2006-05-03

Association Rule Based Classification

Senthil Kumar Palanisamy
Worcester Polytechnic Institute

Association Rule Based Classification

by

Senthil K. Palanisamy

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

May 2006

APPROVED:

Professor Carolina Ruiz, Thesis Advisor

Professor Matthew Ward, Thesis Reader

Professor Michael Gennert, Head of Department

Abstract

In this thesis, we focused on the construction of classification models based on association rules. Although association rules have been predominantly used for data exploration and description, the interest in using them for prediction has rapidly increased in the data mining community. In order to mine only rules that can be used for classification, we modified the well known association rule mining algorithm Apriori to handle user-defined input constraints. We considered constraints that require the presence/absence of particular items, or that limit the number of items, in the antecedents and/or the consequents of the rules. We developed a characterization of those itemsets that will potentially form rules that satisfy the given constraints. This characterization allows us to prune during itemset construction itemsets such that neither they nor any of their supersets will form valid rules. This improves the time performance of itemset construction. Using this characterization, we implemented a classification system based on association rules and compared the performance of several model construction methods, including CBA, and several model deployment modes to make predictions. Although the data mining community has dealt only with the classification of single-valued attributes, there are several domains in which the classification target is set-valued. Hence, we enhanced our classification system with a novel approach to handle the prediction of set-valued class attributes. Since the traditional classification accuracy measure is inappropriate in this context, we developed an evaluation method for set-valued classification based on the E-Measure. Furthermore, we enhanced our algorithm by not relying on the typical support/confidence framework, and instead mining for the best possible rules above a user-defined minimum confidence and within a desired

range for the number of rules. This avoids long mining times that might produce large collections of rules with low predictive power. For this purpose, we developed a heuristic function to determine an initial minimum support and then adjusted it using a binary search strategy until a number of rules within the given range was obtained. We implemented all of our techniques described above in WEKA, an open source suite of machine learning algorithms. We used several datasets from the UCI Machine Learning Repository to test and evaluate our techniques.

Acknowledgement

I would like to thank Prof. Carolina Ruiz for her guidance and encouragement in completing the thesis. This would not have been possible if not for her belief in me. I am also grateful to Prof. Matthew Ward for his comments in shaping the thesis. I would also like to thank fellow students of Knowledge Discovery and Data Mining Group (KDDRG) at WPI for their insights and advice when I needed. I cannot thank enough my wife, Elisabeth, for all her support and encouragement in completing this work. Finally, I would like to dedicate this work to my parents who have been there all along to support me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Knowledge Discovery and Data Mining (KDD) is playing an important role in extracting knowledge in this era of data overflow. KDD consists of many methods and techniques that can be applied to different data to extract knowledge. Some of the methods include association, classification, and clustering. In this work, we primarily focus on association and classification.

Association rule mining is the discovery of association relationships among a set of items in a dataset. Association rule mining has become an important data mining technique due to the descriptive and easily understandable nature of the rules. Although association rule mining was introduced to extract associations from market basket data [AIS93], it has proved useful in many other domains (e.g. microarray data analysis, recommender systems, and network intrusion detection). In the domain of market basket analysis, data consists of transactions where each is a set of items purchased by a customer. A common way of measuring the usefulness of association rules is to use the support-confidence framework introduced by [AIS93].

Support of a rule is the percentage of transactions that carry all the items in the rule, and the confidence is the percentage of the transactions that carry all the items in the rule among those transactions that carry the items in the antecedent of the rule.

The problem of association rule mining can be stated as: Given a dataset of transactions, a threshold support (*minsupport*), and a threshold confidence (*minconfidence*); Generate all association rules from the set of transactions that have support greater than or equal to *minsupport* and confidence greater than or equal to *minconfidence*.

Classification is another method of data mining. Classification can be defined as learning a function that maps (classifies) a data instance into one of several predefined class labels [Mit97]. The data from which a classification function or model is learned is known as the training set. A separate testing set is used to test the classifying ability of the learned model or function. Examples of classification models include decision trees, Bayesian models, and neural nets. When classification models are constructed from rules, often they are represented as a decision list (a list of rules where the order of rules corresponds to the significance of the rules). Classification rules are of the form P $\rightarrow$ c, where P is a pattern in the training data and c is a predefined class label (target).

As part of this thesis, we study and compare different ways of building models or classifiers from association rules. Given that association rules are descriptive in nature, they are useful in learning about relationships in the data. The learned relationships can be helpful in analyzing the domain. But usefulness of the rules can be further extended if predictive models can be extracted from the rules. Given that the number of rules produced is a function of the minsupport and the minconfidence thresholds, the challenge is to generate an appropriate number of rules that can be

useful in developing predictive models.

Association rule based classification is introduced in [LHM98]. They propose an Apriori like algorithm called CBA-RG for generating rules and another algorithm called CBA-CB for building the classifier. The rules generated by CBA-RG are called classification association rules (CARs), as they have a predefined class label or target. From the generated CARs, a subset is selected based on the heuristic criterion that the subset of rules can classify the training set accurately.

Many other classification systems have been built based on association rules [ZAC02] and [YLW01]. In our work, we have implemented an association rule-based classifier system in the WEKA [FW00] framework. WEKA is a data mining system developed at the University of Waikato and has become very popular among the academic community working on data mining. We have chosen to develop this system in WEKA as we realize the usefulness of having such a classifier in the WEKA environment. To generate classification association rules, we make use of the AprioriSetsAndSequences algorithm [Pra04] with some optimizations. AprioriSetsAndSequences is an extended version of the Apriori algorithm that is capable of mining associations from set-valued and temporal datasets. We have optimized the AprioriSetsAndSequences algorithm to generate only itemsets that can potentially yield classification rules that we desire (with the class label as the consequent). More generally, we have adapted the algorithm to generate only rules that satisfy user specified constraints. We achieve this by integrating these constraints into the mining phase so that we can use the constraints to prune itemsets that would not yield rules of the type that the user desires.

We have also extended our classification system to handle set-valued classes. To the best of our knowledge, the problem of multi-class classification has not been studied in the association rule mining domain. We evaluate our system with gene

expression data and compare our results with previous projects which have used this application domain.

In many cases, the number of rules produced by an association rule mining system is either too low or too high to be useful. Therefore, a goal of this thesis is to experiment with techniques to limit the cardinality of the set of association rules to predefined ranges. We hope to achieve this by using an adaptive minimum support approach [LAR02], where the support is modified based on the cardinality of the set of association rules, until the cardinality matches the predefined range.

1.2 Problem Statement

The overall goal of this thesis is to design and develop a classification system that is based on Association Rule Mining in the WEKA environment. Our problem can be further broken down as follows:

- Adapt the Apriori algorithm to generate classification association rules (CARs) efficiently.
- Build Classification Models.
  - Build a framework to generate models from CARs.
  - Provide different modes for deploying a model in order to classify novel instances.
  - Extend the framework to handle set-valued class attributes.
- Implement an adaptive scheme that searches for an optimal minimum support threshold allowing the user to specify only the desired range of rules and the minimum confidence threshold.

As part of our extensions to the classification system to handle set-valued class labels, we propose and discuss two different methods for predicting set-valued attributes. In this regard, we use the notions of E-measure [LG94] and F-measure [LG94] from information theory to compare the predicted set attribute with the actual set attribute.

Chapter 2

Background and Related Work

In this chapter, we introduce association rules, classification, and integrated associative classification. Also, we look at different existing classification systems built upon association rule mining systems.

2.1 Association Rules

Association rule mining was introduced in [AIS93] as a way to find associative patterns from market basket data. The market basket data consist of transactions where a transaction is a set of items purchased by a customer. The motivation for applying this data mining approach on market basket data was to learn about buying patterns and use that information in catalog design, and store layout design. Since then, association rule mining has been studied and applied in many other domains (e.g. credit card fraud, network intrusion detection, genetic data analysis). In every domain, there is a need to analyze data to identify patterns associating different attributes. Association rule mining addresses this need.

Many association rule mining algorithms have been proposed in the data mining literature. Apriori [AS94] and FP-growth [HPY00] are two of them. Apriori uses

the property, all nonempty subsets of an frequent itemset must also be frequent [AS94] to prune the search space. Apriori follows a breadth-first-search strategy while FP-growth follows a depth-first search strategy. Several extensions of the basic association rule mining algorithm have been published. One of them is the CBA-RG algorithm [LHM98], which adapts Apriori to generate classification association rules efficiently. The generated rules are used in CBA-CB [LHM98] to extract a classification model. We have implemented CBA-CB as part of our model building system. Other extensions to Apriori include mining rules with constraints [SVA97a].

2.1.1 Problem Description

The general problem of association rule mining is: Given a data set of transactions where each transaction is a set of items, a minimum support threshold and a minimum confidence threshold, find all the rules in the data set that satisfy the specified support and confidence thresholds. Let I be a set of items, D be a data set containing transactions (i.e, sets of items in I) and t be a transaction. An association rule mined from D will be of a form $X \rightarrow Y$, where $X, Y \subset I$ and $X \cap Y = \emptyset$. The support of the rule is the percentage of transactions in D that contain both X and Y. The confidence is, out of all the transactions that contain X, the percentage that contain Y as well. Confidence of a rule can be computed as $support\{X \cup Y\} \div support\{X\}$. The confidence of a rule measures the strength of the rule (correlation between the antecedent and the consequent) while the support measures the frequency of the antecedent and the consequent together.

2.1.2 Apriori Algorithm

The Apriori algorithm was introduced in [AS94] as a way to generate association rules from market basket data. The Apriori algorithm is a two stage process: A

frequent itemset (itemsets that satisfy minimum support threshold) mining stage and a rule generation stage (rules that satisfy minimum confidence threshold).

In Table 2.1, we show a subset of the contact-lenses data set from the University of California Irvine (UCI) Machine Learning Repository [KPSB00]. We will generate association rules from this data set (Figure 2.1).

age	**astigmatism**	**tear-prod-rate**	**contact-lenses**
young	no	normal	soft
young	yes	reduced	none
young	yes	normal	hard
pre-presbyopic	no	reduced	none
pre-presbyopic	no	normal	soft
pre-presbyopic	yes	normal	hard
pre-presbyopic	yes	normal	none
presbyopic	no	reduced	none
presbyopic	no	normal	none
presbyopic	yes	reduced	none
presbyopic	yes	normal	hard

Table 2.1: Subset of the contact-lenses data set

Each attribute-value pair is referred to as an item. For brevity, attribute-value pairs are denoted only by their values. For example, age = young will be written as young.

2.1.2.1 The Frequent Itemset Mining Stage

In the first iteration of Apriori's frequent itemset mining stage, each item becomes part of the 1-item candidate set C_1. The algorithm makes a pass over the data set to count support for C_1, see Figure 2.1. Those itemsets satisfying the minimum support will form L_1, the set of frequent itemsets of size 1. The ones that have support less than the minimum support threshold are shown in gray in Figure 2.1.

To generate candidates of size 2 (C_2) itemsets, the level 1 collection of frequent itemsets is joined with itself. This join is denoted by $L_1 \bowtie L_1$ and is equal to the

collection of all set unions of different itemsets in L_1. The algorithm scans the database for support of the items in C_2. Those itemsets satisfying the minimum support condition will form L_2.

When generating candidates of size 3 (C_3), $L_2 \bowtie L_2$ is performed but with a condition. Apriori assumes that the items in an itemset are sorted according to a predefined order (e.g. lexicographic order). The join, $L_k \bowtie L_k$ for $k > 1$, has the condition that for two itemsets from L_k to be joined, the first $k-1$ item(s) must be the same in both itemsets. This ensures that the generated candidate is of size k and that most of the subsets of the set are frequent. Before counting support for all the items in C_3, the Apriori property is applied. The Apriori property [AS94] states that all nonempty subsets of an itemset must be frequent for this itemset to be frequent. The Apriori property prunes the search space.

The Apriori algorithm continues to generate frequent itemsets until it cannot generate any more candidate itemsets.

2.1.2.2 The Rule Generation Stage

The frequent itemsets produced are used to generate association rules that satisfy minimum support and minimum confidence. For each frequent itemset, all possible splits of the itemset into two part (antecedent and consequent) are generated and the rule so generated is outputted by the Apriori if the rule satisfies the minimum confidence condition as seen in Figure 2.2.

2.2 Classification

Classification is the process of learning a function or a model from a data set (training data) so that the function can be used to predict the classification of a novel instance,

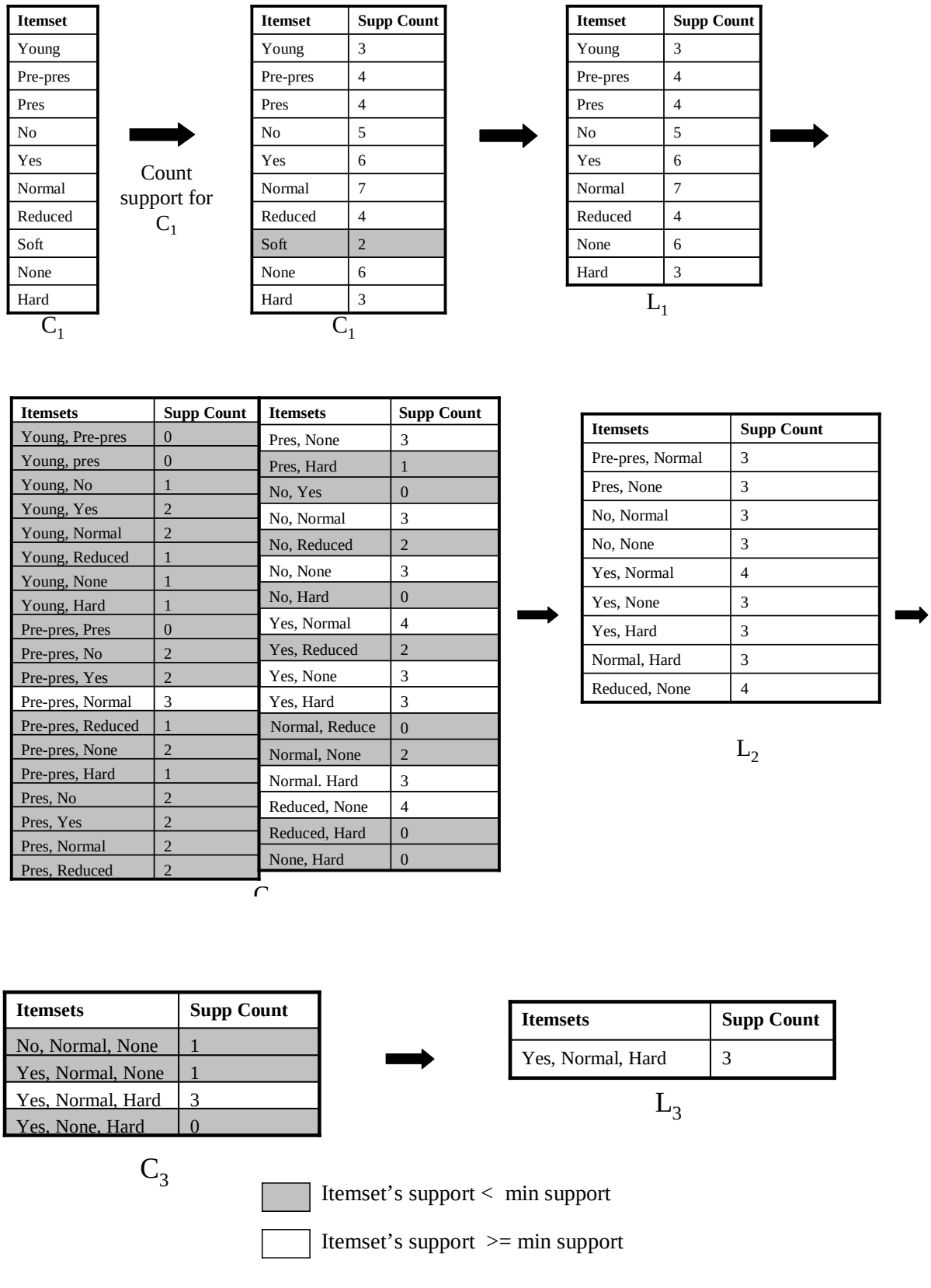

Figure 2.1: Generation of candidate itemsets and frequent itemsets from the dataset in Table 2.1 when support count is 3

Tear-prod-rate = reduced ➔ contact-lenses = none [Conf: 1.0, Sup: 0.36]

Contact-lenses = none ➔ tear-prod-rate = reduced [Conf: 0.67, Sup: 0.36]

Astigmatism = yes ➔ tear-prod-rate = normal [Conf: 0.67, Sup: 0.36]

Tear-prod-rate = normal ➔ astigmatism = yes [Conf: 0.57, Sup: 0.36]

Figure 2.2: Generated rules from frequent itemsets with confidence greater than or equal to 50%

whose classification is unknown. Classification models are frequently represented as rules of this form: $P \rightarrow c$ where P is a pattern in the training data (P forms the set of predicting attribute(s)) and c is the class label or target attribute.

Some of the common classification techniques are decision trees, naïve Bayes, and neural nets [HPY00]. In this thesis, we will study the building of classification models or classifiers from association rules.

2.2.1 Classifier Performance

Classifier performance is usually measured by accuracy, the percentage of correct predictions over the total number of predictions made. Many other measures are also used to understand the different aspects of the generated model such as: sensitivity, specificity precision and recall [HPY00]. In this thesis, we will primarily focus on precision and recall, which are measures borrowed from information retrieval. These measures are defined as follows:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

To understand true/false positives and negatives, let us use an example from information retrieval. One common example would be a web search engine returning results based on a user query. Let us define Q as the query. For any given Q, the answer space, G, can be split into what is relevant and what is not relevant. The returned answer A may contain some relevant information and/or some non-relevant information. Among the results A, the relevant information is called true positives, and the non-relevant information is called false positives. Among results that are not returned (G - A), the relevant information for Q is called false negatives and the non-relevant information is called true negatives. So precision is a ratio of relevant results to all results and recall is a ratio of relevant results to all relevant information.

The initial phase of the process is model(classifier) construction. A model is defined as a function that can map an unlabeled instance to a predefined class label. A model is constructed from data where each instance has a class label. These data are called the training set. The constructed model is tested to determine how well it predicts new instances. Testing can be done in different ways: test over the training set or test over an independent test set. Testing the classifier over the training set is usually not a good way to measure the accuracy of the classifier since the classifier has been constructed from the same data. But the testing on the training set is useful in identifying any errors in model construction. A poor accuracy rate on the training set may mean a poorly learned classifier. Using a separate test set is a good way of determining how well the classifier will perform on novel instances.

Training and testing can be accomplished in different ways depending on the amount of available data. If the number of instances available is large, the available data set may be split into a training set and a testing set (usually 66% for training and the rest of testing is considered a good split). This method of training is often a luxury in many domains as the data available for training may be insufficient.

The number of training instances has a direct effect on the classifying ability of the model built from that number of instances. When there is a limited amount of data for training and testing, n fold cross-validation is a preferred way to maximize the use of available data to produce a good classifier. In n fold cross-validation, the data is divided into n folds, and each fold in turn is used for testing, while the other folds are used for training. The reported accuracy is the average over the n iterations of training and testing.

2.3 Classification Association Rules

The use of association rules for classification was proposed in [LHM98]. In associative classification, the focus is to produce association rules that have only a particular attribute in the consequent. These association rules produced are called class association rules(CARs).

Associative classification differs from general association rule mining by introducing a constraint as to the attribute that must appear on the consequent of the rule. The produced rules can be used to build a model or classifier. CARs are a particular case of constrained association rules. There has been research in this area about integrating (pushing) these constraints into the mining phase rather than filtering the enormous number of rules produced using the constraints as post-processing filters. One paper on this area [SVA97b] proposes different ways of pushing the constraints into the mining phase. The general advantages are faster execution and lower memory utilization.

The CBA-RG algorithm is an extension of the Apriori algorithm. The goal of this algorithm is to find all rule items of the form $< condset, y >$ where $condset$ is a set of items, and $y \in Y$ where Y is the set of class labels. The support count of the

rule item is the number of instances in the data set D that contain the *condset* and are labeled with y. Each rule item corresponds to a rule of the form: $condset \rightarrow y$.

Rule items that have support greater than or equal to minsup are called frequent rule items, while the others are called infrequent rule items. For all rule items that have the same *condset*, the one with the highest confidence is selected as the representative of those rule items. The confidence of rule items are calculated to determine if the rule item meets minconf. The set of rules that is selected after checking for support and confidence is called the classification association rules (CARs).

2.4 Other Classifiers

2.4.1 Zero-R

Zero-R is a very basic classification technique that predicts the majority class from the training set and is useful as a benchmark to compare performances of other classifiers [FW00]. In the case of numeric attributes, Zero-R predicts the average value of the target attribute from the training set.

2.4.2 J4.8

J4.8 is Weka's [FW00] implementation of the C4.5 decision tree algorithm [Qui93].

2.5 The WEKA System

The Waikato Environment for machine learning, Weka, [FW00] is an open source machine learning environment with many useful data mining and machine learning

algorithms. Currently, Weka is the de-facto machine learning and data mining environment at Worcester Polytechnic Institute (WPI). Members of the WPI Knowledge Discovery and Data mining Research Group (KDDRG) have modified algorithms as well as embedded their work into the Weka environment. One such work includes merging the Apriori implementation in Weka with the Apriori implementation in another data mining system called ARMiner [SS02]. This work has improved the working of the association rule mining part in terms of speed and memory utilization. The ARMiner system was adapted earlier by Shoemaker [Sho01] to generate association rules from set-valued datasets. The merged algorithm is called AprioriSets [SS02].

AprioriSets was further modified to handle sequence type data [Pra04]. The new algorithm is known as AprioriSetsAndSequences [Pra04]. Algorithm 1 outlines Weka's procedure for generating association rules. The input parameters include minimum confidence, upperBoundMinSupport, lowerBoundMinSupport, delta, and minNumberOfRules. The upperBoundMinSupport and the lowerBoundMinSupport form the support range within which the algorithm tries to satisfy the minNumberOfRules required. The delta parameter is the value by which the support gets lowered each time the Apriori algorithm is repeated. Initially, support is set to upperBoundMinSupport and if the number of rules generated does not satisfy the minNumberOfRules, the support is reduced by delta and the process is repeated until either the number of rules generated satisfies the minNumberOfRules or the support becomes smaller than the lowerBoundMinSupport. In Step 5, the 1-item itemsets are generated (refer to Section 2.1.2 for the working of Apriori algorithm). In steps 6-10, candidates and frequent itemsets of size starting two are generated until no more candidates can be generated. In step 11, maximum frequent itemsets, that is frequent itemsets that have no frequent supersets, are generated from the

frequent itemsets. In step 12, all possible rules are generated satisfying the minConfidence condition. If the number of rules produced is greater than or equal to minNumberOfRules or if the minSupport is lower than the lowerBoundMinSupport, the while loop is broken and the rules are returned.

In the AprioriSetsAndSequences algorithm, each attribute-value pair is represented by an integer. A mapping of the numbers to the attribute-value pair is stored in a hash table. Before the rules are generated, each number is replaced by its corresponding attribute-value pair.

Algorithm 1 Weka's Procedure for Generating Association Rules

Inputs: UpperBoundMinSupport, LowerBoundMinSupport, delta, minNumberOfRules, minConfidence
Output: rules

1. *rules* = $\emptyset$;
2. *freqItemsets* = $\emptyset$;
3. *support* = *UpperBoundSupport*;
4. ***while*** (*support* $\geq$ *LowerBoundSupport AND rules.size* $<$ *minNumberOfRules*) ***do***
5. L_1 = {*1-item itemsets*};
6. ***for*** (k = 2; $L_{k-1} \neq \emptyset$) ***do***
7. C_k = *generateCandidates*(L_{k-1});
8. L_k = *evaluateCandidates*(C_k);
9. *freqItemsets* $\cup$ *L(k)*;
10. ***end for***
11. *maxFreqItemsets* = *genMaxFreqItemset(freqItemsets)*;
12. *rules* = *GenerateAllRules(maxFreqItemsets, minConfidence)*;
13. *support* = *support - delta*;
14. *freqItemsets* = $\emptyset$;
15. ***end while***
16. *return rules*;

Chapter 3

Classification of Single-Valued Class Attributes

3.1 Classification based on Association Rules (CBA)

In this chapter, we focus on generating association rules for building classification models. The chapter consists of our proposed modifications to an association rule mining algorithm to generate classification rules. The generated rules are used to build a classification model, which is evaluated with different prediction modes to study its predictive capability.

The rules resulting from Associative Classification mining can be evaluated to select a subset of the rules that will form the model or classifier. To the best of our knowledge, Liu, Hsu, and Ma [LHM98] were the first to produce a classifier based on association rules. They show that the classifier built performs as well as or better than well known decision tree algorithms. Since then, many association rule based classifiers have been built for various domains. Among others, [ZAC02] for classifying mammography images, [YLW01] for classifying web documents, [LAR02]

for recommender systems, [CAM04] for classifying spatial data, [YL05] for document classification, and [CYZH05] for text categorization. The process of building the classifier involves selecting rules by confidence or support. Confidence is a popular criterion for rule selection to the classifier as it denotes the strength of a rule. In the case of CBA [LHM98], they use a heuristic to select a subset of the rules that classifies the training set most accurately. In some cases, the pruning is as simple as removing contradicting rules [ZAC02] or more complicated like using post pruning techniques that are used in decision trees [YLW01].

In CBA-CB [LHM98], the generated CARs are ordered based on the following definition.

Definition 3.1. Rule Ordering ($\succ$) Association

Given two rules, r_i and r_j, $r_i \succ r_j$ (r_i precedes r_j) if

- *the confidence of r_i is greater than that of r_j or,*
- *their confidence are the same, but the support of r_i is greater than that of r_j, or,*
- *both the confidence and the support of r_i and r_j are the same, but r_i is generated earlier than r_j.*

Let R be the set of CARs and D be the training data. The aim of the model construction algorithm is to choose a set of highly predictive rules in R to cover the training data D. The classifier built is of the following form: $< r_1, r_2, ..., r_n, \text{default class} >$ where $r_i \in R$, $r_a \succ r_b$ if $a < b$. Default_class is the default label used when none of the rules can classify an instance.

Algorithm 2 shows the CBA-CB procedure[LHM98]. In step 1, the rules are sorted according to the order mentioned above; then each rule is considered in turn.

Algorithm 2 CBA-CB Algorithm

Inputs: rules R, training set instances D
Output: classifier C

1. *R = sort(R);*
2. ***for*** *each rule r ∈ R in sequence* ***do***
3. *temp = ∅*
4. ***for*** *each instance d ∈ D* ***do***
5. ***if*** *d satisfies the conditions of r* ***then***
6. *store d.id in temp and mark r if it correctly classifies d;*
7. ***end if***
8. ***end for***
9. ***if*** *r is marked* ***then***
10. *insert r at the end of C;*
11. *delete all the cases with the ids in temp from D;*
12. *select the default class for the current C;*
13. *compute the total number of errors of C;*
14. ***end if***
15. ***end for***
16. *Find the first rule p in C such that C_p, the list of rules in C up to p, has the lowest total number of errors. and drop all the rules.*
17. *Add the default class associated with p to the end of C, and return C*

The rule under consideration is marked if it can classify at least one instance in the training set correctly (steps 5 and 6). If the rule is marked, all the instances covered by the rule are removed from the training set and the majority class of the rest of the training instances becomes the default class label (steps 11 and 12). The marked rule is added to the end of the classifier.

Let C_r denote the lists of rules ending in rule r that have been selected for inclusion in the classifier so far. In step 13, the classifier C_r is used to classify the instances of the training set, and evaluate the performance of the classifier. Since the classification values of the instances are known, each classification attempt or prediction can be recorded as a correct classification or wrong classification. When all the instances are classified, the classifier will be assigned an error rate which is the total number of wrong classifications over the total number of classifications.

The rule for which C_p has the lowest number of errors is found and all rules

added after this rule are removed. The default class label attached with that rule becomes the default class label of the classifier (step 17).

3.2 Post Pruning Classification Association Rules

With association rule mining, the number of rules produced might be overwhelming. As all the produced rules may not be interesting or significant, it is important to prune those rules deemed uninteresting or overfitting (rules that are very specific to the training set). Similar to decision tree post-pruning, association rules can be post-pruned to reduce the number of rules produced. Many ideas on post-pruning of decision trees were introduced by Quinlan [Qui93]. There are basically two approaches to post-pruning based on error rates [FW00]. One is to divide the data set into training, validation and testing sets. With this approach, the rules will be built using the training set, and pruning will be done based on the performance of the rules on the validation set. With the second approach, there is no separate validation set, but the training set is used as the validation set. The latter technique is known as pessimistic error pruning.

Pessimistic error pruning is a heuristic based on statistical reasoning [Qui93] (see also [FW00]). For each rule, let the number of errors on the training set be E and the number of cases covered on the training set be N (those instances containing the antecedent of the rule). The observed error rate is $f = E/N$. Let the true error rate (unknown) be q. Here, we assume the N instances are generated by a Bernoulli process with probability q and error rate E.

The mean and variance of a single Bernoulli trial with success rate p are p and $p(1-p)$ respectively. For N Bernoulli trials, the success rate f is a random variable with mean equal to p, and the variance is reduced to $p(1-p) \div N$. For large N, the

value of the random variable f approaches a normal distribution.

The probability that a random variable, X, with 0 mean lies within a confidence range of width $2z$ is

$$Pr\left[-z \leq X \leq +z\right] = c$$

where c is the confidence level.

For random variable f to have a 0 mean and unit variance, we subtract mean p from f and divide by standard deviation σ, where $\sigma = \sqrt{p(1-p)/N}$.

$$Pr\left[\frac{f-q}{\sqrt{q(1-q)/N}} > z\right] = c$$

The upper confidence limit for q in the expression above provides a pessimistic estimate of e (see [FW00]) error rate at a given node:

$$e = \frac{f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}}$$

Rule R is compared with its subrules, that is rules in which one or more items are removed from the antecedent of R. If a rule R has a higher pessimistic error rate than any of its subrules, R is pruned while retaining the subrules.

3.3 Association Rule Based Classification Model Construction

In this section, we describe the approach we have taken to accomplish our primary goal of building a classification system based on association rules. This includes generating classification rules from AprioriSetsAndSequences and carrying out post-pruning to reduce the cardinality of the set of generated rules and building models from the pruned rules.

3.3.1 Generating Classification Association Rules

As we mentioned in the Chapter 2, classification association rules (CARs) are a subset of association rules with a predefined target or class in the consequent. An inefficient way of obtaining CARs is to generate all the frequent itemsets for a data set and in the process of generating rules from these itemsets, prune away rules that do not conform to CARs.

In our work, we have generated only those frequent itemsets that can produce CARs while the others are pruned away at the frequent itemset mining phase. Every CAR has a class attribute or target on the consequent of the rule. As this target is predefined, we can use this target as a semantic constraint to generate frequent itemsets consisting of the class attribute.

Definition 3.2. Semantic Constraints

A semantic constraint is a requirement that an attribute(s) must appear or must not appear in the antecedent and/or consequent of a rule.

Definition 3.3. Syntactic Constraints

A Syntactic constraint is a requirement placed on the number of attribute-value pairs on either the antecedent or consequent of a rule.

3.3.2 Generating Rules with Semantic Constraints

In many cases, we are interested in generating rules with one or more semantic constraints. In the contact-lenses data set depicted in Table 2.1, we may want to generate rules such that the contacts, age and tear_prod_rate are represented in each of them. These three attributes contribute to the semantic constraints. For the rules to have these three attributes, the frequent itemsets must contain them. Therefore, it will suffice if we generate only items sets that include the three attributes we

are interested in. The frequent itemset may have other attributes. We are able to use the semantic constraints as conditions in the join step of the Apriori candidate generation phase.

The approach we have used to prune itemsets that do not contain the required attributes is closely related to the implementation of AprioriSetsAndSequences. Our goal is to generate only itemsets that have all the required attributes (constraints).

In the AprioriSetsAndSequences algorithm [Pra04], each attribute value pair is mapped to a number (item number), see Section 2.5. This numbering is done in such a way that the attribute values of the first attribute in the data set receives the lowest numbers followed by the attribute values of the second attribute and so on. A hash table stores the mapping between the numbers and the attribute values. Numbers assigned to an attribute's values are consecutive.

To allow for pruning of itemsets that may not contain the attributes we desire, we reorder the attributes so that the attributes that are semantic constraints (required attributes) are given smaller numbers than the non-required attributes. Therefore, in the contact-lenses data set, attribute-values of contacts, age, and tear_prod_rate will be assigned smaller numbers than the values of the other attribute, astigmatism, as show in Table 3.1.

contact-lenses=soft	1
contact-lenses=none	2
contact-lenses=hard	3
age=young	4
age=pre-presbyopic	5
age=presbyopic	6
tear-prod-rate=normal	7
tear=prod-rate=reduced	8
astigmatism=yes	9
astigmatism=no	10

Table 3.1: Attribute-values renumbered to give lower numbers to required attributes

Definition 3.4. Sorted Set of Semantic Constraints

Let $A = \{a_1, a_2, ..., a_n\}$ *be the set of attributes in the data set. A sorted set* $C = \{c_1, c_2...c_k\}$ *is defined as the set of semantic constraints such that* $c_i \in A$ *for all* i, $1 \leq i \leq k$, *and constraints are sorted such that* c_i *precedes* c_k *if* $i < k$ *and* $a_1 = c_1, a_2 = c_2 \ldots a_k = c_k$.

Modified Itemset Generation Join Step

A candidate itemset of size $(k+1)$ is generated from 2 itemsets X and Y of size k where X precedes Y in lexicographic order if the following two conditions are satisfied:

- if X contains m constraints where $m > 1$, the constraints must be the first m constraints from the set of constraints (i.e., c_1, c_2, $\ldots c_m$).
- if X contains less than k constraints, it cannot contain a non-constrained item.

Theorem 3.1. *Let* $A = \{a_1, a_2, \ldots, a_n\}$ *be the set of attributes in the data set. Let* $C = \{c_1, c_2, \ldots, c_k\}$ *be the set of semantic constraints such that* $c_i \in A$ *for all* i, $1 \leq i \leq k$. *Let* X *and* Y *be two itemsets such that* $X \leq Y$ *and* $a_1 = c_1, a_2 = c_2 \ldots a_k = c_k$. *If itemsets* X *and* Y *do not satisfy the the following conditions then the join of* X *and* Y *cannot generate a rule that satisfies all the constraints in* C.

- *if* X *contains* m *constraints where* $m > 1$, *the constraints must be the first* m *constraints from the set of constraints (i.e.,* c_1, c_2, $\ldots$, c_m*).*
- *if* X *contains less than* k *constraints, it cannot contain a non-required attribute (non-constrained attribute).*

Proof. We reiterate that the attribute-value pairs will be ordered such that those that are required attributes(constrained attributes) will be placed lower than the

non-required attributes in the lexicographic order. In the join step, we can make a determination as to whether the join of X and Y will produce an itemset that has the potential to end up with all the required attributes.

Given $X = \{x_1, x_2, .., x_{n-1}, x_n\}$ and $Y = \{y_1, y_2, .., y_{n-1}, y_n\}$ where $x_1 \dots x_n$ and $y_1 \dots y_n$ are items representing attribute-value pairs. For X and Y to join, the apriori join condition, $x_1 = y_1, x_2 = y_2, x_{n-1} = y_{n-1}$ and $x_n \neq y_n$ must be true. The join of X and Y will result in an itemset of size $n + 1$.

The first condition we have introduced as part of theorem 3.1 is that if X contains items from m constraints, those m constraints must be the first m constraints based on lexicographic order. Otherwise, X will join with a Y that does not have the constraint. Let us suppose that X does not contain an item from c_j where $j \leq m$. As we know that X will join with Y such that the first $n - 1$ items from both X and Y are the same, which means Y will also not contain an item from c_j. The resulting itemset Z of size $n + 1$ will not contain c_j. Using the previous argument, we know that Z cannot join with another itemset such that the missing constraint c_j can be included in the resulting set. This shows that any itemsets resulting from the original X will not have c_j and therefore neither X nor any of its supersets will form rules that satisfy all the constraints.

The other condition we have introduced as part of theorem 3.1 is that the X cannot contain item(s) from non-required attributes as long as it does not contain all the required attributes. Let us consider the different cases that X can take in terms of having required and non-required attributes:

Case 1: If itemset X contains only items from required attributes while adhering to the first condition mentioned above, i.e., items from $1 \dots n$ are required and are in order. In this case, X can join with Y, such that the X and Y have the same number of items n and the first $n - 1$ items of X and Y are the same. This join

will go ahead as we cannot make a determination as to if this resulting itemset will have all the constraints.

Case 2: If itemset X contains items some required attributes, c_1, c_2, c_j where $j < k$ and one non-required attribute a_m. In this case, X can only join with a Y that has the same set of items from the same set of required attributes, c_1, c_2, c_j with one non-required attribute. The resulting Z itemset will not become a potential candidate for rule generation as it is missing the required attributes $c_{j+1} \ldots c_k$. Therefore, X will not be joined with Y.

Case 3: If itemset X contains all the required attributes and one non-required attribute. In this case, as X has met the criteria, we can join X with the appropriate Y. □

3.3.2.1 Generating Frequent Itemsets

Let us generate frequent itemsets from the contact-lenses data set:

In Figure 3.1, we showed the attribute-value pairs and how they are numbered. The required attributes are: contacts, age and tear_prod_rate.

itemset	Support	itemset	Support	itemset	Support	itemset	Support
{ 1, 2 }	0	{1, 3}	0	{1, 4}	1	{1, 5}	1
{1, 6}	0	{1, 7}	2	{1, 8}	0	{1, 9}	0
{1, 10}	2	{2, 3}	0	{2, 4}	1	{2, 5}	2
{2, 6}	3	{2, 7}	2	{2, 8}	4	{2, 9}	3
{2, 10}	3	{3, 4}	1	{3, 5}	1	{3, 6}	1
{3, 7}	3						

Table 3.2: Candidate itemsets in the second level of itemset generation

In Tables 3.2, 3.3 and 3.4 we show the candidate itemsets generated and their support until no more candidate itemsets can be generated. We use minimum support as 1, at least one data instance must contain the itemsets for the itemsets to be considered for the next level. Those itemsets with support less than 1 were

itemset	Support	itemset	Support	itemset	Support
{1, 4, 5}	0	{1, 4, 7}	1	{1, 4, 10}	1
{1, 5, 7}	1	{1, 5, 10}	1	{2, 4, 5}	0
{2, 4, 6}	0	{2, 4, 7}	0	{2, 4, 8}	0
{2, 4, 9}	0	{2, 4, 10}	0	{2, 5, 6}	0
{2, 5, 7}	1	{2, 5, 8}	1	{2, 5, 9}	1
{2, 5, 10}	1	{3, 4, 5}	0	{3, 4, 6}	0
{3, 4, 7}	1	{3, 4, 9}	1	{3, 5, 6}	0
{3, 5, 7}	1	{3, 5, 9}	1	{3, 6, 7}	1
{3, 6, 9}	1				

Table 3.3: Candidates itemsets in the third level of itemset generation

itemset	Support	itemset	Support	itemset	Support
{1, 4, 7, 10}	1	{1, 5, 7 , 10}	1	{2, 4, 8, 9}	0
{2, 5, 7, 8}	0	{2, 5, 7, 9}	1	{2, 5, 7, 10}	0
{2, 5, 8, 9}	0	{2, 5, 8 , 10}	1	{3, 4, 7, 9}	1
{3, 5, 7, 9}	1	{3, 6, 7, 9}	1		

Table 3.4: Generated itemsets and their support in the third level of itemset generation

dropped from the frequent itemsets group that was used in generating candidate itemsets for the next level.

In Table 3.2, we observe that only itemsets starting with an item from the first attribute, contact-lenses, is generated. This is a result of itemset pruning that is part of itemset generation.

3.3.2.2 Generating Maximal Frequent Itemsets

All the frequent itemsets generated from the itemset generation step are used to generate maximal frequent itemsets. A maximal frequent itemset is an itemset that is not a subset of any other itemset. This stage reduces the number of itemsets we are working with significantly.

3.3.2.3 Counting Support for those Itemsets without Support

Using the maximal itemsets, we generate all the subsets of the maximal itemsets and determine if each subset has support counted. As we know from the itemset generation stage, some itemsets may not be generated because of the itemset pruning step and therefore will not have their support counted. Those itemsets without support will need to have their support counted prior to the rule generation stage. When generating rules from the maximal itemsets, a pruned itemset may appear on the antecedent or consequent of a rule. Suppose X represents the antecedent and Y represents the consequent, confidence of a rule is computed as $support\{X \cup Y\} \div support\{X\}$. Therefore, it is essential that all subsets of a maximal itemset have their support counted.

3.3.2.4 Generating Classification Association Rules

In the case of generating association rules, all possible splits of a frequent itemset into antecedents and consequents are considered. However, in generating classification association rules, syntactic constraints are placed such that the rule generated has only items from the classification attribute on the consequent side, while no items from the classification attribute are present on the antecedent side. Further, confidence is calculated for each rule and those rules with confidence greater than or equal to the minimum confidence will form the final set of classification association rules.

3.3.3 Classification Models

In the previous sections, we discussed generating constrained classification association rules (CARs). In this section, we focus on building and using classification

models.

3.3.3.1 Building Classification Models

A classification model is a function that maps a novel unlabeled instance to a predefined class. In our work, we consider two types of models, all rules models (where all the produced CARs are used in the model) and the CBA model [LHM98], described in Section 3.1

The CBA algorithm [LHM98] is based on a heuristic and selects a subset of rules that classifies the training set most accurately. The CBA algorithm satisfies the condition that each new instance is predicted by the rule with the highest confidence.

3.3.3.2 Deploying Classification Models

Given a model and a new instance whose class is unknown, the problem of predicting the instance's class using the model is an interesting problem. There is more than one way to use the model to predict the instance's class. In association rule based classification models, rules in the model are ordered as follows:

- if rule r_i has greater confidence than r_j, then r_i precedes r_j, or
- if r_i has the same confidence as r_j, then the rule with greater support will precede the other, or
- if r_i has the same support and the same confidence as r_j, then the rule with then smaller number of items in the antecedent will precede, or
- if r_i has the same support, confidence and antecedent size as r_j, then the order between the two rules is random.

Rules of high confidence are thought to be good for classification. Confidence alone may not make a rule very good. For instance, a rule from an instance that appears only once (high confidence but low support) may not be a good rule for classification. Rules with very high confidence and low support are useful in identifying rare events.

3.3.4 Single Rule and Multiple Rules Classification

In CBA [LHM98], a single rule is used to classify a new case. Though this may be a simple and logical way to classify, it has been shown to be less effective than using multiple rules [LHP01]. Suppose, we want to determine if a person is eligible for a bank loan with the following attributes (housing = rent, employ_status = yes, income $\geq 50K$). Imagine we have a model to classify a new case as eligible for loan or ineligible for loan. If the top three rules that match this case are as follows:

- housing = rent $\rightarrow$ loan = NO (Sup:0.01, Conf:1.0)
- income $\geq 50K \rightarrow$ loan = YES (Sup:0.05, Conf:0.93)
- employ_status = yes $\rightarrow$ loan = YES (Sup:0.15, Conf:0.9)

If we use just one rule to classify as in CBA[LHM98], we would classify the new case as loan = NO. If we consider all the three rules together, we would classify the new case as loan = YES. This shows that the class label assigned will depend on the modes of classification and it is important to have both the modes available to be able to compare and contrast the accuracies resulting from the two modes.

The following sample model will be used to explain the different ways of predicting an unknown subject or case. Let us assume our model consists of three rules in the following order:

1. age=young ⟶ contact-lenses=none [Sup:0.05, Conf: 0.9]
2. age=young AND tear-prod-rate=normal ⟶ contact-lenses=none [Sup:0.03, Conf: 0.8]
3. age=young AND astigmatism=no ⟶ contact-lenses=none [Sup:0.02, Conf:0.78]
4. age=young AND astigmatism=yes ⟶ contact-lenses=soft [Sup:0.015, Conf:0.75]

Let us consider the data instance: age = young AND tear-prod-rate=normal AND astigmatism=yes and assume that we want to predict the class label for this instance, that is, if the user requires a contact-lenses and if so what type.

3.3.4.1 Single Rule Prediction

All the rules are sorted by confidence and then by support. The class associated with the first rule that covers the instance is selected as the prediction.

If we have to use the above mentioned model to predict the contact-lenses type for a young individual, we will select the first rule (has the highest confidence) and the prediction is that contact-lenses are not prescribed.

3.3.4.2 Prediction by Weighted Majority

All rules that cover the new instance are selected and confidence (or support) is used to weigh the predictions made by each of these rules. The majority of this weighted prediction is selected as the model's prediction.

In this mode, we will use confidence to select the appropriate class label. Rules 1, 2 and 4 can be used to predict the contact-lenses type. We will use confidence as the weight to select which class is appropriate. The weight of each prediction is calculated by summing up the confidence values for different classifications from the rules which cover the new instance:

$\sum Conf_{R_{class}} = 1.70$ where $class = none$

The weight of the new instance being soft is:

$\sum Conf_{R_{class}} = 0.75$ where $class = soft$

We select the class label with the highest weight and contact-lenses=none is predicted as the class label for the test instance.

3.4 Implementation

We have implemented our classification system in WEKA [FW00]. WEKA is an open-source suite of machine learning algorithms. The motivation for implementing our thesis in WEKA is the extensive use of this system in WPI's Knowledge Discovery and Data Mining Research Group. WEKA is developed in the Java Programming Language. Figure 3.1 shows the architecture of our classification system. We modified the existing Apriori like algorithm, AprioriSetsAndSequences [Pra04] (see Section 2.5), to generate classification association rules. The generated rules are used for building models. The resulting models are tested for accuracy.

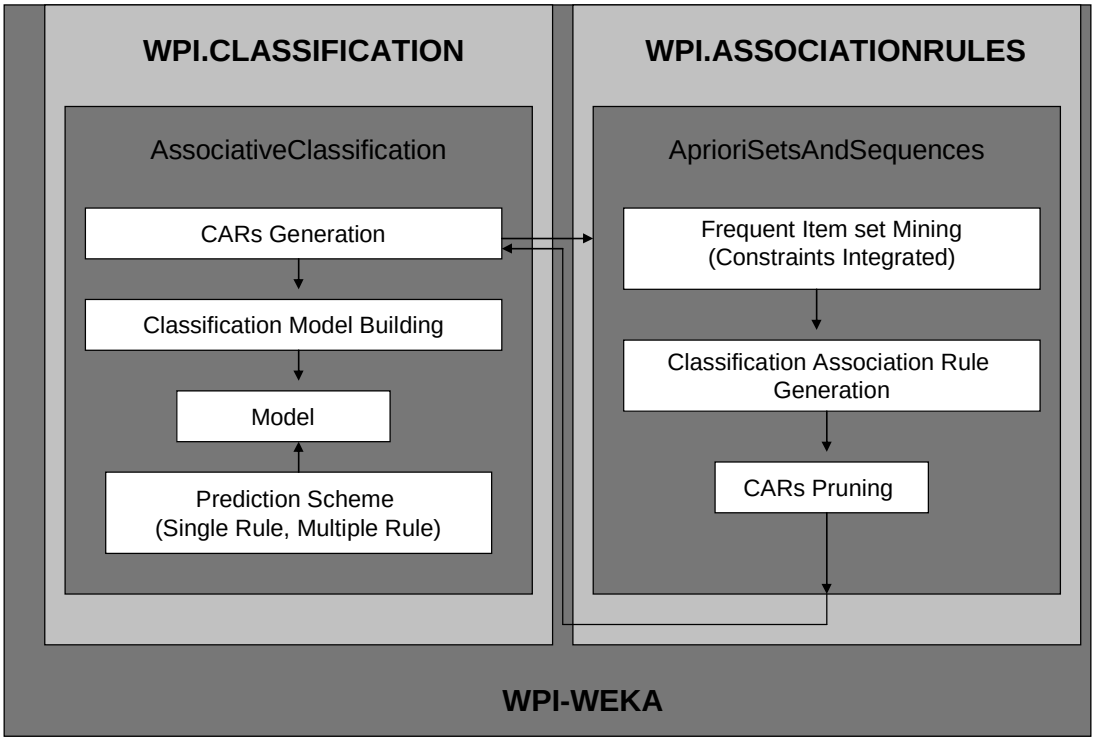

Figure 3.1: Architecture of WPI Classification System

Referring to Figure 3.1, the association rule based classification algorithm is called AssociativeClassification and is part of the Wpi.Classifiers package. We show the interaction between AssociativeClassification and AprioriSetsAndSequences. We also show the different modules in both the algorithms.

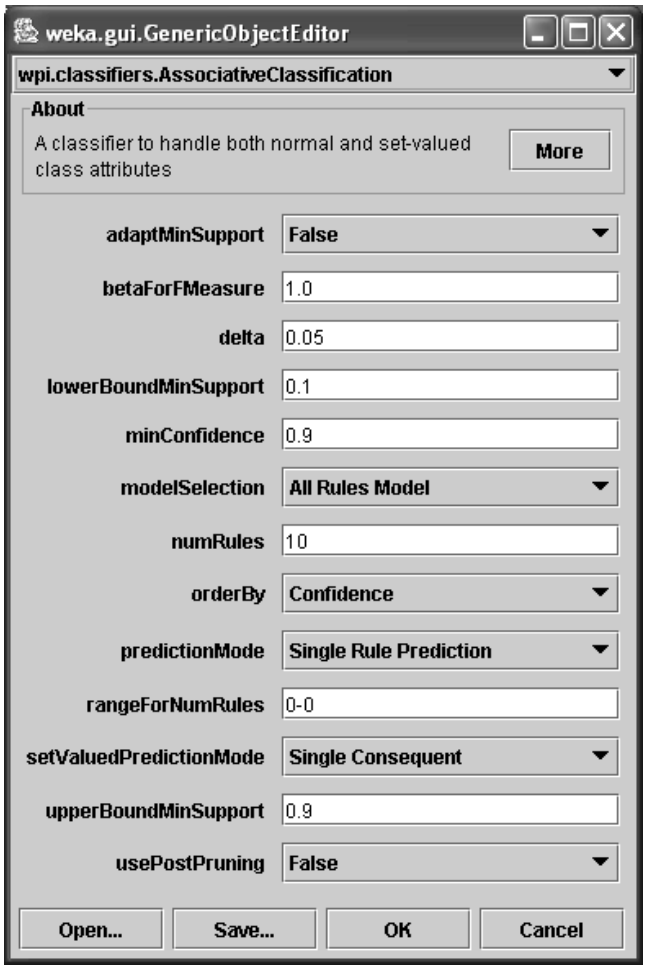

Figure 3.2: Parameter Menu For Associative Classification

Figure 3.2 shows the parameter menu for AssociativeClassification. In this menu, the user can specify the following options: minimum confidence, minimum support, starting support, support delta, minimum number of rules, which model to build (CBA or All Rules Model), if post-pruning is allowed, and how to predict (single rule or multiple rule). The new parameters added by our work include: modelSelection, predictionMode and usePostPruning. For description of other parameters, please see Chapter 2.

- modelSelection - the type of algorithm used to select a classification model

(e.g., CBA, All Rules Model)

- predictionMode - which prediction method to follow (e.g., Single Rule, Multiple Rules)

- usePostPruning - whether to post prune the association rules based on pessimistic error before building a classification model. See Section 3.2

Algorithm 3 is the modified control procedure to mine (constrained) CARS. In this thesis, we have modified the original control procedure (see Algorithm 1) to allow pruning of rules based on pessimistic error. We have also modified the algorithm to allow for presence or absence of items (semantic constraints) in the rules. More precisely, users can specify an item to appear or not to appear on either the antecedent or the consequent of a rule. We use a pruning technique to generate only itemsets that can potentially become part of the user specified rules. In the rule generation step, before a rule is generated, it is checked to see if it satisfies the user specified constraints and, if so, the rule gets generated. WEKA contains many well known classification algorithms and one significant contribution of this thesis is the classifier based on Association rule mining algorithm.

Input parameters include requiredAntecedent, requiredConsequent, disallowedAntecedent and disallowedConsequent. The while loop in Step 5 repeats itself until the support threshold is below the minsupport or the number of rules generated suffices according to the user specified number of rules. If we look into the iterative process of generating itemsets and rules from them: In Step 6, we generate the 1-item itemsets. In steps 11-15, the condition exhausts all possible itemsets that can be produced until no more items of size k can be joined to produce items of size (k+1). Only those itemsets that will potentially yield rules with the required itemsets are generated. The algorithm for this can be seen in Section 3.3. The

Algorithm 3 Modified AprioriSetsAndSequences Control Procedure

Inputs: requiredAntecedents, requiredConsequents, disallowedAntecedents, disallowedConsequents, numRules
Outputs: rules

1. *rules* = $\emptyset$;
2. *support* = *UpperBoundSupport;*
3. *freqItemsets* = $\emptyset$;
4. *requiredItems* = *requiredAntecedents* $\cup$ *requiredConsequents*
5. ***while*** *(support* > *minsupport AND rules.size* < *numRules)* ***do***
6. L_1 = {*1-item itemsets*};
7. ***for*** *(k = 2;* $L_{k-1} \neq \emptyset$*)* ***do***
8. C_k = *generateCandidates(*L_{k-1}*, requiredItems);*
9. L_k = *evaluateCandidates(*C_k*);*
10. *freqItemsets* $\cup$ *L(k);*
11. ***end for***
12. *maxFreqItemsets = genMaxFreqItemset(freqItemsets);*
13. *rules = GenerateAllRules(maxFreqItemsets, requiredAntecedents, requiredConsequents);*
14. *rules = PruneRules(rules);*
15. ***if*** *(rules.size* > *minRules)* ***then***
16. *return rules;*
17. ***end if***
18. *support = support - delta;*
19. ***end while***

generated candidates are evaluated to see if they have minimum support (step 13). In step 16, the frequent itemsets are used to generate the maximal frequent itemsets. In 17, we generate all rules according to user requests on required antecedents and consequents. In step 18, the rules may be pruned if the pruning option is set on. If the resulting number of rules is equal to or exceeds the user desired number of rules, the rules are returned.

Algorithm 4 is the AssociativeClassification algorithm that we have developed as the core of our classification system. In step 1, the input parameters are passed to AprioriSetsAndSequences algorithm to generate classification rules. Using the user specified model parameters, a model is generated from the classification rules. In step 4, the model is tested against the test set and the results are shown on the screen.

Algorithm 4 AssociativeClassification

Inputs: minrules, minimum confidence, minimum support, starting support, rule post-pruning (boolean), model, prediction, trainingSet
Output: modelTestResults

1. *rules = AprioriSetsAndSequences(trainingSet, numRules, minSupport, startingSupport, minConf, numRules);*
2. *rules = sort(rules);*
3. *model = generateModel(rules, model);*
4. *testModel(model);*
5. *outputStats();*

Figure 3.3 shows the parameter menu for association rule mining. In this thesis, we included the following parameters:

- disallowedAntecedents - those attributes not to appear on the left hand side of the rules.
- disallowedConsequents - those attributes not to appear on the right hand side of the rules.

- adaptMinSupport - switch to use adaptive minimum support (see Chapter 5).
- numRules - applies in the case when adaptive minimum support is used.
- useItemSetPruning - switch to use itemset pruning in the mining stage based on required antecedents and consequents or do not prune itemsets.
- usePostPruning - switch to use post pruning based on pessimistic error to reduce the number of generated rules.
- maxEvents - AprioriSetsAndSequences is capable of handling set-valued and sequential data. See [Pra04] for details on this and other sequence related parameters.

3.5 Experimental Evaluation

In this section, we describe the experiments carried out to compare and evaluate our classification system. We break down this section into data description, evaluation metrics and experimental results. We show the performance improvement obtained by pushing pruning of itemsets at the frequent itemset generation level and the models built by the classifier with different experimental settings.

3.5.1 Evaluation Metrics

We evaluate the classifier based on error rate with different prediction schemes. We also report the accuracy rate. The error rate signifies the number of wrong predictions over the total number of predictions. The accuracy rate signifies the number of correct predictions over the total number of predictions.

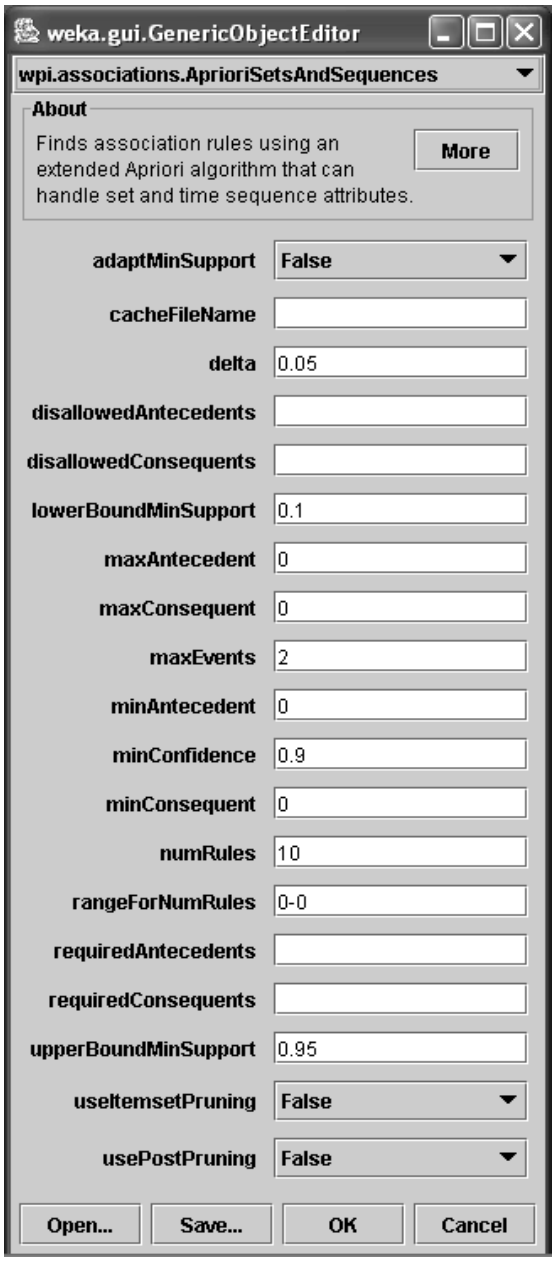

Figure 3.3: Parameter Menu for Our Extended Association Rule Mining

$$\text{accuracy} = \frac{\text{number of correct classifications}}{\text{total number of classifications made}}$$

$$\text{error} = \frac{\text{number of incorrect classifications}}{\text{total number of classifications made}}$$

A prediction involves selecting an appropriate class label for a case whose class label is unknown. For example, let $< x_1, x_2, \ldots x_k, ? >$ be a data instance whose class label is unknown (denoted by a question mark). x_i represents the value of attribute i of the instance. If this data instance is given as an input to a model, the rule(s) that covers this instance (the features of the rule are a subset of the features in the data instance) will determine the class label for the data instance.

3.5.2 Experimental Results

We divide this section into two parts. In part 1, we focus on the improvements made to AprioriSetsAndSequences by itemset pruning in the presence of constraints. Here we evaluate performance based on time taken for mining and generating rules and the number of maximal frequent itemsets generated. A frequent itemset is considered maximally frequent if none of its supersets is frequent.

3.5.2.1 Data Set

We tested the classification system with the following datasets obtained from the UCI Machine Learning Repository [SHM98]: census-income, mushroom and forest cover. Table 3.5 shows the properties of these datasets. As part of pre-processing, continuous valued attributes were discretized using WEKA's instance based discretization filter with the number of bins set to 10 [FW00].

Dataset	# attr	class	# class values	# instances
sonar	61	rocks/mines	2	208
census-income	15	income-level	2	32,561
mushroom	23	edible/poisonous	2	8,124
forest cover	17	forest cover type	7	74,056

Table 3.5: Dataset Properties

3.5.2.2 Itemset Pruning in the presence of Constraints

As part of our experiments, we were interested in comparing itemset pruning vs. non-pruning. We ran experiments with the mushroom, census-income and forest cover datasets. We generated single and multiple constraint classification rules. We observed the resulting parameters such as the number of itemsets produced, number of maximal itemsets produced and time taken for generating rules.

Table 5.3 shows the parameters used in running the experiments. In these experiments, the goal was to generate as many rules as possible with the support greater than or equal to 1%. The minimum confidence was set to 50%.

Support	Confidence
1%	50%

Table 3.6: Experimental Parameters

Prune	Req. Ant	Req. Con	itemsets	Rules	Max. itemsets	Time(s)
No	none	class	45391	21101	158	4951
Yes	none	class	42620	21101	42	4357
No	odor	class	45391	8288	158	1153
Yes	odor	class	33160	8288	26	1813

Table 3.7: Comparison of Constraint-based Pruning vs. Non-Pruning for Mushroom Dataset

In Table 3.7, we present the results for the mushroom dataset. The first column shows if constraint based pruning was selected or not. In the case of pruning being switched off, all candidate itemsets are used in generating valid itemsets at each level of the Apriori process. In comparing the first two rows (single constraint), we observe

the reduction in the number of itemsets produced and the reduction in time taken for generating the rules. But interestingly, in the next two rows (double constraint) even though the number of itemsets produced decreases, the time taken increases. We figured this is a case where a large number of item-subsets are dropped from consideration due to the constraint based pruning. The final scan of the database for support of those items costs a significant time, increasing the overall time (see Section 3.3.2.3).

Prune	**Req. Ant**	**Req. Con**	**itemsets**	**Rules**	**Max. itemsets**	**Time(s)**
No	none	class	1071	350	82	85
Yes	none	class	410	350	36	88
No	relationship	class	1071	22	82	87
Yes	relationship	class	100	22	31	16

Table 3.8: Comparison of Constraint-based Pruning vs. Non-Pruning for Census-Income Dataset

As seen in Table 3.8, in the case of a single constraint, the results for pruning and non-pruning are very similar. In the case of two constraints, the pruning leads to better performance in terms of time, approximately 1/5 of the time taken without pruning.

Prune	**Req. Ant**	**Req. Con**	**itemsets**	**Rules**	**Max. itemsets**	**Time(s)**
No	none	class	4297	1247	45	651
Yes	none	class	2673	1247	19	613
No	aspect	class	4297	144	45	678
Yes	aspect	class	605	144	22	208

Table 3.9: Comparison of Constraint-based Pruning vs. Non-Pruning for Forest-Cover Dataset

In Table 3.9, we observe reduction in time with pruning in both single constraint and double constraint.

3.5.2.3 Comparison of Different Classifiers

In this set of experiments, we compared the performance of the CBA classifier with the All Rules Model (ARM) classifier. We also compared these performances with other well-known classifiers such as Zero-R and J-4.8 (Decision Trees). In the case of CBA and ARM, we also experimented with the different prediction modes such as Single Rule and Weighted by Confidence. In these experiments, we used a split of 66% for the training set and the rest for the testing set.

Classifier	**Pred Mode**	**Test Option**	**Accuracy**	**Num Rules**
CBA	Single Rule	66% split	74.65%	44
CBA	Weight(conf)	66% split	73.42%	44
ARM	Single Rule	66% split	76.05%	33733
ARM	Weight(conf)	66% split	64.78%	33733
Zero-R	-	66% split	54.93%	1
J4.8	-	66% split	70.43%	11

Table 3.10: CBA, ARM, J48 and Zero-R on Sonar Dataset (minsupp = 1%, minConf = 50%)

Table 3.10 shows the results for the sonar dataset using CBA, ARM and other classifiers. Both CBA and ARM perform better than J48, Prism and Zero-R. In fact, ARM produces the best accuracy of 76.05% with single rule prediction. In the case of CBA, the best accuracy is obtained with Single Rule prediction mode of 74.65%.

Classifier	**Pred Mode**	**% Split**	**Accuracy**	**Num Rules**
CBA	Single Rule	66%	83.5%	545
CBA	Weight(conf)	66%	83.5%	545
ARM	Single rule	66%	80.87%	9754
ARM	Weight(conf)	66%	80.87%	9754
J4.8		66%	84.07%	331
Zero-R		66%	76.27%	1

Table 3.11: CBA, ARM, Zero-R and J48 on Census-Income Dataset (minsupp = 1%, minConf = 50%)

Table 3.11 shows the results for CBA, ARM and other classifiers with the census income data. In this case, J4.8 performs marginally better than CBA. ARM performs below CBA in both modes. In the case of CBA, both modes perform similarly.

Classifier	**Pred Mode**	**% Split**	**Accuracy**	**Num Rules**
CBA	Single Rule	66%	99.02%	23
CBA	Weight(conf)	66%	99.02%	23
ARM	Single Rule	66%	98.58%	12496
ARM	Weight(Conf)	66%	98.58%	12496
J4.8		66%	100%	25
Zero-R		66%	50.4%	1

Table 3.12: CBA, ARM, J48 and Zero-R on Mushroom Dataset (minsupp = 1%, minConf = 50%)

Table 3.12 shows the results for CBA, ARM and Other classifiers on the mushroom dataset respectively. CBA performs well, just marginally below J4.8. Again, Single Rule Prediction mode performs better than other modes.

Classifier	**Pred Mode**	**% Split**	**Accuracy**	**Num Rules**
CBA	Single Rule	66%	62.53%	144
CBA	Weight(Conf)	66%	62.53%	144
ARM	Single Rule	66%	61.74%	6901
ARM	Weight(Conf)	66%	61.74%	6901
J4.8		66%	88%	5801
Zero-R		66%	61%	1

Table 3.13: CBA, ARM, J48 and Zero-R on Forest Cover Dataset (minsupp = 1%, minConf = 50%)

Table 3.13 shows results for classification of the forest cover dataset using CBA, ARM, Zero-R and J4.8 Classifiers. There is marginal difference of less than 1% between CBA and ARM. In both CBA and ARM, the prediction modes do not differentiate the results. J4.8's ability to handle numeric attributes directly, in contrast with association rules, gives it a clear advantage on the Forest Cover dataset, which contains several numeric attributes.

Chapter 4

Adaptive Minimum Support

4.1 Adaptive Minimum Support

The problem of association rule mining can be broken down into two subproblems: generating all combinations of items(frequent itemsets) that appear in transactions with support greater than or equal to a support threshold, called *minsupport*; and generating rules from the frequent itemsets that have confidence greater or than equal to a confidence threshold, called *minconfidence*. When mining for associations, picking the right *minsupport* is a challenging problem. By the right *minsupport* we mean the *minsupport* that will produce a number of rules within a desired range, denoted as $[R_{min}, R_{max}]$. It is best if rules can be produced so that their number is within $[R_{min}, R_{max}]$ and ensure those rules have the highest support out of all the possible rules for a given *minconfidence*. However, to be able to produce a number of rules within $[R_{min}, R_{max}]$, we need a way to automate the process of finding the right *minsupport*. In [LAR02], an adaptive minimal support algorithm is introduced that will oversee the mining process to ensure the number of rules returned falls within a desired range. The proposed algorithm is used in association rule mining

for classification.

Although *adaptive-minsupport* has been discussed in the general framework of association rule mining, we find that it applies more importantly to classification associative rules [LHM98]. In classification association rules, target items are specified by users to appear in the consequent of the rule. The general association rule mining problem is narrowed to mining rules with specific items in the consequent. These rules are called classification association rules(CARS). Classification association rules have been used in real-time recommender systems [LAR02]. According to [LAR02], the number of rules generated is important to ensure a good recommendation. If too many rules are generated, the runtime can be too long, and a small number of rules can lead to poor recommendations. We mined for CARS using our *adaptive-minsupport* approach. Given a database of instances, range [R_{min}, R_{max}], *minconfidence* and a class variable, CARS were mined to build a classification model. A classification model was built from the generated rules. The rules were sorted according to a score, and the model constructed progressively by adding a rule with high score and testing the classifier accuracy. When accuracy starts falling, the model construction was ceased, and the current model formed the final classifier.

4.2 Approach

In our work, we have used a binary search strategy to adapt minsupport until we produce rules in the desired range or very close to it. The use of binary search in this context was suggested to us by S.A. Alvarez [Alv02]. We have a fixed minsupport that is used to start the mining process and then the minsupport is adapted either by reducing by half or doubling depending on the number of rules produced and the

range in number of rules wanted. The higher level description of our algorithm is as follows:

Given a minimum confidence and the number of rules wanted in range [R_{min}, R_{max}]

- Set the initial minsupport based on heuristics.
- Mine association rules until the number of rules produced falls within the range [R_{min}, R_{max}] by adjusting the minsupport and running the mining process again as described in Section 4.4.

4.3 Initial MinSupport Selection

Guessing an initial minsupport is an interesting problem. We can use a fixed value for minsupport, but a way of determining a value based on the data set may be more useful in reducing the number of times the mining process repeats itself. We find x, the number of items required to produce approximately R_{max} rules, ignoring support as a criterion in selecting items. Then we sort the 1-item itemsets and select the support of the x^{th} item from the sorted itemsets as the initial minsupport.

$R_{max} = \sum_{k=2}^{x-2} \sum_{i=1}^{k-1} \binom{x}{k} \binom{k}{i}$

$\binom{x}{k}$ selects the k items that will appear in the rule and the $\binom{k}{i}$ selects from these k items the ones that will appear in the antecedent of the rule.

$R_{max} = \sum_{k=2}^{x-2} \binom{x}{k} \sum_{i=1}^{k-1} \binom{k}{i}$

$R_{max} = \sum_{k=2}^{x-2} \binom{x}{k} 2^{k-1}$

$R_{max} = \frac{1}{2} \left[\sum_{k=2}^{x-2} \binom{x}{k} 2^{k} \right]$

$R_{max} = \frac{1}{2}[(1+2)^x]$

$x = \frac{log(2*R_{max})}{log3}$

The support of the x^{th} item, we use as the initial minsupport is an optimistic value as not all items will result in a rule. This support will give us a better starting point than using a fixed value. Then we reduce the minsupport as needed to find the cardinality of rules that will intersect with the specified range.

4.4 Adaptive Minimal Support Algorithm

Algorithm 5 Adaptive Minimal Support Algorithm

Inputs: Data instances D, target attribute T, Rules Range [R_{min},R_{max}], Minimum Confidence minConf
Output: Rules

```
1.  minsupport = heuristicFunction(D, R_max)
2.  upperLimitForSupp = 100%
3.  lowerLimitForSupp = 0%
4.  rules = generateRules(D, T, minConf, R_max, minsupport)
5.  while (RulesNotInRange && ((upperLimitForSupp - lowerLimitForSupp) ≥ 1%))
    do
6.    if rules.size > R_max then
7.      lowerLimitForSupp = minSupport
8.      minSupport += Min(5, ((upperLimitForSupp - lowerLimitForSupp) / 2))
9.    else
10.     upperLimitForSupp = minSupport
11.     minSupport -= Min(5, ((upperLimitForSupp - lowerLimitForSupp) / 2))
12.   end if
13.   rules2 = rules
14.   rules = generateRules(D, T, minConf, R_max, minsupport)
15. end while
16. if (NOT rulesInRange) then
17.   return maxof(rules, rules2)
18. end if
19. return rules
```

Algorithm 5 is the algorithm for the binary strategy we use for adaptive minSupport. The inputs to the algorithm include the data instances D, target attribute T, rules range [R_{min}, R_{max}], and minimum confidence minConf. The upperLimitForSupp is set to 100%, while the lowerLimitForSupp is set to 0%. The upperLimitForSupp and loweLimitForSupp used in this algorithm are not the same as the

upperBoundForSupp and lowerBoundForSupp used in the association rule mining algorithm in Weka. In this algorithm, upperLimitForSupp and lowerLimitForSupp are modified based on the number of rules returned during each generateRules procedure call, in an attempt to narrow down the support range to find the number of rules requested. Whereas in the association rule mining algorithm in Weka, the upperBoundForSupp and lowerBoundForSupp are fixed during runtime and act as the upper and lower limit for minSupport.

The initial minsupport is calculated by a heuristic process as explained in section 4.3. The association rule generation procedure is invoked with D, T, minConf, R_{max} and minSupport. In line 5, the while loop is entered if the number of rules generated is outside the range [R_{min},R_{max}] or the (upperLimitForSupp - lowerLimitForSupp) $\geq$ 1%. Inside the while loop, in line 7, we check to see if the number of rules generated is greater than R_{max}. If so, the lowerLimitForSupp is increased to minSupport and a new minSupport is increased by the minimum of 5% or (upperLimitForSupp - lowerLimitForSupp)/2. The intention here is to limit the delta of minsupport to at most 5%. In the case that the number of rules generated is less than R_{min} (line 9), upperLimitForSupp is set to minSupport and the minSupport is decreased by the minimum of 5% or (upperLimitForSupp - lowerLimitForSupp)/2. Using the altered minSupport, the rule generation process is repeated. When the while loop is exited, if the generated number of rules is not within the range [R_{min}, R_{max}], the last run or the previous run, whichever produced the most number of rules, is returned.

The generateRules procedure is the same as the one discussed in Chapter 2. It is an Apriori algorithm that uses a two-stage mining process. In the first stage, frequent itemsets are generated. These are itemsets that satisfy the minSupport threshold. In the second stage, association rules are generated from those frequent itemsets such that the confidence of each rule satisfies the minConf threshold. We

have altered the second stage to stop generating rules when $R_{max} + 1$ is reached, thereby preventing the routine from generating all rules. Note that R_{max} is a parameter to the generateRules procedure.

4.5 Experiments

4.5.1 Experiment Design

We wanted to compare the adaptive minSupport algorithm with the linear algorithm used in the WEKA Apriori algorithm. In the linear algorithm, there is only a lower bound on the desired number of rules. There are also bounds on the support, and the mining is repeated until the lower bound for the number of rules is reached or the lower bound for the support (minimum support) is reached. Note the lower and upper bound for support is not the same as the upper and lower limit for support used in the Adaptive Minimal Support (AMS) algorithm shown in Algorithm 4.4.

The linear algorithm starts out with support set to the upper bound value and if the number of rules generated is less than the user defined lower limit, the minimum support is reduced by delta (default value is 5%) and the process is repeated until either the number of rules generated satisfies the lowerbound limit or the minimum support becomes lower than the lowerbound value for minimum support. In our experiments, we used the default values for upperbound, lowerbound support and delta, which are 90%, 10% and 5% respectively. In both the experiments, the minimum confidence was set to 50%.

Dataset	# attr	# classes	# instances
mushroom	23	2	8124
autos	20	7	205

Table 4.1: Dataset Properties

We use the autos and mushroom datasets from the UCI Machine Learning Repository [SHM98] for the experiments. Table 4.1 shows the properties of the above mentioned datasets.

Rules Range	Binary		Linear	
	♯ Rules	Time(s)	♯ Rules	Time(s)
10-20	16	3	24	1
50-100	52	1	52	1
200-250	220	3	296	3
500-600	552	6	558	3
1000-2000	1698	18	1254	5
5000-10000	8526	904	6435	317
10000-11000	10056	2926	15034	1626

Table 4.2: Comparison of binary vs linear minSupport strategies in autos dataset

In Table 4.2, we show the results for the autos dataset. As we mentioned earlier, the linear strategy has only a lower bound on the desired number of rules. We see that the linear strategy performs better in terms of time taken consistently over the binary strategy. However, the number of rules produced does not consistently fall in the desired range in the case of linear strategy as opposed to the binary strategy.

In Table 4.3, we show the results for the mushroom dataset. The Binary adaptive approach takes more time than the linear approach in most instances. The number of rules returned is within the range consistently in the binary adaptive approach, while in the linear strategy, the number of rules produced exceeds the range in 2 instances.

Rules Range	Binary		Linear	
	♯ Rules	Time(s)	♯ Rules	Time(s)
10-20	12	41	12	7
50-100	50	36	50	21
200-250	220	40	218	57
500-600	526	52	662	81
1000-2000	1148	20	1148	104
5000-10000	9819	1037	5767	292
10000-11000	11121	3747	16945	2569

Table 4.3: Comparison of binary vs linear minSupport strategies in mushroom dataset

Required Number of Rules: 10-20 MinConf: 0.5
Minsupport: 0.847 rules: 02
Minsupport: 0.797 rules: 24
Minsupport: 0.822 rules: 16

Figure 4.1: Sample Run

In Figure 4.1, we show a sample run and how the minsupport is modified to generate the required number of rules.

4.5.2 Summary

Our experiments with autos and mushroom dataset show that the binary strategy generates rules whose cardinality can be controlled by the available settings. Though the time taken may be more than the time taken for a similar run with linear strategy, the advantage lies knowing the range for the number of rules that will be returned. It is possible when using the binary strategy, the minsupport is adjusted such that the number of rules produced exceeds R_{max}. This will be followed by a decrease in minsupport in an attempt to obtain a number of rules in $[R_{min}, R_{max}]$. These kinds of adjustments are the likely reason why time taken with the binary approach is greater than the time taken with the linear approach.

Chapter 5

Classification of Multi-Valued Attributes

5.1 Association Rule Mining with Set-Valued Attributes

In many domains, set-valued attributes are common. Some examples are linguistics, movies, and gene expression data. Shoemaker and Ruiz [SR03] propose extensions to Apriori Algorithm to mine association rules from datasets containing set-valued and single-valued attributes. The proposed algorithms are called Set Based Apriori (SBA) and Transformation based Apriori (TBA). In SBA, frequent itemsets are mined from set-valued attributes as the first step, then followed by mining frequent itemsets across the set-valued itemsets and single-valued attributes. In the case of TBA, the set-valued attributes are transformed into single-valued or binary attributes (using various transformations) and the Apriori algorithm is applied over the transformed data set. In our work, we make use of TBA to generate rules and build classifiers from set-valued attributes.

5.2 Classification with Set-Valued Class Attribute

It is not uncommon to come across domains where the target attribute is set-valued. One such domain is that of gene expression analysis. We have extended our classifier to be able to handle set-valued class attributes and have developed novel strategies to predict set-valued classification. There has been work done in the area of using set-valued attributes in classification of a nominal attribute [Coh96].

5.2.1 Set-Valued Class Prediction

Classification where the class label is set-valued is an interesting problem that to the best of our knowledge has not been looked at in the association rule based classification domain. How we solve this problem depends on how the set-value attribute is treated before mining frequent itemsets. In the case of AprioriSetsAndSequences, set-valued attributes are transformed into single-valued attributes. To produce rules with set-values on the consequent, the support threshold must be set very low, but this is likely to produce lots of rules and a long running time. We have developed two approaches to solve this issue.

5.2.2 E-Measure

When the target attribute is set-valued, we cannot employ the traditional way (boolean) of measuring if a prediction is the same as the actual value or not. If we did so, the classifier performance is likely to be below par; also we will lose information on the proximity between the predicted value and the actual value. Therefore, we have borrowed E-measure [LG94], discussed in Chapter 2, from the domain of information theory to measure the difference between the two sets.

When comparing two sets, we know that the sets could be the same, may have

some overlap, or no overlap at all. So we are interested in measuring the similarity between the two sets, rather than using accuracy in the traditional sense. But in the real world, it may be necessary to know how close the predicted set value is from the actual set value.

We compute E-measure for each prediction using the recall and precision values. For more on precision and recall, see Chapter 2.

Definitions for recall and precision are as follows:

$$recall = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

$$precision = \frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

In [cR79], recall and precision are fused to form a single measure called E-measure.

$$\text{E-Measure} = 1 - \frac{(\beta^2 + 1) \star precision \star recall}{\beta^2 \star precision + recall}$$

The parameter β ranges between 0 and infinity and is used to control the relative weight given to recall and precision. F-measure is a particular case of E-measure introduced by [LG94] where $\beta = 1.0$. Hence, the F-measure weights precision and recall equally.

$$\text{F-Measure} = 1 - E_{\beta=1}$$

A β value of 0.5 will be used if the user is twice as interested in precision as recall.

We use the E-measure and the F-measure to evaluate the classification of set-

valued target attributes. Note that since the E-measure is a measure of error, the lower the E-measure of a prediction, the better the prediction is. In contrast, the higher the F-measure value, the better.

5.2.3 Building Classification Models

We generated two types of models:

- models derived from all CARs using a CBA like procedure
- models consisting of all CARs

5.2.3.1 SCBA Algorithm

We modified the CBA algorithm described in Section 3.3.3 to handle set-valued classification attributes. We call the resulting algorithm Set-valued Classification Based on Association (SCBA), see Algorithm 6. We recap the CBA algorithm and point out the modifications we made to use it with set-valued attributes. The CBA algorithm selects a subset of rules that predicts the training set most accurately. As explained in Section 5.2.2 the notion of classification accuracy is undefined for set-valued classification. Our SCBA algorithm uses the E-Measure to evaluate the accuracy of a prediction. After a rule is added to the classifier C, the classifier's performance on the training set is evaluated and quantified in terms of E-Measure.

In line 16, the process of computing the average E-Measure involves using the training set instances as unlabeled instances and using the classifier to classify each training instance. If the classifier has n rules in order of $1 \ldots n$, starting with rule 1, each rule is applied on the data instances. If a rule fires on an instance, the rule's prediction is compared against the instance's target value and an E-Measure value is generated. Each instance that is classified by a rule is removed from the instance

collection. The process is repeated until there are no more instances to be classified or rules to classify. The E-Measure value is summed across all classifications and an average E-Measure value is calculated by dividing the total E-Measure by the total number of classifications. This Average E-Measure is stored along with the newly inserted rule in the classifier.

After all the instances have been classified or left unclassified, the rule that produced the lowest average E-Measure value becomes the last rule in the classifier and the rest of the rules are removed.

Hence our algorithm selects a subset of rules that produce the lowest E-Measure value.

Algorithm 6 SCBA Algorithm

Inputs: Instances D, Rules R
Output: Classifier C

1. *R = sort(R);*
2. ***for*** *each rule* $r \in R$ *in order* ***do***
3. $temp = \emptyset$
4. ***for*** *each instance* $d \in D$ ***do***
5. ***if*** *d satisfies the conditions of r* ***then***
6. *store d.id in temp*
7. ***if*** *consequent of* $rd \neq$ ***then***
8. *mark r*
9. ***end if***
10. ***end if***
11. ***end for***
12. ***if*** *r is marked* ***then***
13. *insert r at the end of C*
14. *delet all the data instances with the ids in temp from D*
15. *select the default class for the current C*
16. *compute the Average E-Measure for C and attach it with r*
17. ***end if***
18. ***end for***
19. *Find the first rule p in C such that* $C_p = \{r_1, r_2, ..r_p\}$ *has the lowest Average E-Measure*
20. *Output* C_p

We have made two modifications to the CBA-CB algorithm to be able to classify set-valued attributes. We mark a rule if the antecedent of the rule is contained in the

data instance and if at least one attribute-value pair of the consequent is contained in the data instance. Instead of computing the accuracy of the classifier, we compute E-Measure and treat it like accuracy in eliminating all rules that increase E-Measure.

5.2.3.2 All Rules Model

The All Rules Model, as it name states, is a model consisting of all the produced rules. It is a naive approach and the results of this approach are used to compare the results of the SCBA approach.

5.2.4 Model Prediction

We described in Section 5.2.3 how to construct association rule models. In this section, we describe how we use the models to predict the classification of an unlabeled instance when the classification target is set-valued.

In Table 5.1, we show a sample model that will be used in explaining the different approaches that we have developed to predict an unlabeled case. The model consists of five rules. The attributes shown are Year (movie release year), Award Won (any prestigious awards such as an Academy Award), DirCountry (movie director's country of origin), Print (color or black and white) and genre (set-valued target attribute). For more information on the attributes, refer to Section 5.3.

Classifier
Year=(1937 - 1944) ⇒ Genre= {classic,family} [C:1.0, S:0.2]
Award won=Academy ⇒ Genre= {drama} [C:0.95, S:0.2]
Year=($\geq$ 1989)$\wedge$ DirCountry=US $\wedge$ Print=col ⇒ Genre={action, drama} [C:0.5, S:0.05]
DirCountry=US $\wedge$ Print=col ⇒ Genre={action, thriller} [C:0.5, S:0.04]
DirBirth=(1926-1935) ⇒ Genre={drama, adventure} [C:0.5, S:0.02]

Table 5.1: Classifier Model. C stands for confidence and S stands for support

Figure 5.2 depicts an instance or case for which the above mentioned classifier will be used to predict the unknown class label.

Year=($\geq$ 1989)$\wedge$ DirCountry =US $\wedge$ Print=col $\wedge$ Award=Academy $\wedge$ Genre= ?

Table 5.2: Instance whose classification will be predicted

5.2.4.1 Single Rule Prediction

Given an unlabeled instance, we gather all the rules that cover the instance ordered by confidence and support. The rule with the most number of items in the consequent is selected as the rule to classify the instance. We have taken this approach to predict the class value with a set of values instead of a single value. If we used the approach of predicting the unlabeled instance with the rule that has the highest confidence as in the case of traditional associative classification, we are likely to predict every time with a single value class attribute.

The success of this approach depends on the number of rules that are generated, the more rules generated, the better the chance that there is a rule with multiple consequent values with high confidence.

When we try to predict the genre for the unlabeled instance in Table 5.2 using the classifier, we have rules 2, 3 and 4 predict the instance. Rule 3 has the highest consequent cardinality and highest confidence from the predicted class and therefore, the prediction for genre is {action, drama}.

5.2.4.2 Multiple Rule Prediction

In this approach, we collect all the consequents of the rules that cover the test instance and predict the union of those consequents as the test instance's class. If we use multiple rules to predict the case in Table 5.2, we have rules 2, 3 and 4 predict the instance. The union of the set-valued consequents from these three rules will make the prediction for genre as {action, drama, thriller}.

5.3 Experimental Evaluation

We tested the set-valued based classification system using a movie dataset constructed by [Kaw03] from three online movie-related databases: Each Movie Database [McJ97], Movie Lens Database[Gro], and the Stanford Database [Wie]. This movie dataset consists of 1628 instances. For experimental purposes, we used a 66% split, where 66% of the instances were used for training and the rest of the instances were used for testing the classifier.

Parameter	**Value**
Confidence	$\geq 50\%$
Support	$\geq 1\%$

Table 5.3: Experimental Parameters

These are the movie attributes of the dataset:

- Year of Release
- Director's Name
- Director's Year of Birth
- Director's Year of Death
- Director's Nationality
- Leading Actors' Names (set-valued attribute)
- Leading Actors' Roles (set-valued attribute)
- Print Type - Color, BW
- Awards Received - AA, AAN, AFI, BFA, Emmy, Halliwell's Film Guide, Palm

Attribute	% Missing Value
Director's Name	7%
Director's Year of Birth	53%
Director's Year of Death	93%
Director's Nationality	56%
Leading Actors' Names	55%
Leading Actors' Roles	89%
Print Type	52%
Awards Received	66%
Genre	3%

Table 5.4: Properties of Movie Dataset

- Movie Genre - action, adventure, animation, children, comedy, crime, drama, family, fantasy, foreign-art, horror, romance, sci-fi, thriller, war (set-valued target attribute)

Movie Genre was used as the target or class attribute.

Table 5.4 shows the sparseness of the dataset used in the experiment. We selected Genre as the classification attribute based on its relatively low missing value percentage.

In Table 5.5, we show the results for our set-based classification system using our adaptive minsupport strategy to mine association rules and CBA to build a classification model on the movie dataset. The parameters that have been included in the table are as follows:

- Range for Rules - This is an input parameter to the mining algorithm and the algorithm uses it as a condition to the number of rules generated. More details on this parameter are given in Chapter 4
- Pred.Mod - mode of Prediction (Single Consequent(SC) or All Consequents(AC))
- ♯Rules - number of rules produced by the adaptive association rule miner

described in Chapter 4

- C♯ - number of rules in the classifier
- E-M - value for E-Measure 1^{st} Quadrant - 2^{nd} Quadrant - 3^{rd} Quadrant
- Prec - value for Precision 1^{st} Quadrant - 2^{nd} Quadrant - 3^{rd} Quadrant
- Rec - value for Recall 1^{st} Quadrant - 2^{nd} Quadrant - 3^{rd} Quadrant
- Ave E-M - average E-Measure
- Ave Prec - average Precision
- Ave Rec - average Recall
- Uncl - percentage of unclassified instances

Results from Table 5.5 show the 20-30 rules range requirement has produced lower E-M values than the 500-1000 rules range requirement. Remember that the lower the E-measure of a prediction, the better the prediction is. We also notice the number of unclassified instances is significantly high, too high to be conclusive, in all four results. Given the fact that the classifier made with 20-30 rules range requirement had just one rule makes the results skewed. It is interesting to note that there is no difference between the two prediction modes with in a given rule range requirement. The precision values are high for all the four experiments. The recall values are 45% and 60%, not as high as precision.

Table 5.6 shows the results for set-based classification using adaptive minsupport technique and All Rules model on the movie dataset. The parameters are the same as defined for Table 5.5. With the All Rules (AR) Model, we notice that the number of unclassified instances is cut down significantly. This is a result of using

Range	♯ Rules	C ♯	Pred.Mod	E-M	Prec	Rec	Uncl
				Ave E-M	Ave Pr	Ave Rec	
500-1000	403	27	AC	0.33-0.5-0.6	1.0-1.0-1.0	0.25-0.33-0.5	88.89%
				0.45	0.83	0.45	
500-1000	403	27	SC	0.33-0.5-0.6	1.0-1.0-1.0	0.25-0.33-0.5	88.89%
				0.45	0.83	0.45	
20-30	26	1	AC	0.0-0.33-0.6	1.0-1.0-1.0	0.25-0.5-1.0	97.41%
				0.32	0.93	0.60	
20-30	26	1	SC	0.0-0.33-0.6	1.0-1.0-1.0	0.25-0.5-1.0	97.41%
				0.32	0.93	0.60	

Table 5.5: Adaptive Classification-CBA over the Movie Dataset

all the rules produced in the mining stage. The prediction modes make a difference in E-M, precision and recall values. When using the All Consequent (AC) mode, the recall values are better than when using Single Consequent (SC) mode. This is contrast to precision, which registers low values when using All Consequent (AC) mode. Average E-M is lower with All Consequents (AC).

Range	♯ Rules	C ♯	Pred.Mod	E-M	Prec	Rec	Uncl
				Ave E-M	Ave Pr	Ave Rec	
500-1000	403	403	AC	0.33-0.5-1.0	0.0-0.5-1.0	0.0-0.5-1.0	49.26%
				0.55	0.53	0.45	
500-1000	403	403	SC	0.33-0.55-1.0	0.0-0.5-1.0	0.0-0.33-0.5	49.26%
				0.6	0.55	0.33	
20-30	26	26	AC	0.33-0.6-1.0	0.0-0.75-1.0	0.0-0.75-1.0	52.2%
				0.62	0.54	0.32	
20-30	26	26	SC	0.33-0.6-1.0	0.0-1.0-1.0	0.0-0.25-0.5	52.2%
				0.63	0.56	0.30	

Table 5.6: Adaptive Classification-AR over the Movie Dataset

In Table 5.7, we show the results for non-adaptive set-based classification, where all rules with support $\geq 1\%$ were generated. Both CBA and All Rules models were used to create the classifier. CBA performs much better than AR in comparison, except for the number of unclassified instances. The precision values are excellent (91%) in the case of CBA. The Average E-M for CBA is 0.36 , much lower than the average E-M for AR. In the case of CBA, both modes produce the same results. In the case of AR, All Consequents (AC) does marginally better than SC, and this is

consistent with the previous experiments.

Model	♯ Rules	C ♯	Pred.Mod	E-M	Prec	Rec	Uncl
				Ave E-M	Ave Pr	Ave Rec	
CBA	88	4	AC	0.0-0.33-0.6	1.0-1.0-1.0	0.25-0.33-1.0	95.74%
				0.36	0.91	0.54	
CBA	88	4	SC	0.0-0.33-0.6	1.0-1.0-1.0	0.25-0.33-1.0	95.74%
				0.36	0.91	0.54	
AR	88	88	AC	0.33-0.5-1.0	0.0-0.5-1.0	0.0-0.33-0.5	49.63%
				0.59	0.54	0.36	
AR	88	88	SC	0.33-0.6-1.0	0.0-1.0-1.0	0.0-0.25-0.5	49.63%
				0.63	0.54	0.29	

Table 5.7: Non-Adaptive Classification using CBA and AR over the Movie Dataset

5.4 Further Experiments

We carried further experiments with the Motif Dataset gathered by [Tha06]. The Motif Dataset is derived from 85 promoter regions from *C. elegans* model organism. The dataset consists of two attributes, motifs and expression, both are set-valued. Each instance in the dataset corresponds to a gene. The gene's value of the motif attribute consists of the set of motifs (candidate binding sites) that appear in the promoter region of the gene; and the gene's value of the expression attribute consists of the set of cell types where the gene is expressed (i.e., transformed into protein).

Table 5.8 gives the results for set-value based classification on the motif dataset. The original dataset consisted of 85 instances. 66% of the data was for model building and the rest of the data was used for model testing. The expression attribute is used as the target of the classification. The parameters used to constraint the rules were: support 15% and confidence 50%.

Using All Rules (AR) Model, we are able to classify all the instances, while the CBA models fails to classify 35% of the instances. Overall, CBA produces lower E-M values which is mainly contributed by the high precision values relative to those

produced by AR. We notice a significant difference between the precision values in these experiments between the two models. AR producers higher recall values than CBA.

Model	**♯ Rules**	**C ♯**	**Pred.Mod**	**E-M**	**Prec**	**Rec**	**Uncl**
				Ave E-M	**Ave Pr**	**Ave Rec**	
CBA	15413	22	SC	0.33-0.55-0.71	0.0-0.5-0.5	0.0-0.5-1.0	35.71%
				0.55	0.44	0.50	
CBA	15413	22	AC	0.33-0.55-0.71	0.0-0.5-0.5	0.0-0.5-1.0	35.71%
				0.55	0.44	0.50	
AR	15413	15413	SC	0.55-0.71-0.75	0.14-0.17-0.28	0.5-1.0-1.0	0.0%
				0.65	0.26	0.76	
AR	15413	15413	AC	0.55-0.71-0.75	0.14-0.17-0.28	0.5-1.0-1.0	0.0%
				0.65	0.26	0.76	

Table 5.8: Set-Valued Based Classification of Motifs

Chapter 6

Conclusions and Future work

6.1 Itemset Pruning

We developed a technique to incorporate user-defined input constraints during the mining of association rules. We considered two types of constraints namely semantic and synctatic. Semantic constraints require the presence/absence of particular itemsets in the rules; while synctatic constraints limit the number of attribute-value pairs in either the antecedent or the consequent of a rule. We developed a characterization of those itemsets that will potentially form rules that satisfy the given constraints. This characterization allows us to filter out from consideration all the itemsets such that neither they nor any of their supersets will form valid rules.

Our results show that using itemset pruning in the presence of constraints decreases the number of resulting maximal itemsets and also improves the time taken for the process of itemset mining. However, in our current implementation the overall mining time (i.e., the itemset mining time plus the rule construction time) may take longer than the overall mining time when non-itemset pruning is used. This happens when the pruning process throws away too many itemsets that will be re-

quired later in the rule generation stage to calculate confidence of a rule. Further work should be done in this area to improve the rule construction process so that the rule generation time does not overshadow the time savings obtained by itemset pruning during itemset generation.

6.2 Classification Models

We developed a classification system that is based on association rule mining in the WEKA environment. We implemented the CBA model building algorithm [LHM98] and compared the performance of CBA with All Rules Model (ARM) where all the mined rules are part of the model. We developed multiple modes to predict an unclassified instance such as single rule or multiple rule prediction weighed by confidence/support.

6.2.1 Single-Valued

We studied building an association rule based classifier and particularly tested and evaluated the CBA classifier [LHM98] with different prediction modes. Our results show that ARM performs as well as CBA if not better using the single prediction mode. Multiple rule prediction results in lowered accuracy leading to the conclusion that using many rules is likely to introduce noise that affects accuracy significantly. In some cases, the J4.8 classifier exceeds the performance of both CBA and ARM, especially on datasets that contain numeric attributes. This is expected however, as J4.8 has the ability to handle numeric attributes directly, in contrast with association rules.

6.2.2 Set-Valued

We enhanced the classification system to handle the prediction of set-valued class attributes. We developed two modes, single consequent and all consequent to use in predicting the set-valued class attribute. In single consequent, we use the rule with the most number of items in the consequent that classifies a given instance. While in all consequent, we merge the consequents from all the rules that classify a given instance. We adopted E-Measure to compare the predicted set-value against the expected set-value and quantify the prediction.

In comparison of performance of CBA and ARM over the movie dataset, CBA produces lower E-measure values than ARM. However, the results can be deemed inconclusive because of the high number of unclassified instances in the case of CBA. In the case of ARM, the prediction mode All Consequents (AC) produces lower E-measure values; while with CBA, there is no difference between the two modes. The results with the motif dataset are far more acceptable in terms of the percentage of unclassified instances. Again, CBA produces lower E-measure values with higher number of unclassified instances.

Further work should be done to reduce the number of unclassified instances. When using associative classification to produce an accurate model, we require a large number of rules be generated. We think this problem can be solved if we can mine itemsets for each class label separately, thereby setting different minimum supports for the classes. This should ensure the presence of a good number of strong rules for each class label.

6.3 Adaptive Minimal Support

We modified the association rule mining algorithm to produce rules within a range $[R_{min}, R_{max}]$ given by user input. In this pursuit, we heuristically determined a starting minimum support and adjusted the minimum support using a binary search strategy until we obtained a number of rules within the given range.

Using this adaptive minimum support strategy, we were able to generate rules in the range $[R_{min}, R_{max}]$ consistently. We encountered many cases where the time taken with the binary search approach is greater than the time taken with a linear approach. It is likely that this has happened because our heuristic function to select an initial minimal support produced values that were too high. Further work should be done in determining a starting support that will be closer to the actual support that produces the number of rules desired.

Bibliography

[AIS93] Rakesh Agrawal, Tomasz Imirlinksi, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

[Alv02] Sergio A. Alvarez. Binary search strategy for adaptive minimal support. Personal Communication, 2002.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 September 1994.

[CAM04] Michelangelo Ceci, Annalisa Appice, and Donato Malerba. Spatial associative classification at different levels of granularity: A probabilistic approach. In *PKDD*, pages 99–111, 2004.

[Coh96] William W. Cohen. Learning trees and rules with set-valued features. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 709–716, Menlo Park, August 4–8 1996. AAAI Press / MIT Press.

[cR79] C. J. can Rijsbergen. *Information retrieval*. Butterworths, 1979.

[CYZH05] Jian Chen, Jian Yin, Jun Zhang, and Jin Huang. Associative classification in text categorization. In *ICIC (1)*, pages 1035–1044, 2005.

[FW00] Eibe Frank and Ian H. Witten. *Data Mining.* Morgan Kaufmann Publishers, 2000.

[Gro] GroupLens Dataset. Grouplens research project. http://www.cs.umn.edu/Research/GroupLens. Department of Computer Science and Engineering. Univ. of Minnesota.

[HPY00] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(2):1–?, 2000.

[Kaw03] Takeshi Kawato. Multi-expert neural networks for recommendation. Undergraduate Graduation Project (MQP). Worcester Polytechnic Institute, April 2003.

[KPSB00] Dennis Kibler, Michael J. Pazzani, Padhraic Smyth, and Stephen D. Bay. The UCI KDD archive of large data sets for data mining research and experimentation, December 30 2000.

[LAR02] W. Lin, S.A. Alvarez, and C. Ruiz. Efficient adaptive–support association rule mining for recommender systems. *Data Mining and Knowledge Discovery*, 6(1):83–105, January 2002.

[LG94] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In W. Bruce Croft and Cornelis J. van Rijsbergen, editors, *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 3–12, Dublin, IE, 1994. Springer Verlag, Heidelberg, DE.

[LHM98] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Knowledge Discovery and Data Mining Integrating*, pages 80–86, 1998.

[LHP01] Wenmin Li, Jiawei Han, and Jian Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In Nick Cercone, Tsau Young Lin, and Xindong Wu, editors, *ICDM*, pages 369–376. IEEE Computer Society, 2001.

[McJ97] P. McJones. Eachmovie collaborative filtering data set. http://www.research.compaq.com/SRC/eachmovie, 1997. Compaq Systems Research Center.

[Mit97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[Pra04] K. Pray. Mining association rules from time sequence attributes. Master's thesis, Department of Computer Science, Worcester Polytechnic Institute, May 2004.

[Qui93] J. Ross Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[SHM98] C.L. Blake S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[Sho01] Christopher A. Shoemaker. Mining association rules from set-valued data. Technical report, Worcester Polytechnic Institute, May 2001.

[SR03] Christopher A. Shoemaker and Carolina Ruiz. Association rule mining algorithms for set-valued data. In Jiming Liu, Yiu ming Cheung, and Hujun Yin, editors, *IDEAL*, volume 2690 of *Lecture Notes in Computer Science*, pages 669–676. Springer, 2003.

[SS02] Zachary Stoecker-Sylvia. Merging the association rule mining modules of the weka and arminer data mining systems. Undergraduate Graduation Project (MQP). Worcester Polytechnic Institute, April 2002.

[SVA97a] Ramakrishnan Srikant, Quoc Vu, and Rakesh Agrawal. Mining association rules with item constraints. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining, KDD*, pages 67–73. AAAI Press, 14–17 August 1997.

[SVA97b] Ramakrishnan Srikant, Quoc Vu, and Rakesh Agrawal. Mining association rules with item constraints. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining, KDD*, pages 67–73. AAAI Press, 14–17 August 1997.

[Tha06] D. Thakkar. Distance-based sequential data mining for genetic data. Master's thesis, Department of Computer Science, Worcester Polytechnic Institute, 2006. In progress.

[Wie] G. Wiederhold. Stanford movie database. http://www-db.stanford.edu/pub/movies/doc.html.

[YL05] Yongwook Yoon and Gary Geunbae Lee. Practical application of associative classifier for document classification. In *AIRS*, pages 467–478, 2005.

[YLW01] Qiang Yang, Tianyi Li, and Ke Wang. Building association-rule based sequential classifiers for web-document prediction. Technical report, Simon Fraser University, School of Computing Science, 2001.

[ZAC02] Osmar R. Zaiane, Maria-Luiza Antonie, and Alexandru Coman. Mammography classification by an association rule-based classifier. International Workshop on Multimedia Data Mining, 2002.