 9, 26]

[13] J. Dougherty and R. Kohavi and M. Sahami. Supervised and Unsupervised Discretization of Continuous Features. In *Machine Learning: Proceedings of the 12'th International Conference*, pages 194–202. Morgan Kaufmann, 1995. [cited at p. 10]

[14] J. Han, J. Pei and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD '00, pages 1–12, 2000. [cited at p. 2, 3, 8, 15, 17]

[15] J. Han, J. Pei, Y. Yin and R. Mao. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Mining Knowledge Discovery*, vol. 8(no. 1):53–87, January 2004. [cited at p. 2]

[16] J. Li and G. Dong and K. Ramamohanarao. Instance-Based Classification by Emerging Patterns. In *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 191–200, 2000. [cited at p. 3]

[17] J. R. Quinlan and R. M. Cameron-jones. FOIL: A Midterm Report. In *In Proceedings of the European Conference on Machine Learning*, pages 3–20. Springer-Verlag, 1993. [cited at p. 3, 22, 23]

[18] R. J. Bayardo Jr. Brute-Force Mining of High Confidence Classification Rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 123–126, 1997. [cited at p. 2]

[19] R. J. Bayardo Jr. and R. Agrawal. Mining The Most Interesting Rules. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 145–154, 1999. [cited at p. 2]

[20] Maria-Luiza Antonie and Osmar R. Zaïane. An associative classifier based on positive and negative rules. In *Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, DMKD '04, pages 64–69, 2004. [cited at p. 3]

[21] Maria-Luiza Antonie and Osmar R. Zaïane. Mining positive and negative association rules: an approach for confined rules. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, PKDD '04, pages 27–38, 2004. [cited at p. 2, 3]

[22] P. Leng and F. Coenen . The effect of threshold values on association rule based classification accuracy. *Journal of Data and Knowledge Engineering*, vol.60:360, 2007. [cited at p. 2]

[23] Pang-Ning Tan, V. Kumar and J. Srivastava. Selecting the right interestingness measure for association patterns. In *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 32–41, New York, NY, USA, 2002. [cited at p. 2]

[24] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. [cited at p. 22, 41]

[25] R. C. Agarwal, C. C. Aggarwal and V. V. V. Prasad . A tree projection algorithm for generation of frequent item sets. *Journal of Parallel and Distributed Computing*, vol. 61(no. 3):350–371, March 2001. [cited at p. 2]

[26] S. P. Syed Ibrahim, K. R. Chandran, and C. J. Kabila Kanthasamy. LACI: Lazy Associative Classification using Information Gain. *International Journal of Engineering and Technology*, vol. 4 no. 1:1–6, 2012. [cited at p. 2, 4]

[27] Surendra K. Singhi and Huan Liu. Feature subset selection bias for classification learning. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 849–856, New York, NY, USA, 2006. [cited at p. 32, 35]

[28] J. Han W. Li and J. Pei. CMAR:Accurate and Efficient classification based on multiple class-association rules. In *Proceedings of the International Conference on Data Mining*, page 369376, 2001. [cited at p. 2, 3, 8, 14, 15, 17, 19, 20, 41]

[29] J. Wang and G. Karypis. HARMONY:efficiently mining the best rules for classification. In *Proceedings of the SIAM International Conference on Data Mining*, 2005. [cited at p. 2]

[30] William W. Cohen. Fast Effective Rule Induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995. [cited at p. 41]

[31] X. Yin and J. Han. CPAR: Classification based on predictive association rules. In *Proceedings of the 3rd SIAM International Conference on Data Mining*, 2003. [cited at p. 2, 3, 8, 23, 24, 25, 41]

[32] L. Yao Z. Hao, X. Wang and Y. Zhang. ICPAR: Improved Classification based on Predictive Association Rules. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics Mining*, San Antonio, TX, USA, October 2009. [cited at p. 2]