CLASSIFYING CATEGORICAL DATA

By

Risi Thonangi

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE BY RESEARCH
(COMPUTER SCIENCES)

at the

International Institute of Information Technology
Hyderabad
December 2005

To My Parents

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Classifying Categorical Data" by Risi Vardhan Thonangi, has been carried out under my supervision and it is not submitted elsewhere for a degree.

Date	Advisor

Contents

List of Figures

Abstract

Recent studies in classification have proposed ways of exploiting the association rule mining paradigm. These studies have performed extensive experiments to show their techniques to be both efficient and accurate. However, existing studies in this paradigm either do not provide any theoretical justification behind their approaches or assume independence between some parameters. *In this thesis, we propose a new classifier based on association rule mining. Our classifier rests on the maximum entropy principle for its statistical basis and does not assume any independence not inferred from the given dataset. We use the classical generalized iterative scaling algorithm (GIS) to create our classification model. We show that GIS fails in some cases when itemsets are used as features and provide modifications to rectify this problem. We show that this modified GIS runs much faster than the original GIS. We also describe techniques to make GIS tractable for large feature spaces – we provide a new technique to divide a feature space into independent clusters each of which can be handled separately. Our experimental results show that our classifier is generally more accurate than the existing classification methods. We also prove that ACME reduces to the standard Naive Bayes classifier when there are no statistical dependencies between the variables of the domain.*

Chapter 1

Introduction

Today's world is deluged by data – made possible by our increasing capabilities at collecting and storing information. With rapid digitization of business transactions and knowledge onto the WWW the data that is collected is increasing at a fast rate. This data acts as a representative of the process from which it is collected. These processes range from buying patterns of general public to interactions between astronomic objects. Understanding such processes can benefit us in business planning and market analysis to engineering design and science exploration. The importance of this data is very high, because of the difficulties at understanding the complexities behind the processes that generate this data.

Various fields have cropped up to help humans analyze this data with the help of a computer, and two of the most used today are (1) Machine Learning, and (2) Data Mining. Machine Learning is an area of Artificial Intelligence concerned with the development of techniques which allow computers to learn. The general idea here is as follows – after capturing a good number of the process' input-output pairs, use them to model a system that closely resembles the process that generated it. It can also be looked at as an attempt to automate parts of the *scientific method* which deals with the acquisition of knowledge from available physical evidence. This field is particularly useful, when a human is given the task of understanding a large amount of data or to help automate decision making. It has a wide spectrum of applications including search engines, stock market analysis, classifying DNA sequences, speech and handwriting recognition, game playing and robot locomotion.

Knowledge-Discovery in Databases (KDD) or simply Data Mining, is the practice of automatically searching large data stores for patterns. This field helps humans find *important* trends in a process, where a user defines what a trend/pattern is and under what conditions is it considered important. Such patterns are useful to quickly understand *important* correlations existing in the data and improvise on business or scientific decision making. The large amounts of data which needs to be searched for the patterns makes Data Mining mainly a field of algorithms specializing in efficient memory usage and I/O management.

In this thesis, we look at the problem of classification which falls in both the areas of Machine Learning and Data Mining. Classification is a technique used to automatically classify an unknown object to the class it belongs to, after looking at available object-class pairs. It has been successfully applied to many areas such as medical diagnosis, detecting credit card fraud, speech recognition, hand-writing recognition, etc.

1.1 The Classification Problem

Classification has been an age old problem. Early in the 4th century BC, Aristotle tried to group organisms into two classes depending on whether they are beneficial or harmful to a human. He also introduced the concept of classifying all forms of life for organizing the rich diversity in living organisms. Today's Classification systems, try to find differentiating features between classes and use them to classify unknown instances. It has been recognized as an important problem in data mining [12], among other knowledge discovery tasks and has attracted great attention for the past years in the data mining research community [23, 25, 27, 28, 35, 38]

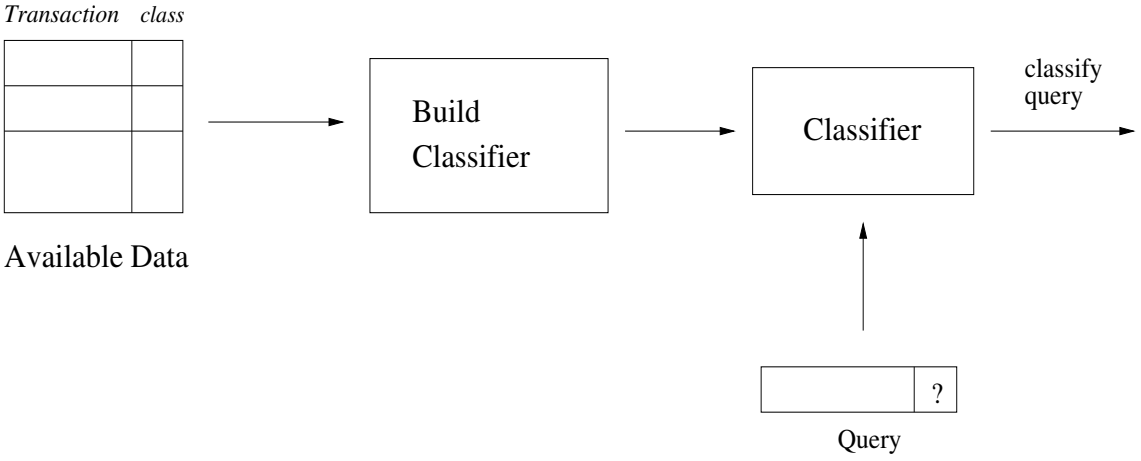

Figure 1.1 Classification System

Figure 1.1 shows the various stages involved in classification. The input dataset called as the "Available Data" consists of $\langle transaction, class \rangle$ pairs. In the first step we use this available data to build a classifier which is used in the second step to classify a new query transaction.

The problem of classification involves finding robust relationships/dependencies between the class variable and the other variables and using them accurately to predict the class of an unseen query. These

relationships are stored as a *classification model* in the form of rules [4, 31], decision trees [30], or mathematical formulae.

Previous studies such as decision trees [30], rule learning [4], naive-bayes [10] and other statistical approaches [24] have developed heuristic/greedy search techniques for building classifiers. These techniques build a set of rules covering the given dataset and use them for prediction. The rules encode the relationships between the class variable and other variables. Machine learning approaches like SVMs [6] do classification by learning boundaries between classes and checking on which side of the boundary the query lies.

Recent studies in classification have proposed ways to exploit the paradigm of association rule mining for the problem of classification. These methods mine high quality association rules and use them to build classifiers [25] [9] [27]. We refer to these approaches as *associative classifiers*. Associatve Classifiers have several advantages – (1) Frequent itemsets capture all the dominant relationships between items in a dataset, (2) Efficient itemset mining algorithms exist, (3) These classifiers naturally handle *missing values* and *outliers* as they only deal with *statistically significant* associations, (4) Developed classifiers are robust since only frequent itemsets which are robust identifiers of a domain are used, and (4) Extensive performance studies [23] [28] have shown such classifiers to be generally more *accurate*. However, existing studies in the associative classification paradigm either do not provide any theoretical justification behind their approaches [23] or assume independence between some *parameters* in the domain [27] [10].

This thesis introduces a new classifier called **ACME** - An Associative Classifier based on Maximum Entropy. The maximum entropy principle is well-accepted in the statistics community. It states that given a collection of known facts about a probability distribution, choose a model for this distribution that is consistent with all the facts but otherwise is as uniform as possible. Hence, the chosen model does not assume any independence between its parameters that is not reflected in the given facts. We use the *Generalized Iterative Scaling (GIS)* algorithm to compute the maximum entropy model.

1.2 Contributions

The main contributions in this work are as follows:

1. we developed ACME, a new classifier for classifying categorical data. ACME combines association rules mined from the input data using the principle of maximum entropy thereby not making any assumptions that were not already inferred from the dataset.

2. we show that the existing maximum entropy framework does not have a solution for every input case. In particular the framework does not have a solution when used with itemsets as features that are not *closed*[1]. We propose a modification to the framework to fix this problem.

3. we provide a modification to the GIS algorithm to learn the new maximum entropy model. The modified GIS is much faster than the original GIS and can also handle cases with *closed itemsets* as input features.

4. we describe techniques to make GIS tractable for large feature spaces – we provide a new technique to efficiently divide a feature space into independent clusters each of which can be handled separately and the global answer can be computed efficiently from the answers of the clusters.

5. we prove that when there are no conditional dependencies between variables in the domain ACME reduces to the standard Naive Bayes classifier.

1.3 Problem Definition

Let $\{a_1, a_2, \ldots, a_n\}$ be the set of categorical attributes required to describe a transaction with an attribute a_j taking one of the categories $\{v_{j1}, v_{j2}, \ldots v_{j|a_j|}\}$. Let $L = \{c_1, c_2, \ldots, c_l\}$ be a set of l classes. A transaction can be represented as $\{v_{1a}, v_{2b}, \ldots, v_{|I|c}\}$. For ease of discussion, we convert such multi-category attributes into boolean valued attributes by including each value as a new attribute. Thus the new set of attributes will be $\{v_{11}, v_{12}, \ldots, v_{1|a_1|}, v_{21}, v_{22}, \ldots, v_{2|a_2|}, \ldots, v_{n|a_n|}\}$ and we represent them as $I = \{i_1, i_2, \ldots, i_{|I|}\}$. A transaction containing items $\{i_{k_1}, i_{k_2}, \ldots, i_{k_j}\}$ is denoted by x_r where r is a *whole number* whose $k_1, k_2, \ldots, k_j$ bits are set to 1 and the remaining bits are set to 0. Hence the set $X = \{x_0, x_1, \ldots, x_{2^{|I|}-1}\}$ represents the set of all possible transactions.

We are provided a dataset of transactions, called as the training dataset and denoted as D, where each transaction is labeled with the class it belongs to. The problem of classification is defined as:

Problem 1 *Given a transaction x to classify (i.e. x is a* query*), label it with class c_i where*

$$c_i \;=\; \underset{c_j}{argmax}\;\; p(c_j|x)$$

where $p(c_j|x)$ stands for the probability of c_j appearing given that x has already appeared. *Bayes formula* allows us to compute this probability from the prior probability $p(c_i)$ and the class-conditioned probability $p(x|c_i)$ as follows

$$p(c_i|x) = \frac{p(x|c_i) * p(c_i)}{p(x)}$$

[1] An itemset is not closed iff it has the same frequency as one of its supersets

Since the denominator $p(x)$ is common for all the classes, it is ignored. $p(c_i)$ is the relative frequency of class c_i in D, which can be trivially calculated by going through the dataset once. Hence, the classification problem is translated to:

Problem 2 *Given a dataset D, learn a probability function $p(\cdot|c_i)$ for every class c_i over the space of transactions X.*

The probability $p(x|c_i)$ can be calculated by measuring the frequency of the transaction x in the training dataset D that are labeled with the class c_i. But, because most training datasets are not large enough, this method would not be accurate if x is infrequent or non-existent in the dataset.

Learning the probability function $p(\cdot|c_i)$ need not be accurate but need to be differentiating between the classes for all c_i, as we are primarily concerned with the problem of classification. In this work we first mine *important* itemsets from the dataset and use these as parameters of the distribution $p(\cdot|c_i)$ and *learn* a proper parameter instantiation which maximizes the entropy of the distribution while satisfying the frequent itemsets. An itemset is *important* if it appears in the training dataset *frequently* and can distinguish between the classes.

1.4 Prefix Trees

Before we go into any details, we look at a special data structure called *Prefix Tree* which is widely used in the mining and storage of categorical data. Specifically, they have been used for mining frequent itemsets [14, 15] and join-processing in set-valued databases [16]. We use prefix trees for the purpose of storing itemsets as they enforce a nice structure onto the itemsets which can be exploited for computational efficiency while keeping the storage costs at a minimum.

Figure 1.4 shows an example prefixtree which stores itemsets $\{(i_1), (i_2), (i_3), (i_1, i_2), (i_1, i_3), (i_2, i_3)\}$. For such storage to be possible, the items $I = \{i_1, i_2, i_3\}$ are ordered in the fashion

$$i_1 \prec i_2 \prec i_3$$

so that every itemset can be represented as a unique sequence when this order is enforced on its items. Once the items in an itemset are ordered according to this order, storage of multiple itemsets can be achieved by storing repeating prefixes only once. In the shown example, the itemsets (i_1, i_2) and (i_1, i_3) share the path from the root node $\{\}$ uptill their immediate parent, hence removing the necessity to store the items on that path multiple times. Note that inside the node only individual items are present, which are eventually what is stored but nevertheless the complete set of itemsets are represented as paths from the rootnode to every other node. Each node holds an integer which stores the frequency of it's corresponding itemset. The children of any internal node are ordered based on the '$\prec$'-order defined

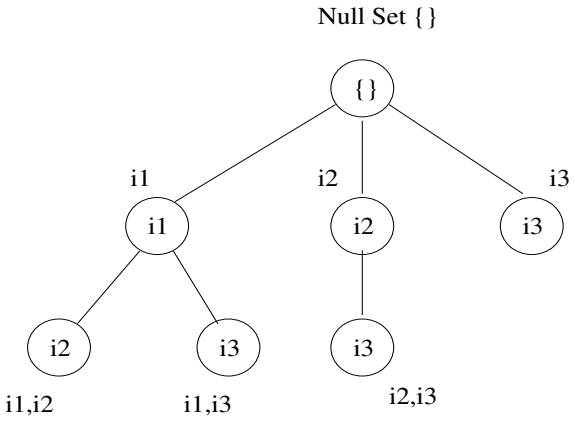

Figure 1.2 Prefix Tree

on the items. Hence to search if an itemset already exists in the tree, the items of the itemset are sorted into a sequence based on '$\prec$' and the first item in this sequence is searched for among the children of the root node. If such a child node exists, the second item in the sequence is searched for among the children of this child node. This search carries on until all the items in the sequence are exhausted, at which time the node we are presently processing will be the one corresponding to the itemset we were searching for. If the itemset was not already stored in the prefix tree, our search will only be able to find a node corresponding to a prefix of the itemset's sequence which is existing in the tree.

1.5 Outline of Thesis

The remainder of this thesis is organized as follows: Chapter 2 reviews related work. Chapter 3 decribes the ACME classifier. Chapter 4 describes problems associated with the existing Maximum Entropy modeling techniques and proposes solutions for these problems. These modifications are also incorporated into the GIS algorithm used for learning the maximum entropy model. Chapter 5 describes a procedure to improve the execution time of GIS by splitting the set of items I into subparts such that the maximum entropy models of these subparts can be combined easily to get the model for the global I. Chapter 6 shows that ACME reduces to the standard Naive Bayes classifier when there are no statistical dependencies between the variables of the domain. Experimental results are given in Chapter 7, followed by Conclusions (Chapter 8) and Bibliography.

Chapter 2

Related Work

Classification is an age-old problem, and several classifiers were suggested in the last few decades. Some important classification paradigms include Decision Lists, Decision Trees, SVMs, Statistical Classifiers and Rule-based as well as Associative Classifiers. For a good review of these classification methods look in [24]. Since our classifier falls in the paradigm of Associative Classifiers, we discuss and analyse the differences between various classifiers belonging to this paradigm.

2.1 Classifiers based on Association Rules

Classifiers based on Association Rules involve the following three steps –

mining Mining Classification Association Rules (CARs),

pruning Pruning the CARs,

building Building a classifier by combining the CARs

classification Classify a given query using the above classifier.

In this section, we explain existing classifiers of this paradigm from the perspective of these four steps and analyze their advantages and disadvantages.

Association Rule (AR) mining [1] algorithms find all rules in a dataset which satisfy a given *support* (denoted as σ) and *confidence* (denoted as τ) thresholds. For example, for the rule $i_1, i_4 \rightarrow i_2$:

$$\sigma \leq P(i_1, i_2, i_4) \ \textit{and} \ \tau \leq \frac{P(i_1, i_2, i_4)}{P(i_1, i_4)}.$$

In the above rule, the set (i_1, i_4) is called as the *precedent* and the set (i_2) is called as the *antecedent*. Classification Association Rules (CAR) are defined as Association Rules with the antecedent restricted to only the class. Hence, for a rule to appear as a CAR in a dataset it should satisfy the following conditions: (1) support and confidence of the rule should be greater than their respective thresholds (2)

The antecedent of the rule should be a class variable. Many efficient algorithms exist today for mining CARs [23,25], and they generally are modifications to well-known Association Rule mining algorithms. Since the output of all the CAR mining algorithms is the same, the first step does not effect the accuracy of the classifer. Hence, while analyzing the classifiers we will not consider this step.

2.1.1 Classification based on Association Rules (CBA)

CBA [25] was the first classifier which took advantage of the Association Rule paradigm. For the first step of mining all CARs, It adopts an apriori-like candidate set generation-and-test approach. For the pruning step, to decrease the number of rules generated during the mining step, it adopts a heuristic pruning method based on a defined rule rank and database coverage.

Definition 1 *Rule Rank: Given two rules r_i and r_j, the rank of r_j is higher than that of r_i (denoted as $r_i < r_j$) if*

1. *the confidence of the rule r_j is higher than r_i, or*
2. *they have the same confidence but the support of r_j is larger than that of r_i, or*
3. *they have same confidence and support values, but r_j has fewer items in its precedent.*

CBA sorts all the CARs according to the above rank definition and selects a small set of high-rank rules that cover all tuples in the database as explained in the following procedure:

1. Pick the rule with the highest rank, go through the dataset and find all the tuples which are correctly classified by the rule.
2. If the chosen rule cannot classify even a single tuple correctly, discard it and go to step 1. Else select the rule and remove all the tuples in the dataset that are correctly classified from consideration and return to step 1.

The above procedure is stopped when all the tuples in the dataset are exhausted or when there are no more rules to consider. The set of rules that are selected are used for building the classifier. A default rule is added to the set of these rules with the antecedent as the majority class of the dataset and the precedent as the null set. CBA sorts the selected rules based on their ranks with the default rule having the lowest rank and uses these rules as the classifier. When a tuple is requested to be classified, it searches the selected rule set from the highest rank and finds out the first rule that matches the tuple. If such a rule is found, then its class label is assigned to the new tuple. Note that every tuple satisfies the default rule. Since the default rule classifies to the class with the highest frequency, any tuple that does not satisfy mined rules is classified to this class.

2.1.2 Association-based Decision Trees (ADT)

ADT [36] proposes to first mine all CARs without a support threshold and uses these rules to build a decision-tree like structure called Association-based Decision Tree (ADT). This ADT is then pruned using techniques that are used for Decision-Tree pruning.

The higher levels of ADT hold general rules and specific rules are placed at lower levels of the tree. Pruning is done in a bottom-up fashion to avoid pruning "good" general rules, i.e. overpruning. If child rules of a parent rule are pruned, their parent rule is made a leaf rule. To decide whether to prune a child rule, we compare the estimated error of ADT on new cases before the pruning with that obtained after the pruning. The estimation is done by the pessimistic estimation for pruning Decision Trees described in [30]. This method guarantees to produce a classifier of the minimum estimated error, with respect to the bottom-up pruning.

2.1.3 Classification based on Multiple Association Rules (CMAR)

CMAR [23] acts in a similar fashion as that of CBA [25], except that it uses multiple rules at the time classification to determine a query's class. This is because using a single rule at the time of classification may not always robustly predict the correct class. For example, suppose we have three rules in the following order:

$$\begin{aligned} i_1 \Rightarrow c_1 \quad &: \quad support \ = \ 0.3, \ confidence \ = \ 0.8 \\ i_2, i_3 \Rightarrow c_2 \quad &: \quad support \ = \ 0.7, \ confidence \ = \ 0.7 \\ i_2, i_4 \Rightarrow c_2 \quad &: \quad support \ = \ 0.8, \ confidence \ = \ 0.7 \end{aligned}$$

To classify a query $\{i_1, i_2, i_3, i_4\}$, CBA picks the rule with the highest confidence that satisfies the given query, i.e. the first rule and uses it to predict the class as c_1. However, a closer look at the rules suggests that the three rules have similar confidences but the rules predicting class c_2 have higher total confidence. These rules also cover the query much better than the first rule. Hence the decision based on the last two rules seems to be more reliable. This example clearly indicates that classification based on a single rule could be prone to errors. To make reliable and accurate predictions CMAR proposes to take into account other rules as well which satisfy the given query.

CMAR runs in two phases: (1) Rule generation (2) classification. A variant of *FP-Growth* method [15] for mining association rules is used to mine CARs. The mined CARs are sorted and pruned in a similar fashion as that of CBA. Given a new query to classify, CMAR selects the set of rules satisfying the query from the mined CARs. The chosen rules are divided into groups according to class labels. All rules in a group share the same class label and each group has a distinct label. For each group, the combined effect of all the rules in that group is calculated by a weighted *chi-squared* measure given in [22] which

was arrived at after extensive experimentation. For more details of CMAR look in [22].

2.1.4 Classification based on Predictive Assocation Rules (CPAR)

Eventhough associative classifiers have high accuracy in general they are substantially slower than the traditional rule-based classifiers like C4.5, FOIL and RIPPER. CPAR tries to combine the advantages of associative classification and traditional rule based classification.

The rule generation in CPAR is similar to that of FOIL (First-Order Inductive Learner) [31], which is a greedy algorithm that learns rules to distinguish positive examples from negative examples. It repeatedly searches for the current best rule and removes all the positive examples covered by the rule until all the positive examples in the dataset are covered. The rule miner of CBA, instead of removing an example after it is covered as in FOIL, maintains a weight for each example and decreases it. This allows covering positive examples multiple times, and generating a larger set of rules for the classifier.

After mining all the rules, each rule is evaluated to determine its prediction power. This is done using the *Laplace expected error estimate* which is defined as follows:

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

where *k* stands for the number of classes, n_{tot} stands for the total nuber of examples satisyfing the rule's body among which n_c examples belong to *c* the predicted class of the rule.

To classify a query, CPAR chooses the best 'k' rules of each class for prediction using the following procedure: (1) Select all rules whose bodies are satisfied by the example (2) From the rules selected in the above step, select the best 'k' rules for Each class by using the *LaplaceAccuracy* measure, and (3) Compare the average expected accuracy of the best 'k' rules of each class and choose the class with the highest expected accuracy as the predicted class.

2.1.5 Large Bayes (LB) Classifier

LB [27] looks at the problem of combining association rules for building a classifier more from a statistical perspective and uses the term *Large Bayes* to describe itself as it happens to reduce to the Naive Bayes classifier when all itemsets selected are of size one only. In the training phase of Large Bayes, frequent itemsets together with their class supports are mined. The class support is a variation of the usual support notion and estimates the probability that the pattern occurs under a certain class label. Given a query to classify, LB builds a local classification model on the fly focussing on the context of this particular query. This approach is in between lazy learning (no work done during training) and

eager learning (full construction of a global model while training).

The actual classification model for a query $\{a_1, a_2, a_3\}$ consists of a formula computing the conditional probability $P(c_i|\{a_1, a_2, a_3\})$ so as to facilitate finding the class with the highest probability of occurrence given the query. [21] shows that this quantity can be estimated using different product approximations where each product approximation implies different independence assumptions between the attributes. For example, $P(\{a_1, a_2, a_3\}, c_i)$ can be estimated as $P(a_1, a_2, c_i) \cdot P(a_3|c_i)$ or $P(a_1, a_2, c_i) \cdot P(a_3|a_1, c_i)$ or any other such approximation. While the first approximation assumes that a_3 is independent of a_1 and a_2 given c_i the second approximation assumes that a_3 is independent of only a_2 given c_i. The first and the second approximations can be calculated as:

$$P(a_1, a_2, c_i) \cdot P(a_3|c_i) = \frac{P(a_1, a_2, c_i) \cdot P(a_3, c_i)}{P(c_i)}$$

$$P(a_1, a_2, c_i) \cdot P(a_3|a_1, c_i) = \frac{P(a_1, a_2, c_i) \cdot P(a_3, a_1, c_i)}{P(a_1, c_i}$$

Note that the second approximation makes less assumptions about the items, but requires more information from the frequent itemsets and their class supports. Hence selecting a certain product approximation for calculating the conditional probability of a class given a query, is equivalent to generating certain itemsets mined in the training phase. Note that the independence assumptions that are made are only valid for the current query and hence this model, given a query, has the convenience to make the assumptions based on the query context unlike that of naive bayes which uniformly assumes complete conditional independence between all the attributes. LB uses the following four rules to pick a product approximation: (1) itemsets should be reliable, hence frequent, also called large in itemset terminology (2) as many itemsets as possible should be used in the product formula (3) the individual itemsets or factors should be as large in size as possible, and (4) the itemsets chosen should be interesting in the sense that they indeed provide more information than their subsets. Detailed information about this classification method can be found in [27, 28].

2.2 Discussion

After CBA [25] there have been many classification algorithms which tried the association rule paradigm for better classification accuracy. Although CBA fared much better than state-of-the-art classifiers in other paradigms, it still lacked robustness in prediction because of its single rule classification mechanism. Later classifiers like LB [27], CMAR [23], and CPAR [38] took multiple rules into account to make more reliable predictions.

Classifiers like LB [27] and Naive Bayes look at the problem of classification as that of building a conditional distribution $P(q|c_i)$ accurately over the query domain. Once this probabilistic model is

The problem of Fully Confident Associations
In the class c_1, let there be a fully confident association in the following way: $$i_1, i_2 \Rightarrow i_3$$ i.e. $P(i_1, i_2|c_1) = P(i_1, i_2, i_3|c_1)$. In such a case, the two itemsets $\{i_1, i_2, c_1\}$, and $\{i_1, i_2, i_3, c_1\}$ will appear in the LB's mined frequent itemsets with the same frequency. Given query $\{i_1, i_2, i_4\}$ to classify, LB goes on building a product approximation without considering the fully confident association between i_1, i_2 and i_3. The final probability of this query is computed as non-zero and it could be very high, if the itemsets taking part in its product approximation have high frequency. But the actual frequency of this query i.e. $P(\{i_1, i_2, i_4\}|c_1)$ should be 0.

Figure 2.1 The problem of Fully Confident Associations

available, classification is done in the following way:

$$\underset{c_i}{argmax}\ P(c_i|q) \quad = \quad \underset{c_i}{argmax}\ P(q|c_i) \cdot P(c_i)$$

Naive Bayes models $P(\cdot|c_i)$ by explicitly assuming conditional independence between all items, thus calculating $P(c_i|q)$ as:

$$P(q|c_i) \cdot P(c_i) \ = \ P(c_i) \cdot \Pi_{i_j \in q} P(i_j|c_i) \cdot \ \Pi_{i_k \notin q}(1 - P(i_k|c_i)).$$

Such explicit assumptions although are very strong, Naive Bayes has continued to perform very well on many datasets. ACME improves Naive Bayes further by considering sets of items which are frequent as well as able to differentiate between the classes and combines these itemsets using the principle of maximum entropy thereby handling conditional dependencies between items.

For more information as to why Naive Bayes performs so well eventhough it assumes such strong relationships, look in [8]. The success of Naive Bayes prompted research into other classifiers which work on similar lines, but have lesser independence assumptions than that of Naive Bayes. Large Bayes [27] is an example of such classifiers. Instead of combining class supports of items, LB uses class supports of frequent itemsets to classify a query. At the time of classification, frequent itemsets are used to form a *product approximation* of $P(\cdot|c_i)$ using four rules given in section 2.1.5. These heuristics may not give us the best product approximation, but nevertheless, the results show that the chosen product does well. Even if the best product is chosen, note that it will still make independence assumptions unless the query itself appears in the frequent itemsets. ACME, however, builds a classification model taking into consideration all the *important frequent itemsets* that are necessary to distinguish between classes and

uses this model for the purpose of classification. This classification model satisfies all the frequent itemsets and also has the highest possible entropy, which ensures that no additional statistical dependencies between items are assumed. Another important disadvantage of LB classifier is discussed in Figure 2.1.

The CMAR [23] classifier is an improvement to the CBA classifier, which uses multiple rules at the time of classification. After finding the CARs which satisfy the given query, the rules are divided into groups based on the class they are inferring. [23] considers the problem of weighing these groups of rules against each other inorder to classify the query. The problem of fully confident associations appears in CMAR as well, since it has no way to track associations between the items.

2.3 Maximum Entropy

The principle of Maximum Entropy has been widely used for a variety of natural language tasks including language modeling [33], text segmentation [3], part-of-speech tagging [32] and prepositional phrase attachment [32]. Ours is the first approach, to our knowledge, which uses it for categorical data classification with mined frequent itemsets as constraints.

The paper [20] discussed the Maximum Entropy model's failure to accommodate *non-closed* constraints and proposed solutions to this problem. It applied Good-Turing discounting to the observed constraint frequencies and ran the classical GIS algorithm on these constraints. For example, a constraint's frequency will be changed to $f_j - \epsilon$, instead of its usual observed frequency f_j before including it for GIS algorithm. In this approach there is no guarantee that the constraints would remain consistent with each other after they have been changed. It also reports results on a *fuzzy maximum entropy framework* in which the objective is to maximize the sum of the entropy function and a penalty function which is heuristically selected to penalize deviations from unreliable(less frequent) constraints. Our solution to this problem, fixes the Maximum Entropy model in a simple fashion, without adding any additional complexity, such that the GIS algorithm will converge and will generate the *correct* maximum entropy distribution. Further, this new solution runs much faster than the classical GIS algorithm(see the Section 7).

Chapter 3

The ACME Classifier

The operation of the ACME classifier can be broadly divided into three phases: (1) *Mining Phase* (2) *Learning Phase*, and (3) *Classification Phase*. The mining phase and the learning phase form the pre-processing step of the classifier, and the classification phase involves using the knowledge gained in this pre-processing step to classify queries. In this section, we discuss these three phases. The notation we follow was introduced in section 1.3. The complete notation is summarized in the figure 3.1.

Notation	Meaning
i_k	Item
I	Set of all items
c_k	Class k
L	Set of all classes
s_j	Itemset
S_k	Set of all frequent itemsets in class c_k
S	Set of itemsets which are frequent in atleast one class
x	Transaction or Query
X	Set of all possible queries (a.k.a *query space*)
$P(s_j)$	Probability of the itemset s_j
σ	Min. support threshold
τ	Min. confidence threshold
$\mathcal{I}(s_j)$	Interestingness of itemset s_j
C_i	Constraints of class i
π	Normalization constant
μ	Parameter of the Maximum Entropy model
$f_j(x)$	Feature corresponding to the j'th constraint

Figure 3.1 Notation

3.1 Mining Phase

The first phase of ACME is the mining phase in which the dataset is examined to find patterns which can be used for modeling the problem domain. As a first step the training data, a set of $\langle transaction, class \rangle$ pairs, is divided into $|C|$ datasets such that transactions labelled with class i are placed in the i'th dataset. See figure 3.2 for a pictorial representation. The problem is how to use these individual datasets to model each class. We propose to use the frequent itemsets appearing in each of these classes as features in modeling the classes. Figure 3.3 gives the various steps involved in building the set of identifiers for each class.

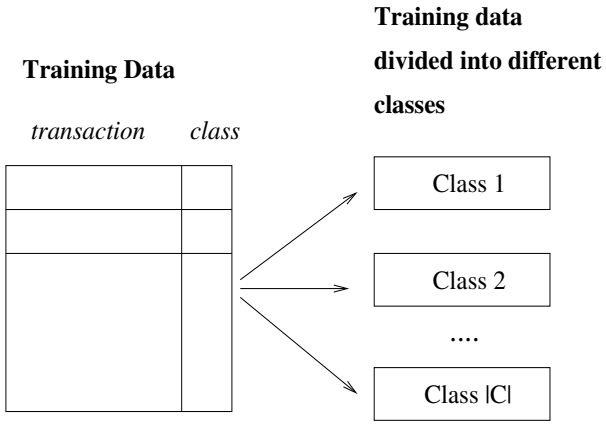

Figure 3.2 Dividing the Training Data

The first step involves finding the constraints for a class, using a standard frequent itemset miner like Apriori [2]. This step is described in the remainder of this section. The second step of pruning constraints is given in section 3.1.1.

Let S_i denote the set of frequent itemsets mined from class i, then:

$$S_i = \{\ s_j \mid P(s_j|c_i) \geq \sigma\ \}$$

where s_j is an itemset and $P(s_j|c_i)$ stands for the support of s_j in the class c_i. Since the mined itemsets are frequent in the class, they form a robust representation of that particular class. However, they are not sufficient for the task of classification as we require features that can differentiate between classes. Hence, to model a class we consider all the itemsets which are frequent in atleast one class. The below equation shows the set of such itemsets.

$$S = \{\ s_j \mid\ \exists c_i\ \ s.t.\ \ s_j \in S_i\ \} \tag{3.1}$$

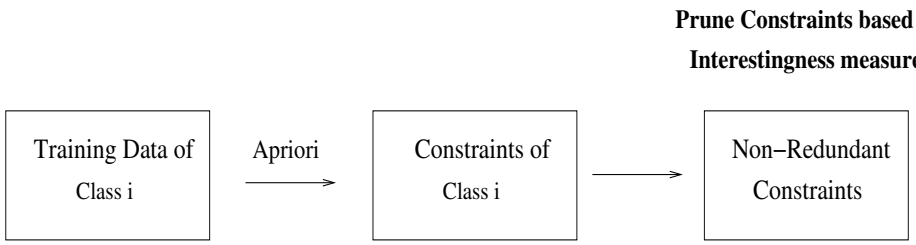

Figure 3.3 Building Constraints for Class i

We use the itemsets in set S as parameters to model each class, such that each itemset s_j together with its support $P(s_j|c_i)$ in class c_i forms a $constraint$ that needs to be satisfied by the statistical model that we build for class c_i. The set of constraints for a class c_i are given as:

$$C_i = \{\ (s_j,\ P(s_j|c_i))\ |\ s_j \in S\ \}.$$

The set S is the union of sets of frequent itemsets $\{S_1, S_2, \ldots, S_{|C|}\}$. To compute S, we first store itemsets of S_i in a prefix tree structure T_i. For an introduction to prefix trees, see Section 1.4. After constructing the prefix trees $T_1, T_2, \ldots, T_n$, all of them are traversed parallely checking for nodes which don't appear in all the trees. Copies of such nodes are created and added to those trees which don't have them. Finally, the input dataset is traversed once again to collect the frequencies of itemsets in classes in which they are new additions.

3.1.1 Pruning the Constraints

While the set of frequent itemsets form a nice summary of a class, they can be huge in numbers. Typically, the number of frequent itemsets in a domain is $O(|I|)$. Since we consider all itemsets which are frequent in atleast one class for the constraint set of each class, we will have more constraints than the frequent itemsets and hence higher computation costs during the learning phase. As a solution to this problem we use an *interestingness measure* to choose only those itemsets among the available ones which we consider to be useful for the task of classification.

Below, we define the term $confidence$ of an itemset s_j for a class c_i.

Definition 2 *We call $P(c_i|s_j)$ as the* confidence *of seeing class c_i in transactions containing the itemset s_j. $P(c_i|s_j)$ (confidence of s_j in c_i) is computed as $P(c_i|s_j) = \frac{P(s_j \cup \{c_i\})}{P(s_j)}$* □

The *confidence*, defined in Definition 2, is considered a measure of interestingness as the higher the confidence value the more the association between s_j and c_i. We denote this interestingness measure by $\mathcal{I}(s_j)$.

$$\mathcal{I}(s_j) = \max_{c_i} \ P(c_i|s_j) \tag{3.2}$$

An itemset s_j is included as a constraint in all the classes, if there exists atleast one class c_i such that s_j has high confidence in c_i (i.e. $P(c_i|s_j) > \tau$, for a given minimum confidence threshold τ). Notice that an itemset s_j is either included as a constraint in all the classes or is not included in any of the classes.

Since $\sum_{c_i} P(c_i|s_j) = 1$, a large value for a particular $P(c_i|s_j)$ would imply that $\forall c_k \neq c_i, \ P(c_k|s_j)$ will be very less. Hence when τ threshold is significantly large and when $P(c_i|s_j) > \tau$, it means that s_j can act as a good distinguishing factor between the classes as occurrence of it in a transaction implies a class c_i with high probability. Classes other than c_i will have lesser probabilities which means that s_j is a good distinquisher between the classes. By including s_j in all the classes, we ensure that this information will remain in the system of constraints.

For example, in a text classification problem every class has the word 'and' occurring frequently. There could be many other such words which will creep into the system of constraints because of their high frequencies. However, since this word is equally frequent in all the classes it does not help in distinguishing between classes. Such identifiers can be removed by using confidence as an interestingness measure.

Note that an effect of this pruning is that the new distribution that will be computed in the learning phase is not the actual distribution of the data. However, it is a good representative for the task of classification.

3.2 Learning Phase

The pruned constraints that were mined for a class in the first phase, form a robust summary of that class and these summaries can be used for the task of classification as each constraint is selected only when it differentiates between classes. With the constraints for each class available, we don't use the input dataset anymore.

ACME uses the constraints of each class to build a statistical model for that class, so that a new query's probability of occurrence in this class can be calculated from its statistical model. Building a probability distribution for a class given its constraints is the problem we consider in this section.

There could be multiple distributions satisfying a given set of constraints. For example, if $I = \{i_1, i_2\}$ and in a particular class if there are two constraints (1) i_1's frequency is 0.5, (2) i_2's frequency is 0.5, then the two distributions P_1 and P_2 given in the figure 3.4 can be possible. A query is said to satisfy a constraint if items of the constraint are present in the query. In the above example, query $\{i_1, i_2\}$

satisfies both the constraints (i_1) and (i_2). If the probability of each possible query is taken as an unknown variable, then a constraint can be represented as a linear equation:

$$P(s_i) = \sum_{s_i \subseteq x \ \wedge \ x \in X} P(x) \tag{3.3}$$

Since a distribution P uniquely determines the probability of every query and vice versa, we can say that P has $|X|$ (=$2^{|I|}$) variables. Hence, to fix P we need to know the values of all these variables. The values of these variables stand for probabilities of their respective queries, and hence to uniquely determine all of them we need so many equations connecting them each other. Since the only information available to connect sets of variables are the constraints (see Equation 3.3), we will need $|X|$ constraints which is the complete power-set itself. Since the mining phase seldom generates the power-set, we will be almost always left with more variables to solve for with lesser equations. Hence, there is a huge possibility that there could be many distributions satisfying the given set of constraints among them we to pick one distribution judiciously.

i_1	i_2	$P_1(.)$
0	0	0.25
1	0	0.25
0	1	0.25
1	1	0.25

i_1	i_2	$P_2(.)$
0	0	0.4
1	0	0.1
0	1	0.1
1	1	0.4

Figure 3.4 Distributions P_1 and P_2

Among these distributions we pick the one with the highest entropy following the well established Principle of Maximum Entropy [13]. This is referred to as the Maximum Entropy model. [32] proves that such a model is unique and can be expressed in the following product form:

$$p(x|c_i) = \pi \prod_{j=1}^{|C_i|} \mu_j^{f_j(x)} \tag{3.4}$$

$$\begin{aligned} where \ \ f_j(x) &= 1 \quad if \ \ s_j \subseteq x \\ &= 0 \quad otherwise \end{aligned}$$

In Equation 3.4, π is a normalization constant which ensures $\sum_{x \in X} p(x|c_i) = 1$.

The learning phase of ACME, as given in figure 3.5, involves finding the parameters $\{\mu_1, \mu_2, \ldots\}$. These values form the parameters of the distribution P. In section 3.2.1, we discuss the Generalized Iterative

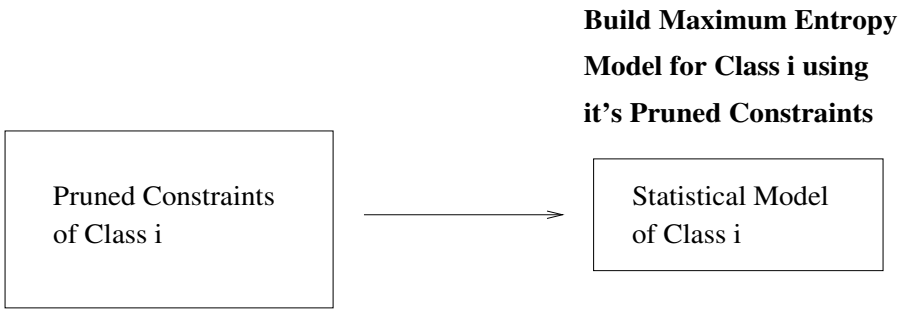

Figure 3.5 Learning Phase

Scaling (GIS) algorithm which is used to find the μ values. The model given in Equation 3.4 is referred to as the classical Maximum Entropy model.

3.2.1 Computing Parameters of the Maximum Entropy Model

The μ_j's in Equation 3.4 are estimated by a procedure called the Generalized Iterative Scaling (GIS) [32]. This is an iterative method that improves the estimation of the parameters with each iteration. The algorithm stops when there is no significant change in the μ_j values. This solution is globally optimal as it has been proved that the search space of μ_j's over which we are searching for the final solution is concave leading to the conclusion that every locally optimal solution is globally optimal [7]. In this section, we present the GIS algorithm and discuss some issues about it.

The GIS algorithm runs with the constraint set C on the domain X computing a model which has the highest entropy and satisfies all the constraints. We call X and C, the parameters of GIS. The GIS algorithm first initializes all the μ_j's to 1, and executes the following procedure until convergence:

$$\mu_j^{(n+1)} = \mu_j^{(n)} \left[\frac{P(s_j)}{P^{(n)}(s_j)} \right] \tag{3.5}$$

where

$$P^{(n)}(s_j) = \sum_{x \in X} p^{(n)}(x) f_j(x)$$

$$p^{(n)}(x) = \pi \prod_{j=1}^{|C|} (\mu_j^{(n)})^{f_j(x)}.$$

The variable n in the above system of equations denotes the iteration number. $P^{(n)}(s_j)$ denotes the expected support of s_j in the distribution that is modelled using $\{\mu_1^{(n)}, \mu_2^{(n)}, \ldots\}$. $P(s_j)$ is the true support of s_j that was calculated from the training dataset. The above iterative procedure is stopped at a

GIS(X,C) :

1. set $\mu_i = 1 \ \forall \ i = 1, 2, \ldots, |C|$
2. set $max\_iterations = 100$
3. for $i = 1$ to $max\_iterations$ do
4. for each x in X
5. set $P^{(i)}(x)$ = product of μ_j's s.t. $s_j \subseteq x$
6. for each $(s_i, P(s_i)) \ \in \ C$
7. set $P^{(i)}(s_i)$ = sum of $P^{(i)}(x)$, $\forall x$ where $s_i \subseteq x$
8. for each μ_j, where $j = 1, 2, \ldots, |C|$
9. set $\mu_j^{(i+1)} = \mu_j^{(i)} \left[\frac{P(s_j)}{P^{(i)}(s_j)} \right]$
10. return $\{\mu_1, \mu_2, \ldots\}$

Figure 3.6 Learning the probability distribution $P(\cdot)$

particular n, when the distribution that is modelled by $\{\mu_1^{(n)}, \mu_2^{(n)}, \ldots\}$ satisifies all the constraints.

The GIS algorithm is given in Figure 3.6. For the proof of convergence of GIS, see [7]. The algorithm starts by first instantiating all μ_j's to 1 (step 1), and then building a distribution $P^{(1)}(x)$ over all possible x (steps 5-6). Next, the support for each constraint is calculated (steps 7-8) and based on them the μ_j values are updated (steps 9-10). The running time of each iteration in GIS is dominated by the steps 5-8. Building the distribution involves going through the complete transaction set X and generating the probability for each query in this set. To generate the probability of a query, the constraints which satisfy this query need to be found out and this operation takes $O(|X| \cdot |C|)$. Steps 7-8 involve finding the support of each constraint in the distribution $P^{(n)}(\cdot)$ which again involves $O(|X| \cdot |C|)$ amount of computation. Since the number of iterations upto which the algorithm is run is $max\_iterations$, the complexity of GIS will be $O(max\_iterations \cdot |X| \cdot |C|)$. Note that the number of iterations is hard-coded instead of waiting for the distribution $P(\cdot)$ to converge as often the process of convergence is slow and the time taken (and there by the number of iterations) could be very high.

The μ values computed in this learning phase form the parameters of the probability distribution. The constraints as well as their computed μ values, together represented as (s_i, μ_i), are stored in a prefix tree for efficient access. Figure 3.7 considers an example where there are 6 constraints. The μ values of these constraints are stored in the prefix-tree such that the node that corresponds to a constraint's itemset will hold the μ value of that constraint. In section 3.3, we will see how such a storage organization of the constraints and their μ values is efficient when classifying a query.

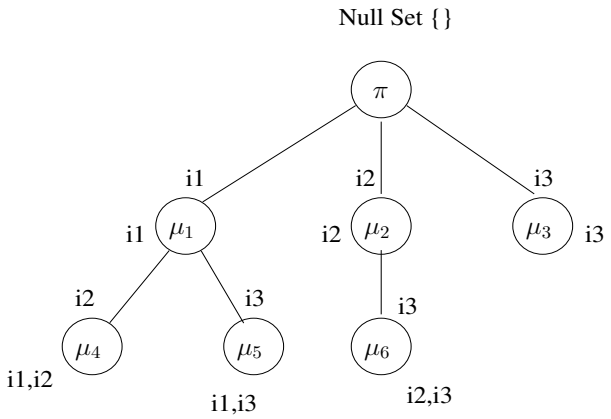

Figure 3.7 Storing Constraints and μ-values in a Prefix Tree

One drawback in the above approach is that the probability model expressed in Equation 3.4 *fails* in some cases [32]. In particular, it fails in the presence of itemsets that are not *closed*. In Section 4.3 we discuss this drawback in detail, and in Section 4.2 we propose a solution to this problem.

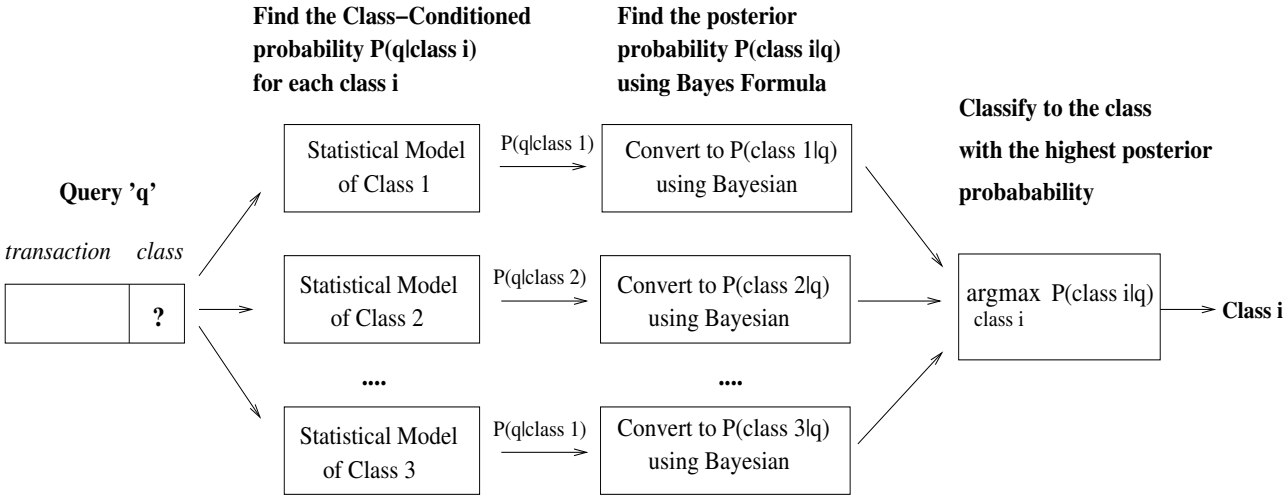

Figure 3.8 Classification Stage

Another drawback in the above approach is that it is not tractable when the set of items I is very large. This is because the complexity of the GIS algorithm is exponential w.r.t. $|I|$. We describe a technique to overcome this drawback in Section 5. It involves partitioning I into independent clusters of items so that GIS can be run on each cluster separately.

Compute-Prob(q,$node$) :

1. if $node.\mu \neq -1$
2. $\quad prob = prob * node.\mu$
3. for each $node.child$ s.t. $node.child[item] \in q$
4. $\quad prob =$ $prob$ * Compute-Prob(q, $node.child$)
5. return $prob$

Figure 3.9 Computing probability of q

3.3 Classification Phase

Figure 3.8 shows the classification phase of the ACME classifier. When a new query arrives for classification, its probability in each of the classes needs to be found out using the statistical model that was learnt and stored for each class. From Equation 3.4, this probability can be calculated by multiplying the μ values of constraints which satisfy the query. Figure 3.9 shows an algorithm **Compute-Prob** which traverses the prefix tree of constraints and their μ values in a depth-first fashion and multiplies all the μs whose constraints satisfy the query q. The input parameters of the algorithm are: (1) the query q, and (2) the root node of the prefix tree. When accessing the root node, the μ value in this node which stores the normalization constant π of the distribution is multiplied to the probability of the query. This algorithm accesses a node only when its corresponding $itemset$ is a subset of the query q because of step 3. Whenever a node is accessed, its μ value is checked if it is positive or not. A μ value is set to -1, then the corresponding itemset is not present in the set of constraints. If μ is +ve, it is multiplied to the probability ($prob$) of the query. The return value of the algorithm is the product of all the μ values whose constrainsts in the subtree rooted at $node$ are satisfied by the query q.

These probabilities are the conditional probabilities of seeing the query q inside a class, which is $P(q|c_i)$. In the second step in the classification phase, refer to figure 1.1, they are inverted using the bayesian rule:

$$P(class\ i|q) = \frac{P(q|class\ i) \cdot P(class\ i)}{P(q)}$$

to calculate the apriori probability of $class\ i$ occurring given the query q. Since for every class, the denominator in the above equation remains the same we can safely remove it from consideration. The modified bayesian is:

$$P(c_i|q) = P(q|c_i) \cdot P(c_i)$$

After finding out this probability for every c_i, in the third step, the query is classified to that class which has the highest posterior, i.e.:

$$\underset{c_i}{argmax}\ P(c_i|q) \quad = \quad \underset{c_i}{argmax}\ P(q|c_i) \cdot P(c_i)$$

Chapter 4

Handling Non-Closed Itemsets

The Maximum Entropy model as given by the product form in Equation 3.4 (of Chapter 3) was first proposed in [26]. It was called as the *log-linear model* since it is a linear combination of the logarithms of its parameters. Because of this property, it is also called as the *exponential distribution*. Various approaches were suggested for estimating the parameters of the log-linear model (see [17] for an introduction). The first approach of iterative scaling was proposed in [34] and subsequently improved in [7] as the Generalized Iterative Scaling (GIS) algorithm. Better iterative scaling algorithms have been proposed since GIS.

In this chapter, we first show that the classical Maximum Entropy model does not have a solution when there are *non-closed* itemsets in the set of constraints (Section 4.1). In such circumstances, the model parameters do not converge under the GIS algorithm. We propose improvements to the product form of the Maximum Entropy model given in Equation 3.4, so as to rectify this problem. Appropriate modifications are made to the GIS algorithm to reflect the changes in the maximum entropy model. The notation we follow was introduced in Section 3. New notation is summarized in Figure 4.1.

Notation	Meaning
s_u, s_v	Itemsets
$[s_u, s_v]$	Fully confident pair of itemsets
C'	Set of closed-constraints
X'	Set of transactions for which prob. is non-zero (a.k.a *closed transactions*)
m	No. of iterations in GIS algorithm

Figure 4.1 Notation

4.1 Failure of Approach due to Non-Closed Itemsets

In this section, we explain why the classical Maximum Entropy model as given by the product form in Equation 3.4 fails in some cases. An itemset is not *closed if and only if* it has the same frequency as one of its supersets. The formal definition of a *non-closed* itemset is given below.

Definition 3 *An itemset $s_v \in S$ is* non-closed *iff*

$$P(s_v) \neq 0 \ \ and$$

$$\exists s_u \in S \ s.t. \ s_u \subset s_v \ \wedge \ P(s_u) = P(s_v)$$

The presence of itemset s_v in a transaction x means that items in s_u will also be present in x. We denote such a pair by $[s_u, s_v]$, a fully-confident *itemset pair.* □

A major disadvantage with the existing Maximum Entropy model is that Equation 3.4 does not have a solution when the system of constraints have *non-closed* itemsets in them. We prove this formally in the following theorem.

Theorem 1 *Equation 3.4 (reproduced in Equation 4.1 below) does not have a solution when the system of constraints has* non-closed *itemsets.*

$$P(x) = \pi \prod_{j=1}^{|C|} \mu_j^{f_j(x)} \tag{4.1}$$

$$where \ \ f_j(x) = 1 \quad if \ \ s_j \subseteq x$$

$$= 0 \quad otherwise$$

Proof 1 *Let $s_v \in S$ be a non-closed itemset. From Definition 3:*

$$P(s_v) > 0 \tag{4.2}$$

Since s_v is a non-closed itemset, $\exists s_u \in S \ s.t. \ s_u \subset s_v \ \wedge \ P(s_u) = P(s_v)$ which means probability of any transaction that contains s_u but not s_v is 0. Let x be a transaction s.t. no other items except the ones in s_u are present in it. It means:

$$i_k \in s_u \Leftrightarrow i_k \in x$$

where i_k is an item in I. Since s_v is strictly a superset of s_u, $s_v \not\subset x$. Hence $P(x) = 0$. From Equation 4.1:

$$P(x) = \pi \prod_{s_w \subseteq s_u} \mu_w = 0$$

$$\Rightarrow \quad \because \pi \neq 0,\ \exists s_w \in S \ \wedge \ s_w \subseteq s_u,\ \ \mu_w = 0$$

$$\because P(s_u) = \sum_{s_u \subseteq x} P(x) = 0$$

$$P(s_u) = 0 \Rightarrow P(s_v) = 0$$

This contradicts Equation 4.2 which says $P(s_v) > 0$*. It follows that Equation 4.1 does not have a solution when the system of constraints has* non-closed *itemsets.* ■

Hence, in cases when the system of constraints has non-closed constraints, the exact solution does not exist in the form of the probability model given in Equation 4.1 and the model parameters will not converge under the GIS algorithm. We propose a modification to the maximum entropy model enabling it to accommodate *non-closed* itemsets in its system of constraints and give a proof of convergence for this new model.

4.2 The Improved Maximum Entropy Model

A *closed* itemset is defined in the following way:

Definition 4 *An itemset* $s_u \in S$ *is is said to be* closed *iff there doesn't exist* s_v *in the system of constraints, s.t.* $s_u \subset s_v$ *and* $[s_u, s_v]$ *form a fully confident itemset pair.* □

The set of all closed constraints of C are denoted C'. The improved product form of the Maximum Entropy model is given as, $\forall x \in X$:

$$\begin{aligned} P(x) = \ & 0 \quad if\ \exists [s_u, s_v]\ s.t\ s_u \subseteq x\ \wedge\ s_v \not\subset x \\ & = \pi \prod_{i=1}^{|C'|} \mu_i^{f_i(x)} \quad otherwise \\ where\ \ f_i(x) = 1 \quad & if\ \ s_i \subseteq x \\ = 0 \quad & otherwise \end{aligned} \tag{4.3}$$

In equation 4.3, note that the model explicitly assigns 0 as probability to queries which satisfy both itemsets in a fully confident itemset pair. Probabilities of queries which don't satisfy any such itemsets are defined using the earlier product form of maximum entropy given in Equation 4.1. However the constraints that are used are only the $closed$ constraints. The non-closed constraints in C are only used

to determine whether the probability of the transaction is 0 or not. The set of transactions for which $P(x) > 0$ are identified by X' and are called as *closed transactions*.

Lemma 1 *The new Maximum Entropy model as given in Equation 4.3 has a solution even when the constraint set has* non-closed *itemsets, and this solution model satisfies the given constraints.*

Proof From the proof of Theorem 1, the presence of non-closed itemsets creates a problem because probabilities of some queries becomes 0 which the product form of the maximum entropy model cannot handle. The model given in Equation 4.3 takes into consideration constraints C' which don't involve *non-closed* constraints. Hence the new Maximum Entropy model has a solution for such input cases.

If two itemsets s_u and s_v form a fully confident pair, i.e. $[s_u, s_v]$, then the information – s_v occurs whenever s_u occurs – is represented in the frequencies of these two itemsets. This information will be lost if s_v and its frequency are removed from the set of constraints. To make up for this loss, the new model that we propose sets to zero the probabilities of all those transactions which satisfy s_u but don't satisfy s_v thus enforcing that $P(s_u) \equiv P(s_v)$. Hence the distribution that is generated by the new Maximum Entropy model also satisfies those constraints which were removed. □

In the remaining part of this section, we give formal proof that the improved product form given above has the highest entropy while satisfying all the constraints, in addition to allowing non-closed itemsets in them. We first prove that the classical maximum entropy model, as given in Equation 4.1 has the highest entropy whenever it has a solution. An important point to note in this proof is that, the feature functions $f_i(\cdot)$'s are input by the user and are not defined by the maximum entropy model itself.

Theorem 2 *If there exists a positive probability function*

(1) of the form $P(x) = \pi \prod_{j=1}^{|C|} \mu_j^{f_j(x)}$, *and*

(2) satisfies constraints $C = \{ P(s_i) \mid \forall i = 1, 2, \ldots, |C| \}$ *in the following way*

$$\sum_{x \in X} f_i(x) \cdot P(x) = P(s_i),$$

where $f_i()$'s are input by the user, then it maximizes the entropy $H(P) = -\sum_{x \in X} P(x) \cdot \log P(x)$ and is unique in doing so.

Proof 2 *Proof outline available in [7].*

Let $Q(\cdot)$ be any probability function satisfying the constraints. $P(\cdot)$ is the probability distribution that satisfies the constraints and also can be represented as the product form given in Equation 4.1. Then

$$H(P) = -\sum_{x \in X} P(x) \cdot \log P(x)$$

$$= \log \pi \cdot (\sum_{x \in X} P(x)) \ + \ \sum_{i=1}^{|C|} \log \mu_i \cdot (\sum_{x \in X} f_i(x) \cdot P(x))$$

Since P and Q are probability distributions $\sum_{x \in X} P(x) = \sum_{x \in X} Q(x) = 1$. Also, the term $\sum_{x \in X} f_i(x) \cdot P(x)$ for a particular 'i' is the frequency of the itemset 'i' in the probability distribution P which is the frequency of the constraint i. Since the distribution Q satisfies all the constraints, the frequency of itemset i in it will be the same as that of P. Hence, $\sum_{x \in X} f_i(x) \cdot P(x) = \sum_{x \in X} f_i(x) \cdot Q(x)$. Therefore,

$$\begin{aligned}
H(P) &= \log \pi \cdot (\sum_{x \in X} Q(x)) \ + \ \sum_{i=1}^{|C|} \log \mu_i \cdot (\sum_{x \in X} f_i(x) \cdot Q(x)) \\
&= -\sum_{x \in X} Q(x) \cdot \log P(x) \\
&= -\sum_{x \in X} Q(x) \cdot \log Q(x) + \sum_{x \in X} Q(x) \cdot \log \frac{Q(x)}{P(x)} \\
&= \ H(Q) + \sum_{x \in X} Q(x) \cdot \log \frac{Q(x)}{P(x)} \\
H(P) - H(Q) &= \sum_{x \in X} Q(x) \cdot \log \frac{Q(x)}{P(x)} \\
H(P) - H(Q) &= K[Q, \ P] \geq 0
\end{aligned}$$

$K[Q, \ P]$ in the above equation stands for the Kullback-Leibler Divergence between the distributions Q and P which is always positive [19]. Hence any distribution Q which satisfies the constraints will have entropy less than distribution P. Hence, the probability distribution taking the product form given in 4.1 will have the highest entropy in addition to satisfying the constraints C. ■

The above theorem proves that the classical Maximum Entropy model as given in the product form Equation 4.1 has the entropy maximized whenever it has a solution. However, this model does not have a solution for all input cases (Theorem 1).

Theorem 3 *Equation 4.3 refers to the model satisfying the given constraints even when it contains non-closed itemsets and has the highest entropy.*

Proof 3 *From Lemma 1, the model given in Equation 4.3 has a solution irrespective of the presence of* non-closed *itemsets in the constraint set C and this solution satisfies all the constraints. The remaining transactions are constrained to have zero-probabilities, because they satisfy fully-confident itemset*

pairs. Hence, they are not considered for entropy calculations. Since the classical Maximum Entropy model is used to define probabilities for the remaining transactions, which are the only ones considered for entropy calculations, from Theorem 2 their entropy will remain maximum. Hence, the improved model suggested in Equation 4.3 will have the entropy maximized while satisfying the given constraints.

4.3 Computing parameters for the Improved Maximum Entropy Model

Instead of the usual *parameters* X and C, the GIS algorithm is run with C' on the domain X' to build the new Maximum Entropy model. It first initializes all the μ_j's in C' to 1 and executes the GIS procedure until convergence:

$$\mu_j^{(n+1)} = \mu_j^{(n)} \left[\frac{P(s_j)}{P^{(n)}(s_j)} \right] \quad \mu_j \in C' \tag{4.4}$$

where

$$P^{(n)}(s_j) = \sum_{x' \in X'} p^{(n)}(x') f_j(x')$$

$$p^{(n)}(x') = \pi \prod_{j=1}^{|C'|} (\mu_j^{(n)})^{f_j(x')} \quad \forall x' \in X' \tag{4.5}$$

Theorem 4 *The modified GIS as given in Equations 4.4 and 4.5 converges to the desired Maximum Entropy model.*

Proof 4 *Same as that of the classical GIS given in [7], except that the transaction set would be X' instead of X and constraint set is C' instead of C.* ■

4.3.1 Algorithm to build the set of Closed Constraints

In our approach, we efficiently compute the closed constraint set C' by storing all the constraints (i.e. the itemset and its frequency) of C in a prefix-tree and identifying those constraints whose frequency is same as one of its subsets. Such constraints are flagged as non-closed constraints, which are not used in the learning stage of P.

Lemma 2 *An itemset s_i will have $|s_i|$ number of immediate subsets.*

Lemma 3 *Let s_a be an immediate subset of s_b, s.t. $s_b = s_a \cup \{i_k\}$. The set of immediate subsets of s_b (denoted by $\mathcal{S}_b$) can be built from that of s_a (denoted by $\mathcal{S}_a$) in the following manner:*

$$\{s_c \cup \{i_k\} \mid \forall\ s_c \in \mathcal{S}_a\} \ \cup \ s_a$$

Find-ClosedCons ($node\_cur$**,** $node\_parent$**,** $Subsets$**,** $path\_items$**) :**

1. set $node\_cur.flag = CLOSED$
2. for each $subset$ in $Subsets$
3. $\quad subset = subset.child[node\_cur.item]$
4. $Subsets = Subsets \cup node\_parent$
5. for each $subset$ in $Subsets$
6. $\quad$ if $subset.freq = node\_cur.freq$ then
7. $\quad\quad node\_cur.flag = NON\_CLOSED$
8. $\quad\quad$ return
9. $path\_items$ = $path\_items \cup node\_cur.item$
10. for each $child$ in $node\_cur.child[]$
11. $\quad C'$ = Find-ClosedCons($child$, $node\_root$, $Subsets$)
12. $C' = C' \cup (\ path\_items,\ node\_cur.freq)$
13. return C'

Figure 4.2 Algorithm to flag *non-closed* constraints in the prefix-tree

The above lemma says that $\mathcal{S}_b$ can be computed from $\mathcal{S}_a$ when s_a is an immediate subset of s_b. The complexity of this calculation is $|\mathcal{S}_b|$ which is the same as $|s_b|$ (from Lemma 2). We use this result to build the set of Closed Constraints efficiently.

The algorithm 'Find-ClosedCons' to achieve this is given in Figure 4.2. Its overall structure is a recursive depth-first traversal of the prefix-tree and it takes the following input: (1) current node that is being processed during recursion ($node\_cur$), (2) parent of the current node ($node\_parent$), (3) nodes which correspond to the immediate-subsets of the parent-node's constraint, and (4) items in the nodes appearing in the path of the current node. Find-ClosedCons returns the closed constraints as well as flags their corresponding nodes in the prefix tree as $CLOSED$. As a first step the current node is flagged as $CLOSED$. The parent node's corresponding itemset will be an immediate subset of the current node's itemset. Hence, the current node's set of immediate subsets $\mathcal{S}$ is computed from that of its parent (steps 2-4). After building $\mathcal{S}$, the frequency of each subset is matched against to that of the current node (steps 5-6). If a positive match is found, the current node is flagged as NON_CLOSED (step 6) as we have found a subset of the current node with equal frequency. The algorithm returns in such a case, as all itemsets occurring in the subtree rooted at the current node also would be *non-closed*. Hence, if a node in the prefix tree is flagged as *non-closed* it implies that all itemsets appearing further below this node are also non-closed irrespective of their flag values. But if there was no positive match, the algorithm runs for these nodes as well to check if they are non-closed and collects the closed constraints among them (C' in step 11). Also, the constraint corresponding to the current node is added to C'.

IS-A-ClosedTrans ($node\_cur$**,** x**) :**

1. if $node\_cur.flag = NON\_CLOSED$
2. return NON_CLOSED
3. for each $item$ where $node\_cur.child[item]$ exists
4. if $item \in x \wedge$ IS-A-ClosedTrans($node\_cur.child[item]$, x) = NON_CLOSED
5. return NON_CLOSED
6. return $CLOSED$

Figure 4.3 Algorithm to check if a Transaction is $closed$

4.4 Algorithm to Build the Set of Closed Transactions

A transaction is said to be *closed* if it does not satisfy any *non-closed* constraints. We make use of the prefix-tree that was constructed in the previous section (Section 4.3.1) which had nodes corresponding to *non-closed* constraints flagged. Algorithm 'IS-A-ClosedTrans' given in Figure 4.3 shows how to find if a given transaction x is $closed$ or not. It takes as inputs: (1) the transaction x, and (2) the prefix-tree of constraints that was constructed in Section 4.3.1 with the nodes corresponding to the *non-closed* constraints flagged. First it checks if the current node is a *non-closed* constraint (steps 1-2). If the node is $CLOSED$, then it traverses all the subtrees which satisfy the given query and checks if the transaction x satisfies any NON_CLOSED constraints in those subtrees (steps 3-5). When no match is found, the algorithm returns saying that the transaction is $CLOSED$.

4.5 Improved GIS Algorithm

Figure 4.4 shows the Improved GIS algorithm used for learning μ values of the new maximum entropy model given in Equation 4.3. This algorithm takes three inputs: (1) the complete set of transactions X which is pruned to X' in the algorithm, (2) the prefixtree $PTree$ whose nodes are flagged to $CLOSED$ or NON_CLOSED by the algorithm Find-ClosedCons (see Figure 4.2), and (3) the set of *closed* constraints C' returned by Find-ClosedCons. First, the algorithm initializes the probabilities of all non-closed transactions to zero (steps 1-2) and all μ_j's to 1. The next step is similar to that of what is done in the old GIS algorithm: Iteratively scale the μ's until the probability distribution satisfies the constraints C'. The final μ's that are learnt are returned by this function.

4.5.1 Time Complexity

The time complexity of the modified GIS algorithm is $O(m * |X'| * |C'|)$ where m is the number of iterations used. Notice that $|C'|$ and $|X'|$ will be smaller than or equal to $|C|$ and $|X|$ respectively. In cases

Improved-GIS (X, $PTree$, C') **:**

1. for each $x \in X$
2. $P^0(x) =$ IS-A-ClosedTrans ($node\_cur$, x)
3. set $max\_iterations = 100$
4. set $\mu_j = 1, \ \forall j = 1, 2, \ldots, |C'|$
5. for $i = 1$ to $max\_iterations$ do
6. for each x in X, where $P^{i-1}(x) \neq 0$
7. set $P^{(i)}(x)$ = product of μ_j's s.t. $s_j \subseteq x$
8. for each $(s_i, P(s_i)) \in C'$
9. set $P^{(i)}(s_i)$ = sum of $P^{(i)}(x)$, $\forall x$ where $s_i \subseteq x$
10. for each μ_j, where $j = 1, 2, \ldots, |C'|$
11. set $\mu_j^{(i+1)} = \mu_j^{(i)} \left[\frac{P(s_j)}{P^{(i)}(s_j)} \right]$
12. return $\{\mu_1, \mu_2, \ldots\}$

Figure 4.4 Improved GIS Algorithm

where some *non-closed* itemsets are removed from C, the size of X' will be significantly smaller than the decrease in the size of C. This is because of the reason that for every *non-closed* itemset removed from C, an exponential number of transactions are removed from X. In our experiments (Chapter 7), we found that the size of X' when compared to X is on an average 10 times smaller. Hence, this new modified GIS algorithm not only computes the correct Maximum Entropy model but also has the advantage of running much faster than the classical GIS algorithm.

Chapter 5

Improving the execution time of GIS

The running time complexity of GIS is $O(m * |C| * |X|)$, where m is the number of iterations required by the GIS for convergence. $|C|$ is typically exponential in $|I|$ as there is one constraint for every frequent itemset mined: The set of frequent itemsets are *generally* exponential. $|X|$ is the set of all possible transactions and hence equals $2^{|I|}$. Hence, the two important parameters which influence the execution time of GIS are $|C|$ and $|I|$.

In Section 3.1.1 (Chapter 3), we presented a method to prune the constraints set C by checking if a constraint is able to differentiate between classes or not. We were also able to decrease the set of transactions X in the learning stage of the classifier (Section 4.4 of Chapter 4.3). In this Chapter we describe a procedure to split the set of items I into smaller mutually-exclusive and collectively exhaustive sub-parts, each of which can be handled separately to produce the global distribution.

5.1 Decomposing the domain I

The effects of Large $|C|$ values on the running time of GIS can be overcome by pruning them using *confidence* interestingness measure. However the algorithm still has to go through the entire possible transaction set X for every iteration in the iterative scaling step. The transaction set X is the power-set of I, and hence it's size can be quite large for even modest values of $|I|$. To decrease the effects of $|I|$ on the running time of GIS, we divide it into *clusters* $\{I_1, I_2, \ldots, I_k\}$ which are mutually exclusive and collectively exhaustive so that the GIS algorithm can be applied to each cluster and the final global distribution can be built by combining the outputs of GIS for these clusters.

Let C_i denote the set of constraints for class c_i obtained. The division of the set of items I into $\{I_1, I_2, \ldots, I_k\}$ is in such a way that a constraint in C_i with itemset s_j is fully contained in only one I_i. Such a division of I ensures that I_i and I_j, where $i \neq j$, will not have any item in common and $\bigcup_i I_i = I$. The items of I_i are said to be independent of items in I_j, for every i and j, since there exist no constraints which overlap with both I_i and I_j. If there were items in I_i which had dependency

relationships with items in I_j, then there should be atleast one constraint containing these items which would have stayed through the pruning process to remain in the final set of constraints. The final global distribution over I can be obtained by combining the local distributions of $\{I_1, I_2, \ldots, I_k\}$ in a naive-bayes fashion.

In the above discussion, I_i and I_j will be independent of each other if there is no constraint which has an overlap with both I_i and I_j. Two cases can exist for such a constraint's absence:

1. The constraint was pruned away based on the interestingness measure $\mathcal{I}(\cdot)$.
2. The constraint was not frequent enough in any of the classes to enter the system of constraints.

If a constraint was not frequent enough, the statistical relationships it encodes are not strongly represented in the dataset. In such a case, we assume that these relationships are not important and the classifier will not lose much by pruning them away and using only those relationships which are strongly represented in the dataset. On the other hand, if the constraint was removed based on the interestingness measure $\mathcal{I}$, it means that the constraint was not *distinguishing* enough between the classes. This means, although the itemset was frequent enough in atleast one of the classes, it is not important for the task of classification.

Chapter 6

ACME as a generalization of Naive Bayes

Even though Naive Bayes assumes complete independence between items and combines their individual frequencies to find the probability of the query in a class, it has surprisingly good performance in many classification problems which have clear attribute dependences [4] [18]. ACME finds sets of items which are frequent as well as able to differentiate between the classes and combines these itemsets using the principle of maximum entropy there by handling conditional dependencies between items. In this chapter, we consider the problem: "Are the classifiers Naive Bayes and ACME same when the attributes in the domain are completely independent?". The answer is yes and we show how it is the case.

Argument 1 *A constraint* (s_i, f_i) *is said to be redundant if the frequency of* s_i *in the distribution* learnt *without* (s_i, f_i) *is the same as* f_i*.*

Let a model $P(\cdot)$ is learnt using the Maximum Entropy framework with the following constraints $C = \{(s_1, f_1), (s_2, f_2), \ldots, (s_{|C|}, f_{|C|})\}$ and $P'(\cdot)$ with the constraints $C \backslash (s_i, f_i)$. If $P'(s_i) = f_i$, then the set of constraints $C \backslash (s_i, f_i)$ implicitly enforce (s_i, f_i) eventhough it's frequency is not explicitly present as a constraint. Hence (s_i, f_i) is said to be redundant in C. Also, the distributions P and P' will be the same in such a scenario, i.e. $P \equiv P'$.

Suppose we are given a distribution which has no attribute dependencies and let $C = \{(s_i, f_i)\}_{i=1,2,\ldots,|C|}$ denote the frequent itemsets mined from this distribution. Let us assume all 1-itemsets have their frequencies greater than the support threshold, and hence the set of constraints C will have all items and their frequencies as well as itemsets of size more than 2 with frequencies above the threshold. Let us denote the constraints in C formed with 1-itemsets as C^1. In the next theorem we prove that the Maximum Entropy model learnt by considering C^1 as the set of constraints is the same the Naive Bayes distribution.

i_1	i_2	...	$i_{\|I\|-1}$	$i_{\|I\|}$	Probability
0	0	...	0	0	μ_0
1	0	...	0	0	$\mu_0 \cdot \mu_1$
0	1	...	0	0	$\mu_0 \cdot \mu_2$
1	1	...	0	0	$\mu_0 \cdot \mu_1 \cdot \mu_2$
...	...	...	...	...	...
1	1	...	1	1	$\mu_0 \cdot \mu_1 \cdot \mu_2 \ldots \mu_{\|I\|}$

Figure 6.1 All possible queries and their product formulas to calculate their probabilities

Theorem 5 *Given a distribution which has no item-item dependencies, the MaxEnt Model of this distribution formed by constraints with 1-itemsets will be the same as the Naive Bayes Model.*

Proof 5 *Each item from $I = \{i_1, i_2, \ldots, i_{|I|}\}$ will form a 1-itemset and together with its frequency will form a constraint. All such constraints make up the set C^1. The possible set of queries and the product expressions each of its probability is given in Figure 6.*

The below equation 6.1 shows the constraint for the i_j'th item that needs to be satisfied by the distribution $P(\cdot)$ and the equation 6.2 shows the constraint that the probabilities of all the queries should add up to 1.

$$P(\{i_j\}) = f_j = \mu_0 \cdot \mu_j \cdot \left(\sum_{s \in 2^{I \setminus i_j}} \prod_{\forall i_k \in s} \mu_k \right) \tag{6.1}$$

$$P(\{\}) = 1 = \mu_0 \cdot \left(\sum_{s \in 2^{I}} \prod_{\forall i_k \in s} \mu_k \right) \tag{6.2}$$

There are a total of $|I| + 1$ constraints and an equal number of variables μ_0 to $\mu_{|I|}$ to solve for. The term in brackets in equation 6.2 is the summation of all possible combinations formed by the variables $\{\mu_1, mu_2, \ldots, \mu_{|I|}\}$ while the term in brackets in equation 6.1 is the summation of all possible combinations formed by the variables $\{\mu_1, \mu_2, \ldots, \mu_{|I|}\} \setminus \mu_j$. From these equations we can expand the term in brackets in equation 6.2 into terms:

$$P(\{\}) = 1 = \mu_0 \cdot \left(\sum_{s \in 2^{I}} \prod_{\forall i_k \in s} \mu_k \right) = \mu_0 \cdot \left(\sum_{s \in 2^{I \setminus i_j}} \prod_{\forall i_k \in s} \mu_k \right) + \mu_0 \cdot \mu_j \cdot \left(\sum_{s \in 2^{I \setminus i_j}} \prod_{\forall i_k \in s} \mu_k \right) \tag{6.3}$$

where the first term stands for all the terms which don't contain the variable μ_j and the second term stands for all the terms which contain μ_j. The bracketed terms in the equations 6.1, 6.3 are the same

and hence the equation 6.3 can be solved as:

$$1 = \mu_0 \cdot \frac{f_j}{\mu_0 \cdot \mu_j} \; + \; \mu_0 \cdot \mu_j \cdot \frac{f_j}{\mu_0 \cdot \mu_j}$$

$$\mu_j = \frac{f_j}{1 - f_j}, \;\; \forall j = 1, 2, \ldots, |I|$$

Finding μ_0:

To calculate μ_0 we solve the bracketed-expression in equation 6.2. Let us assume that $|I| = 2$, i.e. $I = \{i_1, i_2\}$. In such a case,

$$\mu_1 = \frac{f_1}{1 - f_1} \qquad \mu_2 = \frac{f_2}{1 - f_2}. \tag{6.4}$$

From equation 6.3 we can compute the bracketed-expression in the following fashion:

$$\left(\sum_{s \in 2^I} \prod_{\forall i_k \in s} \mu_k \right) = (1 + \mu_1) + \mu_2 \cdot (1 + \mu_1)$$

Substituting μ_1, and μ_2 values from equation 6.4 and then by solving, we get:

$$\left(\sum_{s \in 2^I} \prod_{\forall i_k \in s} \mu_k \right) = \frac{1}{(1 - f_1) \cdot (1 - f_2)}$$

Suppose this result holds true for an arbitrary $|I| = n$, then:

$$\left(\sum_{s \in 2^I} \prod_{\forall i_k \in s} \mu_k \right)_{|I|=n} = \frac{1}{(1 - f_1) \cdot (1 - f_2) \cdot \ldots (1 - f_n)}$$

When $|I| = n + 1$, we can write the expression as:

$$\left(\sum_{s \in 2^I} \prod_{\forall i_k \in s} \mu_k \right)_{|I|=n+1} = \left(\sum_{s \in 2^I} \prod_{\forall i_k \in s} \mu_k \right)_{|I|=n} + \mu_{n+1} \cdot \left(\sum_{s \in 2^I} \prod_{\forall i_k \in s} \mu_k \right)_{|I|=n}$$

Solving this equation, by substituting $\mu_{n+1} = \frac{f_{n+1}}{1 - f_{n+1}}$ we get:

$$\left(\sum_{s \in 2^I} \prod_{\forall i_k \in s} \mu_k \right)_{|I|=n+1} = \frac{1}{(1 - f_1) \cdot (1 - f_2) \cdot \ldots (1 - f_{n+1})}$$

By substituting this result in the equation 6.2, we get:

$$\mu_0 = \prod_{i=1}^{|I|} (1 - f_i)$$

With $\mu_0, mu_1, \ldots, \mu_{|I|}$ values known, we can calculate the frequency of any query q from Figure 6 as:

$$P(q) = \mu_0 \prod_{i_j \in q} \mu_j$$

$$P(q) = \prod_{i=1}^{|I|}(1 - f_i) \cdot \prod_{i_j \in q} \frac{f_j}{1 - f_j}$$

$$P(q) \;=\; \prod_{i_j \in q} P(i_j) \cdot \prod_{i_k \notin q} (1 - P(i_k)).$$

Note that this distribution is the same as that of the Naive Bayes distribution. Hence, given a distribution which has no item-item dependencies, the MaxEnt model of this distribution formed by constraints with 1-itemsets will be the same as the Naive Bayes Model. □

Theorem 6 *ACME reduces to the standard Naive Bayes classifier when there are no statistical dependencies between the items of the domain.*

Proof 6 *Theorem 5 proved that the MaxEnt model reduces to the Naive Bayes model when the constraints considered while learning are of size 1. However, if itemsets of size more than 1 are considered but the distribution that is being modeled has no item-item dependencies then the MaxEnt model still reduces to the Naive Bayes model because all constraints of size more than 1 become redundant as their frequencies will be the same as the frequency which MaxEnt model will learn finally (from Argument 1). Hence the probability distribution that ACME builds for each class will be identical as that of Naive Bayes, which means that both the classifiers will have identical classification models.*

Chapter 7

Experiments

In this section, we show the evaluation of ACME classifier on 11 datasets chosen from the UCI Machine-Learning Repository [29]. We compare ACME with four other state-of-the-art classifiers: (1) the widely known and used decision tree classifier C4.5 [30] (2) a recently proposed associative classifier CBA [25] (3) the naive-bayes classifier and (4) the Tree-Augmented naive-bayes(TAN) classifier [5].

Continuous variables in the dataset were first discretized to interval attributes using entropy based discretization as given in [11]. The same discretized datasets were used for all the classification methods, except for C4.5 which accepts continuous valued variables directly. In all the experiments, accuracy is measured using the 10-fold cross validation. We implemented the classification methods naive-bayes and ACME in C++ and the classifiers TAN and C4.5 were borrowed from the $weka$ machine learning software [37]. The results for the CBA classifier were borrowed from [23]. Experiments were run on an Intel Celeron dual-processor machine with two 3.0GHz processors and running RedHat Linux 9.0.

Dataset	Attrs.	Classes	size	$\|S\|$	largest cluster(I_c)	Cons. in I_c
Australian	14	2	690	354	17	263
Breast	10	2	699	189	17	180
Cleve*	13	2	303	246	15	204
Diabetes	8	2	768	85	15	61
German	20	2	1000	54	20	44
Heart*	13	2	270	115	9	95
Hepatitis	19	2	155	32.1k	17	153
Lymph*	18	4	148	29	9	14
Pima	8	2	768	87	15	55
Waveform	21	3	300	99	18	24

Figure 7.1 Characteristics of the Datasets

Austra	Breast	Cleve	Diabetes	German	Heart	Hepati	Lymph	Pima	Wave
10.1%	8.89%	9.96%	7.42%	9.66%	55.1%	1.3%	12.1%	8.6%	1.3%

Figure 7.2 %'age of queries that remained after the pruning phase

Figure 1 gives the characteristics of the ten datasets used in the evaluation of ACME. The column $|S|$ in this figure, gives the size of the union of all the frequent itemsets extracted from every class (see Figure 3.1). The next column "largest cluster(I_c)" gives the size of the largest cluster in all the classes. Column "*closed* trans." gives the percentage of transactions in the cluster I_c which had non-zero probability (see Section 4.2). Only these transactions were used in the learning phase, instead of the entire domain X. Apart from the transactions which are not *closed*, we also removed transactions which satisfy a constraint whose support was 0. The μ values of such constraints were set to 0 and they were removed from the learning phase. The last column gives the number of constraints in the largest cluster (I_c) among all the classes. If a dataset has multiple classes which contain the largest cluster, it is marked with a '*'. For these datasets, the last two columns give the average calculated over these clusters.

Looking at Figure 1 we can see that even if the decrease in the size of the set of constraints C is modest, the decrease in the size of X is significant. The decrease in the size of X is nearly 99% for datasets $Waveform$ and $Hepatitis$ as the decrease in the size of their constraint set C is very large. Since the complexity of GIS is proportional to $|X|$ (calculated in Section 3.2.1), the new GIS algorithm will run much faster than the classical GIS algorithm in the presence of *non-closed* constraints.

If an attribute a is discretized into $\{a_1, a_2, \ldots, a_z\}$ binary variables, utmost one of these binary variables can be present in any transaction. This fact is not represented in the set of constraints for any class. We explicitly add constraints of the form $(\{a_i, a_j\}, 0)$, for every a_i and a_j, to the set of constraints for every class to ensure the presence of this information. Each constraint means that the probability of seeing a_i and a_j together in a transaction is 0. We call these constraints as *zero-constraints*. This implementation detail improved ACME's accuracy. None of the classifiers that we know, including the ones with which ACME is compared in this section, had the provision to improve their accuracy by using the *zero-constraints*.

In our experiments, the parameters for all the classifiers were set to the standard values as reported in the literature. For CBA the minimum support was set to 0.01 (i.e. 1%) and the minimum confidence is set to 50% with the pruning option enabled. The minimum support threshold for ACME was set to 0.1 and the threshold confidence is set to 0.8. TAN classifier is run with default settings as used in the $weka$ toolkit.

Dataset	NB	C4.5	CBA	TAN	ACME
Australian	84.34	**85.50**	84.9	84.9	85.36
Breast	**97.28**	95.13	96.3	96.56	96.49
Cleve	**83.82**	76.23	82.8	82.5	**83.82**
Diabetes	75.78	72.39	74.5	76.30	**77.86**
German	70.0	70.9	**73.4**	73.0	71.3
Heart	**83.70**	80.0	81.87	81.85	82.96
Hepatitis	80.22	81.93	81.8	81.29	**82.58**
Lymph	83.10	77.0	77.8	**84.45**	78.4
Pima	76.17	74.34	72.9	76.30	**77.89**
Waveform	80.82	76.44	80.0	81.52	**83.08**

Figure 7.3 Accuracies of various Classifiers

Figure 2 gives a detailed description of the datasets employed in the experiments and the accuracies of various classifiers. The winners for each dataset are highlighted in bold font. As can be seen from this table, ACME outperforms all the other classifiers. Out of the ten datasets employed for the testing, ACME wins in five cases. NB wins in 3 datasets and the remaining algorithms perform the best on one dataset each. However, ACME has an additional advantage of handling missing values. Even for other datasets in which ACME does not win, it comes close to the winner.

An important point that needs to be noted is, for the $Hepatitis$ and $Waveform$ datasets, eventhough the new modified GIS algorithm is made to run on only 1.3% of the entire possible domain, it is achieving significant results. ACME was the winner for these two datasets. The reason behind this is that the eliminated portion of the domain is actually set to the correct probability 0. This substantiates our earlier argument that not only ACME runs faster, but also that the distribution it generates will be more closer to the distribution we are trying to mimic. Another important point to note is that the *naive-bayes* classifier is performing better than the remaining classifiers.

Chapter 8

Conclusions

In this thesis, we looked at the problem of classification which falls in the areas of Data Mining and Machine Learning. Classification is a technique used to automatically classify an unknown object to the class it belongs to, after looking at available object-class pairs. We studied in detail the problem of Categorical Data classification where transactions are represented by categorical variables. However, the methods proposed in this thesis can be applied to continuous valued data as well by first discretizing them using discretization methods like [11].

Our main focus here has been on classification based on association rules. Association Rules are a nice way to capture statistical relationships between the class variable and the remaining variables because they are robust identifiers of the problem domain and can handle missing values and outliers. The major issue studied in this thesis is once we mine all association rules what is the best possible way to combine them into a robust classifier.

8.1 Contributions

The main contributions in this work are as follows:

1. we developed ACME, a new classifier for classifying categorical data. This classifier first mines all Classification Association Rules (Association Rules with a class variable in their right side) and uses them to model all the classes. The modeling step is done conforming to the principle of Maximum Entropy which ensures that no additional statistical dependencies between variables are introduced. Our experimental results show that ACME is generally more accurate than existing state of the art classifiers like C4.5, Naive Bayes, TAN (Tree Augmented Bayesian Network) and CBA.

2. we showed that the existing maximum entropy framework does not have a solution for every input case. In particular the framework does not have a solution when used with itemsets as features

that are not *closed*[1]. This is because of the inability of the framework to learn distributions with zero probabilities for some transactions. For such input cases, the learning algorithm GIS that is used to find the parameter values of the maximum entropy model fails to converge. We proposed a modification to the framework to fix this problem.

3. we provided a modification to the GIS algorithm to allow it to learn the new maximum entropy model. The solution involves explicitly setting probabilities of trouble-some queries to zero and building the distribution for the remaining queries. Some association rules are pruned away in this step. Since the association rule set and the transaction set that are used in the learning phase decrease in size, the modified GIS is much faster than the original GIS.

4. we described techniques to make GIS tractable for large feature spaces – we provide a new technique to efficiently divide a feature space into independent clusters each of which can be handled separately and the global answer can be calculated efficiently from the answers of the clusters.

5. we proved that when there are no conditional dependencies between variables in the domain ACME reduces to the standard Naive Bayes classifier.

8.2 Future Work

The work presented in this thesis can be extended and improved in certain aspects. Possible directions are given below:

1. ACME spends the bulk of its time trying to learn μ values for each and every class based on available associatiion rules inferring to that class. This procedure involves high amount of computation and often renders learning infeasible on domains with a large set of variables. A possible solution to this problem is to trade accuracy for speed by building procedures which approximately learn the μ values instead of learning them exactly. Such approximate learning might be able to improve the classification accuracy by having better generalization errors. This could be possible when the support threshold is not set properly in the mining phase allowing non-robust specific rules into the classifier building stage.

2. Decreasing the set of association rules by finding which ones are redundant is also an important problem. This would allow the user to manually look at the rules to gain knowledge about the domain which would otherwise be not possible. There will be significant gains in the complexity of the learning algorithm as well.

3. ACME currently takes two parameters from the user – the support threshold and the confidence threshold. The values for these thresholds can bring a difference to the classifier accuracy as they determine which statistical relationships are to be considered relevant for the task of classification.

[1] An itemset is not closed iff it has the same frequency as one of its supersets

Any help provided to the user at the time of setting these parameters can positively infludence the classifier's accuracy. This will also allow us to give more information regarding the domain to the user.

Bibliography

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Very Large Databases(VLDB)*, 1994.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of 20th Intl. Conf. on Very Large Databases (VLDB)*, September 1994.

[3] D. Beeferman, A. Berger, and J. Lafferty. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177–210, 1999.

[4] P. Clark and T.Niblett. The **cn2** induction algorithm. *Machine Learning*, 2:261–283, 1989.

[5] P. Clark and T.Niblett. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[6] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[7] J.N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470–1480, 1972.

[8] Pedro Domingos and Michael J. Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. In *Proc. of Intl. Conf. on Machine Learning*, 1996.

[9] G. Dong, X. Zhang, L. Wong, and J. Li. Classification by aggregating emerging patterns. In *Discovery Science*, December 1999.

[10] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[11] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Intl. Joint Conf. on Artificial Intelligence(IJCAI)*, pages 1022–1029, 1993.

[12] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy (eds.). *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.

[13] I.J. Good. Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables. *Annals of Mathematical Statistics*, 34:911–934, 1963.

[14] G. Grahne and J. Zhu. Efficiently using prefix-trees in mining frequent itemsets. In *FIMI*, 2003.

[15] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. of 29nd ACM SIGMOD Intl. Conf. on Management of Data*, May 2000.

[16] Ravindranath Jampani. Join processing and selectivity estimation for set-valued attributes. Master's thesis, International Institute of Information Technology, Hyderabad, 2005.

[17] Rong Jin, Rong Yan, and Jian Zhang. A faster iterative scaling algorithm for conditional exponential models. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 2003.

[18] R. Kohavi. *Wrappers for Performance Enhancement and Oblivious Decision Graphs*. PhD thesis, Dept. of Computer Science, Stanford University, 1995.

[19] S. Kullback. *Information Theory and Statistics*. Wiley, New York, 1959.

[20] R. Lau. Adaptive statistical language modeling. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 1994.

[21] P. M. Lewis. Approximating probability distributions to reduce storage requirements. *Information and Control*, 2:214–225, 1959.

[22] Wenmin Li. Classification based on multiple association rules. Master's thesis, Simon Fraser University, 2001.

[23] Wenmin Li, Jiawei Han, and Jian Pei. **CMAR**: Accurate and efficient classification based on multiple class-association rules. In *ICDM*, 2001.

[24] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3):203–228, September 2000.

[25] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. of 4th Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, August 1998.

[26] Bishop Yvonne M. M. Full contingency tables, logits, and split contingency tables. *Biometrika*, 25:383–339, 1969.

[27] D. Meretakis and B. Wuthrich. Extending naive-bayes classifiers using long itemsets. In *KDD*, pages 165–174, 1999.

[28] Dimitris Meretakis, Hongjun Lu, and Beat Wuthrich. A study on the performance of large bayes classifier. In *ECML*, pages 271–279. LNAI, 2000.

[29] C.J. Merz and P. Murphy. **UCI** repository of machine learning databases, 1996. http://cs.uci.edu/ mlearn/MLRepository.html.

[30] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[31] J. R. Quinlan and R. M. Cameron-Jones. Foil: A mid-term report. In *Proc. of European Conference on Machine Learning (ECML)*, pages 3–20, 1993.

[32] Adwait Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, Institute for Research in Cognitive Science, University of Pennsylvania, 1998.

[33] R. Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD thesis, Carnegie Mellon University, 1994.

[34] Brown D. T. A note on approximations to discrete probability distributions. *Information and Control*, 2:386–392, 1959.

[35] Risi Thonangi and Vikram Pudi. ACME: An Associative Classifier based on Maximum Entropy Principle. In *16th International Conference on Algorithmic Learning Theory (ALT)*, pages 122–134. Springer-Verlag LNAI Vol. 3734, 2005.

[36] K. Wang, S. Q. Zhou, and Y. He. Growing decision trees on supportless association rules. In *Proc. of 4th Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, August 2000.

[37] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2 edition, 2005.

[38] Xiaoxin Yin and Jiawei Han. CPAR: Classification based on Predictive Association Rules. In *SDM*, 2003.