A UNIFIED VIEW ON ASSOCIATION AND CLASSIFICATION MINING AND ITS APPLICATIONS

by

DIMITRIOS MERETAKIS

Dipl. Eng. University of Patras

A Thesis Submitted to

The Hong Kong University of Science and Technology

in Partial Fulfillment of the Requirements for

the Degree of Doctor of Philosophy

in Computer Science

January 2002, Hong Kong,

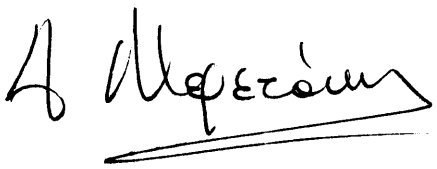
Authorization

I hereby declare that I am the sole author of the thesis

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Dimitrios Meretakis

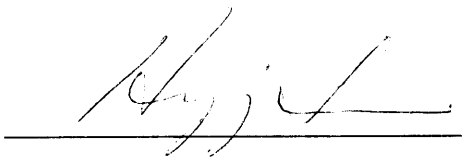


A UNIFIED VIEW ON ASSOCIATION AND CLASSIFICATION MINING AND ITS APPLICATIONS

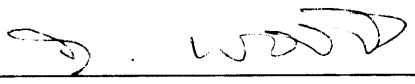
by

Dimitrios Meretakis

This is to certify that I have examined the above PhD thesis
and have found that it is complete and satisfactory in all respects,
and that any or all revisions required by
the thesis examination committee have been made



Prof. Hongjun Lu, SUPERVISOR



Prof. Derick Wood, ACTING DEPARTMENT HEAD

Department of Computer Science

Date: 22 January 2002

ACKNOWLEDGEMENTS

During the last four years, Data Mining has more or less become part of my daily life. Reflecting upon the summer of 1997, I realize that it was the need for knowledge discovery, not only in databases but in real life in general, that led me to choose Hong Kong and HKUST for my postgraduate studies. The trigger of this interesting journey to academic and personal discovery was Dimitris Papadias who very effectively managed to light up my curiosity in his attempt to recruit students from the Computer Engineering Department at the University of Patras. At that time, I was a final year undergraduate student and I wish to thank him for the opportunity he gave me to study at HKUST and to discover the exciting city of Hong Kong.

Being a student at Hong Kong University of Science and Technology was an exciting experience. HKUST offers an enormous amount of resources and provides a "user friendly" structure to help every student reach his full potential and efficiently resolve every problem that might arise. While this is true, however, anyone who has completed a PhD knows that the single most important factor for success during this interesting and challenging journey is a good supervisor.

As a first year postgraduate student at HKUST I knew little about research areas and, consequently, I soon found myself in an infinite loop browsing the web pages of the Computer Science department faculty members to discover nuggets of information that appealed to me. This is how I discovered Knowledge Discovery in Databases and my future supervisor Beat Wüthrich.

Knowledge Discovery was a nice tune in my ears. After all, I like exploring by nature. Beat seemed to be a nice guy, I had even drunk coffee with him once or twice and he looked cool and friendly. As most students at the dawn of their research career, I was simply lacking the knowledge to evaluate Beat from an academic perspective. My intuition said, though, that he was a good supervisor. And I followed my intuition. Being involved in research and with an engineering background that dictates rational decisions based on thorough investigation and on concrete arguments, it is very interesting to realize that the best decision I took over my PhD years was based on pure intuition. (Well, to admit the whole truth, at that time Beat was mainly involved in stock market prediction models and I am sure somewhere in the back of my mind this had a slight influence…)

I wish to thank Beat for the wonderful guidance and encouragement during these years, for adopting a supervising role that allowed me to freely explore research directions that I found interesting, and for being there to offer much needed guidance when I was reaching dead ends. I also wish to thank him for tolerating my student tendency to, at times, incline towards "lazy learning methods" while at the same time motivating me to return to more "eager learning paradigms".

Dr. Hongjun Lu was my supervisor during my final year of studies after Beat left the university to pursue a career in the Financial Services industry. I wish to thank Dr. Lu for his guidance in the final steps of my research as well as for the help he provided whenever needed.

Part of my research was done jointly with Dimitris Fragoudis. The research on the applications of Local Bayesian classification was based on my own research on Local Bayesian classification and Dimitris provided his experience on managing text collections. Dimitris Fragoudis had the core idea on our research on Feature and Instance selection. In both cases, the outcome was the result of joint effort and fruitful collaboration.

I am also thankful to all my friends both in Hong Kong and back in Greece who were there when I needed them. I would like to thank Rachna Kapur for her support. I would also like to thank my friend Kostas Mavroudis for pushing me to the "last mile". Last but certainly not least, I want to express my gratitude to Christina Doukouzgianni for her constant support and encouragement during the final, difficult and tedious phase of writing this thesis.

This personal note would never be complete, without expressing my gratefulness to my family, my brother Giorgos for being there to share thoughts with, my mother Vaia and my father Lefteris for guiding me through all stages of my life while encouraging me to take new initiatives and discover new opportunities.

TABLE OF CONTENTS

A UNIFIED VIEW ON ASSOCIATION AND CLASSIFICATION MINING AND ITS APPLICATIONS

BY

Dimitrios Meretakis

ABSTRACT

Association and classification mining have long been considered as separate research and application areas. The starting point in this thesis is the observation of some key underlying similarities between these two apparently different areas. This observation makes possible the study of well-known classification techniques from an association mining perspective. This different perspective may enable a better understanding of the classification algorithms and help in devising improved or hybrid versions by combining elements from areas that would otherwise be considered incompatible. Our work on building local Bayesian classifiers from itemsets discovered with association mining methods is a result of this different perspective. A new classifier LB, or Local Bayes, is proposed in our work and we show that it is very competitive against established and state of the art classification methods. In addition, LB shows how this different perspective in looking at classification methods can lead to concrete algorithms with superior performance.

Text classification is one of the traditional areas for the application of Machine Learning and Data Mining methods. While traditional classification algorithms have successfully been used for text classification, text collections are particularly suitable for association mining. This is because they usually contain hundreds or thousands of features (for example words) and these features tend to appear in text documents with certain dependencies. Moreover, these dependencies tend to appear locally and for a specific context and the content of words tends to change according to this context. The words "association" and "mining", for example, express different notions when used separately than the expression "association mining". Based on this observation we investigate the applicability of our context-specific classifier called LB in the domain of text classification. Our results are competitive with the most established text classification methods and our approach presents certain advantages, such as a good trade off between accuracy and scalability.

Facing the particular challenges of the text classification problem, we also propose a new text preprocessing method that simultaneously performs both feature and instance selection. This intuitive method is computationally efficient and, most importantly, our experimental results show that it produces outstanding results.

CHAPTER 1. INTRODUCTION AND CONTRIBUTION

The main focus of this thesis is the investigation of the commonalities between association and classification mining methods and the use of these commonalities for the creation of new classification methods based on cross-fertilization between the two apparently disjoint research fields. Our initial work focuses on the review of existing classification methods from a pattern discovery perspective. This creates the basis for the proposal of a classification method that uses an association mining technique as the building block for the construction of efficient and accurate classifiers. We examine the efficiency of our proposed method in both relational databases and in text databases. Our results show that our method is competitive in both fields, both in terms of computational efficiency and in terms of classification quality. Facing the particular challenges of the text classification problem, we also propose a new text preprocessing method that simultaneously performs both feature and instance selection. This intuitive method is computationally efficient and, most importantly our experimental results show that it produces outstanding results.

Our first contribution is the exploration of the similarities between several classification methods. We propose a view that sets the *pattern* (also defined as itemset in the association mining literature) as the basic building block of any classification method over discrete attributes. This different perspective creates an interesting view on the underlying properties, similarities and differences between a diverse set of classification methods such as Decision Tree, Bayesian Network and Nearest Neighbor classifiers as well as some hybrids.

The use of patterns, which essentially are local views of the classification models, opens the door for the exploration of locality and context in classification methods. As a result of this observation, the use of association mining for generating classification methods is briefly described below and examined in a later chapter. Since many apparently different classification methods can be described with patterns as a common denominator, there is potential for the development of hybrid methods that combine characteristics from different algorithms.

Our second contribution directly stems from the pattern-based perspective of classification, which is described above. Given that a variety of classifiers can be described as collections of itemsets and given that efficient association mining algorithms exist for the discovery of itemsets, can we use association mining algorithms for classification? In this thesis, we answer positive to the above question and we describe a new method that uses an association mining algorithm as the underlying engine to discover interesting patterns from the data and then use them for classification.

The main idea behind our work is the combination of the evidence provided by local patterns in order to build a local Bayesian model during classification. This model is in turn consulted to find the most probable class of a new case. In our original publications, [60][61], we called our classifier Large Bayes. In this thesis, we will use the name Local Bayes instead, since its two main properties are the exploitation of locality and the use of Bayesian methods.

Local Bayes is unique in a number of ways. First, it falls in between the eager learning (i.e., Decision Trees, Bayesian Network classifiers) and the lazy learning (i.e., Nearest Neighbor Classifiers) paradigms. This is because it neither creates a complete classification model during training (eager learning) nor it entirely skips the training phase (lazy learning). We call LB a *partially lazy* classifier because it goes through a training phase where it generates local patterns of the data, but the combination of these patterns to build a classification model is performed on the fly during classification.

We prove that the probabilistic models generated by LB during classification are equivalent to instantiations of corresponding Bayesian Networks, which we call local Bayesian Networks. To the best of our knowledge, this is the first work that describes the construction of local Bayesian Network models and it is the first work that investigates the link between association mining and Bayesian Classification. In addition, our results show that LB outperforms many state of the art classifiers while being scalable and efficient.

Our third contribution is the adaptation and use of LB for text classification. Text databases are particularly suitable for association mining because they have a large number of features but each document typically contains only a very small number of words. Additionally, context sensitivity is a very important issue in text classification because of the nature of human language. The application of LB in the text classification domain confirms our intuition that it is particularly suitable for the task because of the above-mentioned properties. Our results show LB to achieve very good classification results while retaining computational efficiency and scalability on large text databases.

Our final contribution comes as a proposed answer to the problems of dimensionality commonly found in text mining domains. As a complement to traditional methods that perform either feature or instance selection, but not both, we describe a method that performs feature and instance selection simultaneously.

Feature selection methods have been an indispensable part of text classification and have been used for decades in Information Retrieval since text databases typically contain tens of thousands of features (words). Instance selection has also attracted attention recently as a means to filter noisy documents and improve the running time of classification algorithms by shrinking the training set.

Our approach is innovative because it combines these two orthogonal methods in a single preprocessing step that often reduces by orders of magnitude both the number of features and the number of examples. In addition, our experiments show that this greatly improves the accuracy of classifiers that use the transformed dataset. To the best of our knowledge, our results are by far the best reported so far by any feature selection or instance selection method.

This rest of this thesis is organized as follows. CHAPTER 2 performs a double role. It presents a review of existing work on association and classification mining, which influenced and formed the basis of our research on the integration of association and classification mining. At the same time, it uses a different perspective in looking at the existing classification methods. This perspective provides the link between association and classification mining, which we exploit in the following chapters.

In CHAPTER 3 we build on the insight gained from the previous chapter and describe Local Bayes, LB, which is a classification method combining the worlds of association and classification mining. We prove that for each probabilistic model built by LB during classification, there is a corresponding Bayesian Network and the model created by LB is an instantiation of the corresponding Bayesian Network. In addition, we experimentally evaluate LB and compare it with well known and state of the art classification algorithms. Our results show that LB is very competitive and in many cases superior to other classification methods.

The focus of CHAPTER 4 is on text classification. While text classification can be thought of as a special form of the general classification problem, it deserves and has received special attention for several reasons, some of which have been briefly discussed above. Our focus is the application of LB to the text classification problem. We present the particularities of text classification, adapt LB to better suit the text classification environment and test two versions of LB against some key text classification algorithms. Our results show that LB is indeed very competitive for the task in terms of both accuracy and computational efficiency.

A common challenge in porting general classification algorithms to the text classification domain is the high dimensionality of text data. In CHAPTER 5, we tackle this problem from a perspective much different from most approaches so far. We describe our method for simultaneous Feature and Instance Selection called *FIS*. Our experiments show that FIS achieves extraordinary results in reducing the number of features, the number of instances and the time required for classification compared to the standard feature selection method based on mutual information. Finally, CHAPTER 6 summarizes our work, presents our conclusions and illustrates possible directions for further work.

CHAPTER 2. ASSOCIATION AND CLASSIFICATION METHODS FROM A PATTERN DISCOVERY PERSPECTIVE

There is a prerequisite that must hold in order for any data mining method to yield useful results. This prerequisite is that there is an underlying model that describes the data where to which the data mining algorithm will be applied. A data set can be thought of as an instance of this model. If the model is known, then the dataset is not needed since it does not contain any more information; the model "explains away" the data.

Assume for example an insurance company that maintains a database with information about its customers. One of the key issues for insurance companies is to increase product density by achieving cross selling, e.g. selling life insurance to the owner of property insurance. Assume now that the company decides to apply data mining techniques on the customer database to identify customers of property insurance that are likely to sign up for life insurance. In doing so the company implicitly assumes that there is a model that defines or predicts what kind of property insurance customers are in need of life insurance (or at least they can be convinced that they are in need of life insurance...). Knowledge of this model leads to accurate identification of potential cross-selling opportunities.

By contrast, applying a data mining technique on the outcomes of a lottery in order to predict the next lucky number is useless since the numbers are random, i.e, there is no model that can predict what is the most likely next number because all number are equiprobable.

The main issue in association mining is the discovery of interesting patterns of the data, so called itemsets. Itemsets, which are formally described below, essentially provide "snapshots" or local views of the underlying model. A sufficiently large number of itemsets then provides a *condensed* representation [55] of the database or otherwise it provides *sufficient statistics* [36] that can be used to discover the underlying model. Classification mining on the other side aims at the discovery of the underlying model that describes the distribution of classes in the data.

In our work, we focus on association mining for classification datasets and we reach the novel and surprising result that classification techniques such as Decision Trees, Decision Rules, Naïve Bayes, Bayesian Networks and Instance-Based classifiers can all be seen as the mining and use of sets of itemsets. This unified representation of apparently different classification methods provides new insights and enables the development of classification techniques that combine characteristics of the previously named methods. Finally, we indicate various research directions that stem from this unified perspective.

2.1 Introduction

The main issue in classification mining is the discovery of a function that maps cases – represented as instantiations of an attribute vector $\boldsymbol{A}=A_1...A_n$ into one of several predefined classes [29]. Among the most prominent classification techniques are decision trees and rule induction [13][74], probabilistic inference [70] and lazy methods such as nearest neighbor [1].

Decision tree and *rule induction* methods aim at finding a small set of highly predictive if-then rules that *completely* define the target classes. Probabilistic methods such as *Naïve Bayes* [26] and *Bayesian Network Classifiers* [32] try to accurately approximate the n^{th} order[1] probability distribution of the attributes using lower order distributions, i.e., sets of attributes with cardinality smaller than n. Both kinds of methods employ a *training phase* where the training data are processed to extract a *global classification model*. Given an instantiation $\boldsymbol{V}$ of $\boldsymbol{A}$ the model provides the result. Since a most processing is done during training, such methods are called *eager*.

For lazy learning methods such as *k-Nearest Neighbor (k-NN)*, the data itself are used for classification. Processing of the data is deferred until classification time when a *local model* tailored towards a specific instantiation $\boldsymbol{V}$ of $\boldsymbol{A}$ is built. This model is subsequently used to derive the result. The two learning paradigms are summarized in Table 1:

	Eager Learning Methods	Lazy Learning Methods
Training phase	1. Process Data 2. Create global model that best describes the data.	1. Do nothing
Testing Phase	1. Consult model 2. Assign most likely class	1. Process Data 2. Create local model to solve given problem 3. Consult model 4. Assign most likely class

Table 1: Differences between eager and lazy learning algorithms

[1] The term "n^{th} order probability distribution" is used with different meanings by different research communities. In our work we use the commonly accepted meaning within the Machine Learning community where the term translates to the joint probability distribution of n variables

In contrast to classification mining, the main purpose of *association mining* [2] is to uncover the relationships between the attributes of the database. Association mining assumes that the data is a set of transactions, each containing a set of distinct attribute values or *items*. The key issue is then to discover all frequent *itemsets*. A *frequent* itemset is a set of items that jointly appear in the data with frequency higher than a user-defined *minimum support* threshold. For classification purposes, we introduce *labeled itemsets*. That is, each itemset has not only one support (i.e., its frequency of occurrence) but rather a support for each class (i.e., its frequency of occurrence when the case is labeled with a particular class).

The rest of this chapter is organized as follows: We first present a survey on several association and classification mining methods. Then we propose an alternative perspective that emphasizes the underlying similarities between association and classification mining. Our perspective is that all classification methods over discrete data can be seen as mining and using labeled itemsets. We generalize this observation to a flexible framework such that diverse classification methods can be seen as instances thereof. Based on this common ground of representation we investigate promising methodologies that combine characteristics from different classification methods and enable the cross sectional use of association and classification mining methods.

2.2 Association Mining

The traditional and simplest setting for the association discovery problem is that of transactional databases of supermarkets. Each transaction contains a set of items bought together by a customer e.g. {diapers, bread, beer}. The goal is to discover frequent itemsets, i.e., sets of items that jointly occur in the database with frequency higher than a user-defined *minimum support* threshold. The *support* of an itemset is the frequency of its occurrence in the database. The itemset ({diapers, beer} 5%), for example means that five percent of the customers bought diapers and beer together. A second, perhaps more informative discovery is that of association rules, which govern the occurrences of items in the database. The rule (diapers⇒beer, support=5%, confidence 60%) states that "60% of the customers who bought diapers also bought beer and this holds in 5% of the transactions". This is probably the most well known association rule and was probably caused by young male parents being sent to the store to buy diapers and deciding to reward themselves for their unavoidable trouble [43].

We now provide some formal definitions. Let $I=\{i_1, i_2, ..., i_m\}$ be a set of literals, called *items*. A *transaction* T is a set of items such that $T \subseteq I$. Let D be a database of transactions. An *association rule* between items in I is an implication of the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \varnothing$. X and Y are sets of items and are called *itemsets*. The rule $X \Rightarrow Y$ holds with *confidence* c in D if $c\%$ of the transactions in D that contain X, also contain Y. The rule $X \Rightarrow Y$ has *support* s if s% of transactions in D contain $X \cup Y$. Similarly, an itemset c has support s if $s\%$ of transactions in D contain all the items of c. Usually we are interested only in itemsets that have support above a given threshold. Such itemsets are called *frequent-itemsets*. The *size of an itemset* is the number of items in the itemset.

Until very recently most association rule discovery algorithms followed a two-step, bottom-up approach introduced by *Apriori* [2]. While newer alternatives [37] promise great computational improvements over the Apriori based methods, the description of the Apriori algorithm provides good insight on the nature of the association rule mining problem, its characteristics and its inherent difficulties. The first step of Apriori involves the discovery of all frequent itemsets of the database, subject to the minimum support constraint. To achieve this, Apriori performs multiple passes over the database. During the k^{th} pass, the frequent k-itemsets (itemsets with size k) are discovered and used as seeds to produce the candidate $(k+1)$-itemsets, which are evaluated against the database in the next pass. The set of frequent itemsets F is used during the second step to produce all association rules with confidence above the minimum confidence threshold.

The key issue and the major bottleneck in Apriori is the discovery of the set *F*. The algorithm performs a general-to-specific search in the space of all possible itemsets, employing minimum support as the only pruning strategy. This is based on the simple observation that if an itemset *l* is not frequent (*support(l)<minsup*) then any of its supersets *l'* will also be not frequent. Although Apriori scales up linearly with the number of transactions, its complexity is exponential in the size of the maximal itemset; In order to discover an itemset of size *s*, it makes *s* passes over the data, but it evaluates all its subsets. This deficiency is particularly visible in data sets with a large number of high-order correlations such as classification data [7][8][53] for which Apriori is highly ineffective, even for high levels of the support constraint.

Earlier research focused on modifications of Apriori, trying to improve its performance, while still following its general two-level approach. The focus of all such methods is mainly on the itemset generation step. Once the itemsets are generated, association rules can be created in a straightforward fashion.

In [69] a hash-based filtering scheme is used to reduce the number of candidate itemsets evaluated against the database during the first passes. This is done by identifying (and removing) some candidate itemsets that cannot be frequent before counting their support. In addition, a reduced version of the database is written in each pass, thereby reducing the cost for successive passes.

The use of partitioning is investigated in [77]. The database is partitioned in memory-sized partitions and a separate set of itemsets is produced for each partition. These sets of itemsets are finally merged and the support of the resulting itemsets is evaluated against the database. This technique, however, always considers at least as many itemsets as Apriori.

Max-Miner [8] follows a more radical approach, trying to discover the set of maximal itemsets only, and therefore improving the running time by orders of magnitude compared to Apriori. The distinguishing property of Max-Miner is that it does not have to evaluate all subsets of an itemset before evaluating the itemset itself. This is achieved by employing look-ahead techniques.

Still, the major issue not addressed by any of these algorithms is that the support threshold alone provides little help in reducing the number of itemsets discovered and it certainly is by no means an efficient way of guiding the search to the most interesting patterns. Incorporating syntactic restrictions to the discovered rules is a partial solution since it gives to the user the option to manually restrict the search. Such work includes [78] that proposes an algorithm for the discovery of association rules in the presence of taxonomies (is-a hierarchies) and Boolean constraints on the items that appear in the rules. A thorough study and categorization of various types of constraints and an algorithm for using them to achieve enhanced pruning of the search space is presented in [67]. Interesting alternatives to the use of the questionable minimum support threshold as a pruning criterion is explored in [65]. The methods proposed in [65], however, work satisfactorily only for low cardinality associations, i.e, containing 2-3 items. The general problem without constraints on the cardinality of the discovered associations has not been addressed.

A radically different alternative to the traditional two-step approach followed by Apriori and its improvements is the *FP-growth* method [37]. This method is based on an alternative representation of the database using a new data structure called *FP-Tree* (Frequent Pattern Tree). FP-trees consist of nodes that represent individual items and transactions are stored as paths within the Tree. By storing more frequent items close to the root and by allowing transactions to share nodes that represent common items FP-trees greatly reduce the size of real world databases. The FP-growth method uses this condensed representation of the original database to discover itemsets and association rules in a fraction of the time needed by Apriori based methods. A recent survey of the most important association rule mining algorithms can be found in [40].

2.3 Classification Mining

In contrast to Association mining, the classification problem has long been studied by the Artificial Intelligence, Machine Learning, and recently by the Database community. The main issue in classification is the discovery of a function that maps cases into one of several predefined discrete classes [29]. A similar problem, long studied in statistics, is the *regression* problem where the class is quantitative. In the sequel, we only consider the classification problem.

The input to a classifier is a database of pre-classified cases or *training cases*. The most common representation for cases is the *feature-value* representation. In this setting, a case can be fully described by a vector of *n attributes* or *features* $\boldsymbol{A}=A_1,A_2,\ldots,A_n$. Each attribute A_j may take any of the values defined in its domain $val(A_j)$, which may be discrete or continuous. In addition, each case is labeled with a *class label* c_i. The database D contains the set of all training cases available to the learner. Given the database D, the classifier aims at constructing a model that can be used to predict the class of any unseen case. This is the *learning phase* of the classifier. During the *classification phase* the classifier is presented with a set of unlabeled cases and the goal is to accurately classify them.

Classification methods can be divided into two broad categories based on the amount of processing of the training data, prior to answering classification queries [1]. *Eager methods* process the training database early on to construct a global model of the data, therefore they are also referred to as *global methods*. *Lazy methods* defer all processing of the data until a new case has to be classified; then they locally learn a model for the data and classify the case based on this model. This allows them to take into account properties of the data that hold in the specific context of the classification query only; therefore, they are also called *local methods*. Many methods fall in between this crisp categorization. Such *hybrid* or *partially lazy* algorithms perform some initial form of processing to produce a transformed (and also more compact) representation of the data. During classification, the transformed data are used to create the local model and subsequently classify the test cases.

In the sequel, we present some popular and successful methods grouped according to the previous categorization.

2.3.1 Eager or Global Classifiers

2.3.1.1 Decision Trees

Decision Trees are probably the most popular and comprehensible models induced from data. Their basic idea is to recursively partition the data until all partitions are pure; i.e., contain cases that belong to the same class [74]. Each internal node of the tree applies a test condition or *splitting rule* on the feature vector and each sub-tree of the node corresponds to an outcome of the condition. A case is classified by traversing the tree, starting from the root, applying the tests of each node and following the corresponding links until a leaf node is reached. The class label of the leaf node is assigned to the case. Internal nodes usually apply *univariate* tests, i.e., the value of a single attribute is tested, although the multivariate case (testing expressions consisting of many attributes) has also been investigated with interesting results [13][14]. In the sequel, we will only refer to the univariate case.

The number of decision trees that is consistent with a given dataset is exponential in the number of features, making enumeration of all possible trees intractable. Therefore, decision tree induction algorithms perform a top-down, greedy search in the space of possible trees. Initially they search for the attribute that produces the best split on the data. One leaf is created for each outcome of this attribute and the cases are partitioned according to their value for this attribute. In subsequent steps, the best splitting rule is considered for each partition separately. The process finishes once all partitions are pure, i.e., all training cases are correctly classified by the constructed tree.

The splitting criteria used by decision tree learners generally aim at reducing the average impurity in the resulting partitions of the data. Entropy is probably the most commonly used measure for the quantification of the impurity in a set of cases. Assume the set of cases in a partition S of the tree. The entropy of the class variable in S is:

$$H(C \mid l) = -\sum_{c_i} P(c_i \mid l) \log P(c_i \mid l)$$

where l is the set of attribute-value pairs that corresponds to the path from the root of the tree to the node that corresponds to S. Entropy takes its maximum value if the distribution of the class in the partition is uniform (i.e, if the attribute test is not predictive) and becomes zero if all cases belong to a single class (maximum predictiveness). The *Information Gain* from a partition of S based on an attribute A_k is:

$$Gain = H(C \mid l) - \sum_{v_j} P(v_j \mid l) \cdot H(C \mid l, v_j)$$

Information Gain can be interpreted as the average reduction in entropy caused by a split. The attribute that maximizes this gain is usually selected for the split. Many other splitting criteria have been proposed, the most popular being Gain Ratio [74] and the Gini Index [13].

Fully grown decision trees completely fit to the data. In real, incomplete and noisy data this prevents the decision tree from generalizing on unseen data, a problem called *overfitting* where the tree models both the underlying concept and the noise. Moreover, the large size of fully-grown decision trees makes them incomprehensible to end-users. Pruning [73][12] is a post-processing function that attempts to remove branches of the tree in order to achieve better performance on unseen data. Decision tree-simplification techniques [12] aim at increasing their comprehensibility by the end user.

Among the most well-known decision tree induction algorithms are ID3 and its successor C4.5 [74], the early but successful Cart [13] and SLIQ [58], which aims at improving the speed of induction from large databases.

2.3.1.2 Rule Induction

Rule learning systems aim at discovering sets of classification rules that predict a target concept. Given a database of training examples, they construct a set of if...then... rules where the antecedents are conjunctions of attribute-values and the consequents are one of the possible classes. Given an unseen case, one or more rules *fire* and they are used to classify the case.

Many of the rule induction algorithms fall into the *sequential covering* framework [64]. Starting from an empty set of rules and the full set of cases, they grow one rule at a time and remove all cases correctly classified by this rule. The process continues until no more cases remain. An evaluation criterion is used to select among alternative rules. Popular criteria include the Entropy of the class on the cases covered by the rule (studied in section 2.3.1.1) and the *m-estimate* [16] of the accuracy of the rule:

$$\frac{N_c + mp}{N + m}$$

where N is the number of cases matched by a rule, N_c the number of cases correctly classified, p is the probability of the class assigned by that rule in the entire set of cases and m is a weighting factor. The ratio N_c/N is the observed probability that a case will be correctly classified, given that it matches the rule. For small values of N the observed probability is overridden by the probability p of the class in the whole data set. The weight factor m determines our confidence on the observed probability. Both the m-estimate and the entropy metrics have been used in CN2 [18], one of the most well known rule induction algorithms.

The output of the rule induction algorithms is usually a *list* of rules. Given a new case, more than one rule may fire. Therefore, a conflict resolution method is needed. Usually the rules are arranged in the order they were created. Given a new case the list is traversed and the class of the first rule that fires is selected. If no rule fires, the default (most frequent) class is selected. This approach is followed by CN2. Another possible approach is the combination of all rules that fire using a majority vote among them.

Decision trees can directly be translated into sets of rules and there is a certain mapping between decision tree and rule induction algorithms. There is one rule for each path from the root to each leaf. *C4.5rules* is a post-processing technique employed by the C4.5 decision tree induction algorithm. It first transforms the tree to the equivalent set of rules, prunes the rules by dropping conditions that do not contribute to their accuracy and sorts the resulting set of rules in order of decreasing accuracy. Decision tree induction algorithms grow the set of rules simultaneously; therefore they can be categorized in the *simultaneous covering* paradigm [64].

It is interesting to note that Decision Trees and Decision Rules are syntactically equivalent. There is a semantic difference, however: The rule set generated by a Decision Tree algorithm has the property that one and only one rule fires for each case. This contrasts to rule sets generated by Decision Rule generation algorithms, where none, one, or more than one rule may fire for a new case.

2.3.1.3 Bayesian Classifiers

Bayesian classification methods approximate the probability distribution $P(\mathbf{A},c_i)$ by making certain *conditional independence assumptions* [10][23][70]. Independence assumptions are usually expressed using directed graphs where the nodes are the attributes and each attribute is independent of its non-descendants given its parents. The independence assumptions allow the theoretically correct computation of the full distribution by its lower order components - provided that the independence assumptions are correct. Such lower order probability distributions are stored in *probability tables* that contain the class distribution for each possible instantiation of the attributes. During classification, the graphical model indicates how the stored low order probabilities can be combined to estimate the desired high-order probability.

Bayesian methods regard attributes as random variables taking values from their domains. From probability theory we recall that two variables x and y are conditionally independent given z, denoted $I(x|z|y)$, if $P(x, y \mid z) = P(x \mid z) \cdot P(y \mid z)$ for all instantiations of x, y and z. Similarly x and y are (marginally) independent if $P(x, y) = P(x) \cdot P(y)$ for all possible instantiations of x and y. The definition also applies for sets rather than single variables. The importance of conditional independence for the approximation of probability distributions is obvious: given (conditional) independence, high-order marginals can be accurately approximated by lower order ones, leading to simple, computationally efficient and more robust models and reducing storage requirements.

Naïve Bayes Classifier (NB) [26] pushes the independence assumptions to the limit making the a-priori assumption that all attributes are conditionally independent given the class. This allows the computation of the probability $P(A,C)$ that the case $A=A_1,\ldots,A_n$ belongs to class C using only the first order conditional probabilities $P(A_i|C)$:

$$P(A,C) = P(C)P(A_1 \mid C)P(A_2 \mid C)...P(A_n \mid C)$$

Classification of A is then done by labeling it with the class c_i with the highest such probability $P(A,c_i)$.

Besides the simplicity of the model and the fact that in most cases the independence assumptions are clearly violated, NB performs surprisingly well in a variety of real domains [32][25] outperforming more sophisticated classifiers. This is mainly because its goal is not to accurately approximate the probability distribution but rather to select the most probable class for each case. Therefore, even if the assumptions are clearly violated, the ranking of the classes may still be unchanged thus yielding accurate predictions [25].

Several extensions of NB aim at relaxing its strong independence assumptions. Among those we can select *Tree Augmented Naive Bayesian classifier (TAN)* [32] *KDB* [76], the *Selective Bayesian Classifier* [54] and *Semi-Naïve Bayesian Classifier* [48]. Following a different approach, A*djusted Probability NB* [81] produces weight adjusted probability estimations and *NBTree* [46] combines Naïve Bayes and Decision Trees.

While all these methods try to increase the performance by relaxing the independence assumptions, they still aim at the generation of global classification models. There is not much work on the use of locality in Bayesian Classification. C*ontext-specific independence* is introduced in [11]. It describes independence relations among variables that hold in certain contexts only. The role of context in feature selection is examined in [20][24][79]. Local structure is exploited in [33] as part of a Bayesian Network Learning algorithm. Nevertheless, the aim in [33] is to use a tree structure to encode locality in a learned global model and not to discover a local model. The only exploitation of locality in Bayesian classification is reported in [39] with *Bayesian Multinets*, which model relationships among variables within each class separately. A more thorough review on all the methods mentioned above can be found in section 3.2.

2.3.2 Lazy or Local Classifiers

2.3.2.1 k-Nearest Neighbor

The *k*-nearest neighbor (k-NN) classifier is a typical and purely lazy learning algorithm [1][22] that generates the classifications directly from the stored training instances. It treats all cases as points in an *n*-dimensional space, where the coordinates are determined by the values of their feature-vector. To classify a new case *A*, it employs a distance function to select the *k* closest cases and combines their evidence.

Several distance functions have been proposed. The most common one is the standard Euclidian Distance. For two cases $A=A_1,A_2,...,A_n$ and $B=B_1,B_2,...B_n$ their distance *d(A,B)* is defined as:

$$d(A,B) = \sqrt{\sum_{m=1}^{n}(A_m - B_m)^2}$$

The simplest method for combining the evidence of the k-closest cases is *majority voting*; the most common class among them is assigned to the new case. A more delicate approach is used by *Distance-Weighted Nearest Neighbor*, where the vote of each case is weighted by its distance from the case under consideration so that the nearest neighbors have greater weight. There are even more sophisticated methods for the combination of evidence (i.e, building a local classifier such as a decision tree from the selected cases). These methods are described in section 2.3.2.3.

The geometric interpretation of the Euclidean distance metric justifies its suitability in domains with numerical attributes. When the attributes are categorical though, Euclidian Distance presents several problems since an ordering is not always defined. A simple metric for symbolic attributes is the so-called overlap metric, which simply counts the number of common attribute values between two cases. A more sophisticated metric for symbolic values is the *modified value difference metric (MVDM),* which is described below.

2.3.2.2 Pebls

Pebls [21] is a lazy learner that is specially designed for domains with symbolic attributes. It also stores all cases and tries to find the closest ones to the query case. The interesting aspect of Pebls is the use of the *modified value difference metric (MVDM)* used to measure the distance between any two cases. It defines the distance $\delta(v_1, v_2)$ between values v_1 and v_2 of the same attribute to be:

$$\delta(v_1, v_2) = \sum_{c_i} \left| \frac{P(v_1, c_i)}{P(v_1)} - \frac{P(v_2, c_i)}{P(v_2)} \right| = \sum_{c_i} \left| P(c_i \mid v_1) - P(c_i \mid v_2) \right|$$

where $P(c_i \mid v_1)$ is the observed probability of class c_i given the value v_1. The distance between two examples A and B is then defined as:

$$d(A, B) = \sum_{m=1}^{n} \delta(A_m, B_m)$$

The interesting characteristic of MVDM is that it incorporates some elements from probabilistic theory into the nearest neighbor framework. Actually, Pebls, keeps exactly the same statistics as Naïve Bayes (the probabilities of joint occurrence of any attribute value with any class) Instead of using these probabilities to directly calculate the class of an unseen case as NB does, Pebls uses them to find the closest stored case and its case is assigned to the new case. The interesting similarities of the two methods are discussed in depth in [45].

2.3.2.3 Hybrid Lazy methods

There is some recent work on combining the lazy learning paradigm with more sophisticated methods for classification. The main issue is to overcome the distance based classification and build a more sophisticated local classifier for each classification query. Lazy Decision Trees [34] is such a hybrid method. Given a case to be classified, [34] proposes to build a decision rule to classify this case. The entropy measure is used for the selection of the best rule. Lazy Bayesian Rules [88] work in a similar way creating a decision rule, where the consequent is a NB classifier instead of a single class prediction. This local NB is used to classify the case.

2.4 A Unifying Perspective of classification methods

In the sequel, we provide a framework that entails most of the methods described above. Its most attractive property is that it provides a common ground for the description of these apparently different classification methods and provides a different perspective in order to evaluate the properties and the comparative advantages or disadvantages of the various algorithms

Let $\boldsymbol{A}$ denote the vector of n attributes $A_1 \ldots A_n$. Each attribute A_i takes any of the values defined in its domain $val(A_i)$. All attributes are assumed discrete; continuous attributes are discretized into intervals using standard algorithms [28]. The resulting intervals are considered as distinct attribute values. Each attribute-value pair is an *item*. A set of items is called an *itemset*. Let D be the training data, which is a set of n-tuples over the domain $val(\boldsymbol{A})$. Each instance or tuple in D is *labeled* with its class c_i. A *labeled itemset* l has associated with it a *class support* sup_i for each class c_i, which contains the frequency of the joint occurrence of l and c_i in the training data. We also define the *support* of l in D, denoted by $l.sup$, as the sum of class supports $l.sup_i$ for all classes c_i. A labeled itemset is called *frequent* if its support matches at least a given minimum support *minsup*. A labeled itemset reduces to an ordinary itemset if only one class exists, i.e., if the data are unlabeled. In the sequel, only labeled itemsets, called itemsets for short, are considered.

Sets of itemsets essentially define the joint probability distributions of the attributes and the class. Let $\boldsymbol{A}$ be a set of attributes; the joint probability $P(\boldsymbol{A},c_i)$ is fully determined by the set of itemsets-instantiations of $\boldsymbol{A}$, i.e., by $P(\boldsymbol{V},c_i)$ for all possible instantiations $\boldsymbol{V}$. The class support $l.sup_i$ of each such itemset l estimates the probability $P(l,c_i)$ and there are $\prod_{A_j \in A} |val(A_j)|$ such itemsets.

The number $|I|$ of possible itemsets that can be generated from a relational database is huge. Assume a relational table of n attributes and assume, for simplicity, that all attributes A_i have the same domain size i.e., $|val(A_1)|= |val(A_2)|=... |val(A_n)|$=k. In this setting, there is a total $|I| = \sum_{i=1}^{n} k^i \cdot \binom{n}{i}$ possible itemsets. These are all $k \cdot n$ itemsets consisting of one item, all $k^2 \cdot \binom{n}{2}$ itemsets consisting of two items and so on up to all the k^n itemsets with n items.

Itemset size	*Maximum number of itemsets*
1	*20*
2	*180*
3	*960*
4	*3360*
5	*8064*
6	*13440*
7	*15360*
8	*11520*
9	*5120*
10	*1024*
TOTAL	*59048*

Table 2: Maximum number of itemsets in a database with 10 binary attributes (***n****=10* and *k=2)*.

As Table 2 illustrates, the number of itemsets that can be produced from a relational table grows extremely quickly and can easily outgrow the number of entries in the table. In this case for example, in the best case the minimum size of the data to provide a minimum support for all itemsets is 1024, i.e., equal to the number of maximal itemsets. This number is unrealistically small since it is based on the assumption that the data follow a uniform distribution where each possible combination appears exactly once in the database.

In real cases most itemsets (especially the longest ones) have zero or very small support. This is particularly true in real data sets, which tend to be skewed, i.e., instances tend to be clustered in certain regions of the attribute space, leaving other regions empty. Non-frequent itemsets provide little or no information for distinguishing the classes. Moreover, in noisy domains they may also provide erroneous or contradicting information, causing overfitting of the data.

Alternatively, the set F of frequent itemsets provides an adequate description of the information contained in D. Although $|F| << |I|$, F may still be too big and in most cases it is $|F| >> |D|$. The discovery and use of F (and all the class supports of its members) for classification would therefore require excessive computational power and huge storage requirements. Hence, a selection or filtering strategy has to reduce F further to a smaller and more manageable set F', with $F' << F$. The set F' ideally should be sufficient to compute the joint probability $P(\boldsymbol{A}, c_i)$ with reasonable accuracy without requiring excessive computational and storage requirements.

In what follows, we describe many of the most popular classification techniques as variations of the following basic scheme. Given a certain instantiation $\boldsymbol{V}$ of the attributes $\boldsymbol{A}$, the classification is done in four steps:

1. Use an appropriate learning bias to discover a subset F' of preferably frequent itemsets (learning phase).

2. Given new evidence V, select a working subset F'' of F'.

3. Use the itemsets of F'' (and their class supports) to estimate the probability $P(V,c_i)$ for each class c_i.

4. Finally, output the class with highest such probability $P(V,c_i)$ as the result.

To illustrate our claim we use the Pima-diabetes database from the UCI ML repository [MM96]. Attributes were pre-discretized and the irrelevant ones were removed. Each attribute-value pair is then treated as a unique item. There are thirteen such pairs; the corresponding items $a_1 \ldots a_{13}$ are shown in Table 3.

Attribute Names	***Values***	***Respective Items***
Pregnant	*-6.5, 6.5+*	a_1, a_2
Glucose	*-99.5, 99.5-117.5, 117.5-154.5, 154.5+*	a_3, a_4, a_5, a_6
BMI	*-27.35, 27.35+*	a_7, a_8
DPF	*-0.5285, 0.5285+*	a_9, a_{10}
Age	*-28.5, 28.5-62.5, 62.5+*	a_{11}, a_{12}, a_{13}
Class: *diabetes*	*pos, neg*	c_1, c_2

Table 3: Attribute values and the respective items in the Pima-Diabetes database.

In the sequel, we illustrate that Decision Trees, Bayesian techniques, and the k-NN algorithm can be regarded as instances of this framework. Based on the insights gained, section 2.5 provides a summary and discussion of our findings.

2.4.1 Decision Trees

Figure 1a shows the pruned decision tree constructed by C4.5 from the database. The pair (p_1,p_2) stored in each leaf contains the percentage of the training cases covered by the leaf, which belong to class c_1 and c_2 respectively. The equivalent set of itemsets F' is shown in Figure 1b. The mapping between the two is straightforward. There is one itemset for each path from the root to a leaf with each node contributing one item to the itemset.

```
glucose = -99.5: pos (24.4%, 1.4%)
glucose =  99.5-117.5:
|  age = -28.5: pos (12.5%, 1.4%)
|  age =  62.5+: pos (0.2%, 0%)
|  age =  28.5-62.5:
|  |  BMI = -27.35: pos (2.3%, 0.6%)
|  |  BMI = 27.35+:
|  |  |  DPF = -0.5285: pos (3.3%, 1.5%)
|  |  |  DPF = 0.5285+: neg (1%, 2.3%)
glucose = 117.5-154.5:
|  BMI = -27.35: pos (5.4%, 0.8%)
|  BMI =  27.35+:
|  |  age = -28.5: pos (7.8%, 2.7%)
|  |  age =  28.5-62.5: neg (6.3%, 9.6%)
|  |  age =  62.5+: pos (0.6%, 0%)
glucose =  154.5+:
|  age =  28.5-62.5: neg (2%, 9.4%)
|  age =  62.5+: pos (0.6%, 0%)
|  age = -28.5:
|  |  BMI = -27.35: pos (0.6%, 0.2%)
|  |  BMI =  27.35+: neg (0.4%, 2.7%)
```

(a) Decision tree

$\{a_3\}$ (24.4%, 1.4%)
$\{a_4,a_{11}\}$ (12.5%, 1.4%)
$\{a_4,a_{13}\}$ (0.2%, 0%)
$\{a_4,a_{12},a_7\}$ (2.3%, 0.6%)
$\{a_4,a_{12},a_8,a_9\}$ (3.3%, 1.5%)
$\{a_4,a_{12},a_8,a_{10}\}$ (1%, 2.3%)
$\{a_5,a_7\}$ (5.4%, 0.8%)
$\{a_5,a_8,a_{11}\}$ (7.8%, 2.7%)
$\{a_5,a_8,a_{12}\}$ (6.3%, 9.6%)
$\{a_5,a_8,a_{13}\}$ (0.6%, 0%)
$\{a_6,a_{12}\}$ (2%, 9.4%)
$\{a_6,a_{13}\}$ (0.6%, 0%)
$\{a_6,a_{11},a_7\}$ (0.6%, 0.2%)
$\{a_6,a_{11},a_8\}$ (0.4%, 2.7%)

(b) F'

Figure 1: Decision tree (a) and its representation as a set of itemsets (b).

In the first step, a decision tree induction algorithm essentially generates and stores a set of labeled itemsets F', the ones shown in Figure 1b. Classification of a new case $V=\{a_1, a_3, a_7, a_9, a_{11}\}$ using itemsets from F', i.e. the computation of $P(V,c_1)$ and $P(V,c_2)$, is done as follows. Select from F' a subset of V (by definition of a decision tree, there is exactly one such itemset in F') and take that itemset's respective class supports. In this case itemset $\{a_3\}$ with class supports 24.4% and 1.4% is selected and the result is c_1.

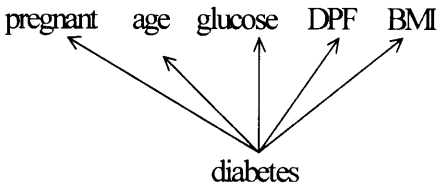
2.4.2 Probabilistic Methods and Naïve Bayes

Probabilistic methods approximate the probability distribution $P(A,c_i)$ by making certain *conditional independence assumptions* [23][70]. Independence assumptions are usually expressed using directed graphs where the nodes are the attributes and each attribute is independent of its non-descendants given its parents. The independence assumptions allow the theoretically correct computation of the full distribution by its lower order components - provided that the independence assumptions are correct. Such lower order probability distributions are stored in *probability tables* that contain the class distribution for each possible instantiation of the attributes. Probability tables are nothing but collections of itemsets.

Consider for example Naïve Bayes (NB). NB assumes that all attributes are independent given the class. NB only stores the probability distributions $P(A_j,C)$ for each attribute A_j i.e., it keeps all labeled 1-itemsets with their class support. There are 13 such 1-itemsets in our example as shown in Figure 2b.

The computation of $P(V,c_1)$ and $P(V,c_2)$ for $V=\{a_1,a_3,a_7,a_9,a_{11}\}$ is as follows. Select all the itemsets of F' which are subsets of V. Based on the class supports of the selected 1-itemsets the probabilities $P(V,c_i)$ are the following:

$$P(V,c_i) = P(c_i)P(a_1 \mid c_i)P(a_3 \mid c_i)P(a_7 \mid c_i)P(a_9 \mid c_i)P(a_{11} \mid c_i)$$



(a) Naïve Bayes Network

$\{a_1\}$ (57.8%, 20.3%)
$\{a_2\}$ (9.6%, 12.3%)
$\{a_3\}$ (24.4%, 1.4%)
$\{a_4\}$ (19.3%, 5.9%)
$\{a_5\}$ (20.1%, 13.1%)
$\{a_6\}$ (3.5%, 12.3%)
$\{a_7\}$ (23.2%, 2.9%)
$\{a_8\}$ (44.1%, 29.7%)
$\{a_9\}$ (49.4%, 16.2%)
$\{a_{10}\}$ (18%, 16.4%)
$\{a_{11}\}$ (39.7%, 7.8%)
$\{a_{12}\}$ (25.2%, 24.8%)
$\{a_{13}\}$(2.5%, 0%)

(b) F'

Figure 2: Naïve Bayes and the corresponding set of itemsets

This is a *product approximation* [52] of the desired probability based on the shortest possible itemsets. The class with highest such probability is chosen. Note that $P(a_1|c_1)$ is nothing but the class support of itemset $\{a_1\}$ (i.e., $P(a_1,c_1)$) divided by the class support of the empty itemset (i.e., $P(c_1)$). Hence, the computation is only based on selecting itemsets and using their class supports.

TAN [32] is a Bayesian network classifier that relaxes the strong independence assumptions of NB by also considering dependencies among some pairs of attributes.

In the learning phase, TAN first measures the degree of dependence between each pair of variables V_j and V_k by calculating their Conditional Mutual Information given the class C:

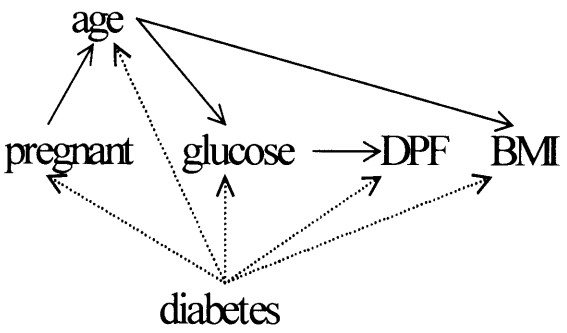
$$CMI(V_j, V_k \mid C) = \sum_{c, v_j, v_k} P(c, v_j, v_k) \cdot \log \frac{P(v_j, v_k \mid c,)}{P(v_j \mid c) \cdot P(v_k \mid c)}$$

The value of the *CMI* quantifies the degree of dependence of two variables given the class. The higher the value of $CMI(V_j, V_k \mid C)$ the more the independence assumption is violated. If V_j and V_k are conditionally independent given the class *CMI* becomes zero. Subsequently, TAN forms a complete weighted graph where each node corresponds to a variable and the weight of the edge (V_j ,V_k) is $CMI(V_j, V_k \mid C)$. A maximum spanning tree algorithm is then applied to the graph resulting in the tree-structured Bayesian Network model that captures the strongest pair-wise dependencies.

All $n \cdot (n-1)/2$ such computations, where n is the feature set size, can be performed with one pass over the data, provided that the $n \cdot (n-1)/2$ probability tables can fit in main memory. This requires bookkeeping during the learning phase of a total of $4 \cdot M \cdot n \cdot (n-1)/2$ counters for the measurement of all pair-wise co-occurrences (or co-absences) under each class label. During classification the Bayesian network is consulted and an estimation of the probability of each class is computed. The document is labeled according to the most probable class. Learning time is linear in the number of examples, but quadratic in the number of features. New examples are classified in time linear to the number of features.

In our example, the result of the learning phase is the Bayesian network of Figure 3a. The equivalent set of itemsets F' is shown in Figure 3b. It consists of the 2-itemsets defined by the Cartesian product of the variable pairs used in the network. In our example, 32 such itemsets are generated (see Figure 3(b)).

The calculation is defined by the topology of the Bayesian network structure. The computation of $P(V,c_1)$ and $P(V,c_2)$ for $V=\{a_1,a_3,a_7,a_9,a_{11}\}$ is as follows. Select all the 2-itemsets of F', which are subsets of V. Based on the class supports of the selected itemsets the probabilities are computed as follows:



(a) TAN network

$\{a_1,a_{11}\}(39.5\%, 7.8\%)$

...

$\{a_2,a_{13},\}(0.8\%, 0\%)$

$\{a_3,a_{11}\}(16.0\%, 0.4\%)$

...

$\{a_6,a_{13}\}(0.6\%, 0\%)$

$\{a_3,a_9\}(18.4\%, 0.4\%)$

...

$\{a_4,a_{10}\}(1.3\%, 7\%)$

$\{a_7,a_{11}\}(15.6\%, 0.8\%)$

...

$\{a_8,a_{13}\}(1.2\%, 0\%)$

(b) F'

Figure 3: TAN and the corresponding set of itemsets

$$P(V,c_i) = P(c_i)P(a_1,a_{11} \mid c_i)P(a_3 \mid a_{11},c_i)P(a_9 \mid a_3,c_i)P(a_7 \mid a_{11},c_i)$$

Again, the result is a product approximation of the desired probability based on the available marginals already stored.

2.4.3 Nearest Neighbor Classifiers

The *k*-nearest neighbor (*k*-NN) classifier is a typical and purely lazy learning algorithm [1] that generates the classifications directly from the stored training instances. Note that instances are synonyms for itemsets of maximal length. Hence, the set F' contains all *n*-itemsets contained in D where n is the number of attributes (see Figure 4). Note that multiple or even contradicting occurrences of the same instance map into single itemsets (e.g., the last itemset of Figure 4).

$\{a_1,a_3,a_7,a_9,a_{11}\}$ (6.25%, 0%)

$\{a_1,a_3,a_7,a_9,a_{12}\}$ (0.4% , 0%)

$\{a_1,a_3,a_7,a_{10},a_{11}\}$ (1.4% , 0%)

$\{a_1,a_3,a_7,a_{10},a_{12}\}$ (0.4% , 0%)

...

$\{a_2,a_6,a_8,a_9,a_{12}\}$ (0.6% , 1.4%)

Figure 4: Set of itemsets F' for *k-NN*

Nearest neighbor classifiers treat instances as points in n-dimensional space and employ a distance function to measure their proximity. Given a classification query, the k closest itemsets to the query point form the working set of itemsets F''. The majority class among the itemsets in F'' is the result. Assume $k=5$ and consider the classification of $V=\{a_1,a_3,a_7,a_9,a_{11}\}$. Figure 4 shows that there is an exact match for V in F' with support 6.25% (or 32 occurrences). Therefore, F'' contains a single itemset and c_1 is returned as the majority class.

2.5 Discussion

We gained a surprising and novel insight, namely that decision trees/rules, Bayesian methods and nearest neighbor classifiers can be represented as collections of patterns/itemsets and all are based on the same generic approach:

1. Generation of labeled itemsets F' during training
2. Selection of a working subset F'' of F' based on the evidence V
3. Computation of the probabilities $P(V,c_i)$ from the class supports of the selected itemsets F''
4. Output of the class with highest estimate of $P(V,c_i)$.

From this point of view, the various classification techniques aim at selecting the class with highest estimated $P(V,c_i)$ without computing the whole probability distribution. The apparent fundamental differences reduce to the use of different biases in the first three steps. Subsequently, we can define the *search bias* (step 1), the *selection bias* (step 2) and the *computation bias* (step 3). The search bias defines the heuristic strategy used for the generation of F', since enumerating the whole set F is computationally intractable. The selection bias defines how the working set F'' is generated upon arrival of a classification query. Finally, the computation bias refers to the method employed for the calculation of $P(V,c_i)$ from the itemsets of F''.

There is a certain dependency among these biases. Fixing any of the three biases restricts the choices for the others. Take C4.5 as an example where classification is based on the class supports of a single itemset. Consequently, the search bias aims at finding itemsets that clearly distinguish among the classes. Most of the work has been done during the first step (search bias), therefore the selection bias degenerates to the selection of the only itemset that is a subset of the classification query and the computation bias simply reports the class with the highest probability in the selected itemset. Probabilistic methods, on the other hand aim at minimizing the independence violations when estimating the probabilities $P(V,c_i)$ from sets of stored itemsets.

It is noteworthy that this framework provides a common ground between lazy and eager learning methods. The connecting point is the use of itemsets as a representational means. Their main differences stem from the bias used in steps (1)-(3).

In the following chapters, we present our research in combining the strengths of both eager and lazy learning paradigms. More specifically, from the eager methods we adapt the probabilistic paradigm. Probabilistic methods enable efficient combination of evidence from *multiple* itemsets and they have been proven useful in many application domains. However, they aim at the discovery of global models, i.e., sets of relationships that hold for all possible instantiations of the attributes. This may not always be feasible. On the other hand, lazy methods classify instances after having created a local model for a specific classification query. This approach yields a highly adaptive behavior, which is lacking in eager algorithms. This deficiency of probabilistic methods is elaborated next.

A Bayesian network (such as the one in Figure 3a) constructed by a Bayesian classifier encodes the global relationships among the attributes. The network encodes the independence assumptions. For example, attributes `age` and `DPF` are independent given `glucose` and `diabetes`. It is well known, however, that certain independence assumptions cannot be captured by such networks [35]. Usually, independence holds only in certain contexts, i.e., for certain instantiations and for certain classes only. In contrast, Bayesian networks encode relationships that hold for all possible instantiations of the variables and for all classes. This representation bias is sometimes too restricting.

Consider for example TAN, described in section 2.4.2. Like many Bayesian Network algorithms, TAN employs an information-theoretic [49] measure to quantify the dependency among variables. For any two variables A_i, A_j, *conditional mutual information* is defined as

$$I(A_i; A_j \mid C) = \sum_{\substack{x \in val(A_i), \\ y \in val(A_j), \\ c_m \in val(C)}} P(x, y, c_m) \log \frac{P(x, y \mid c_m)}{P(x \mid c_m) P(y \mid c_m)}$$

Roughly speaking this function measures how well $P(A_i,A_j|C)$ can be approximated *on average* by $P(A_i|C)$ and $P(A_j|C)$ or how well can the support of the corresponding 2-itemsets be approximated by their 1-itemset subsets. If two variables are independent given the class, no approximation error exists and the result becomes zero. Pairs of variables with high such measure are connected by arcs since using the support of the corresponding itemsets instead of the product of their subsets supports will increase accuracy.

Such averaging is necessary in order to create a global model and is present in most Bayesian methods. It presents, however, the following problems:

1. Ignoring of context-specific dependencies: Itemsets that cannot be accurately approximated by their subsets may wrongly be discarded from the global model because the average score for all itemsets of the corresponding variables is low.

2. Presence of redundant itemsets in the global model because the average score of the corresponding variables is high. (Because some other itemsets contributed to this)

Obviously, storing a separate network for any possible instantiation of the attributes and for each class is infeasible. *Multinets* [H91] provide a partial solution to the problem by constructing |C| different networks one for each class c_i. Still, context-specific independence among attributes remain untreated. Useful insight can be also gained here by research on context-specific independence done by the AI community. See for example [87][86]

Lazy learning and itemset based representation seems a natural approach to address the problem of context specific information. Context-specific independence relationships refer to individual itemsets (instantiations) rather than to sets of variables (as shown in Figure 2 and Figure 3 on the left hand side for instance). However, the large number of alternative instantiations makes it infeasible to prepare all queries for all possible contexts during the training phase. Hence, a lazy approach that computes a local model (i.e., a set of context specific independence assumptions) for a given classification query seems attractive. Such an approach, which we developed inspired by the framework described in this chapter, is discussed in CHAPTER 3.

CHAPTER 3.

INTEGRATING ASSOCIATION AND CLASSIFICATION MINING

Until recently, association (*descriptive*) and classification (*predictive*) mining have been considered as disjoint research and application areas. Descriptive mining aims at the discovery of strong local patterns in the data, so-called *itemsets* [2]. An itemset with its support provides information on the frequency of a certain pattern, therefore it can be thought of as a local descriptor of the data. Such itemsets allow the users to gain insights into the relationships among some of the attributes of the database. The discovery of such interesting patterns from raw data has been recognized as a key topic in data mining [29][36][55]. Predictive mining deals with databases that consist of *labeled* tuples. Each label represents a *class* and the aim is to discover a model of the data that can be used to determine the labels (classes) of previously unseen cases.

Since training a classifier can be thought of as finding a condensed model of the data [80], or in another perspective as discovering a set of interesting patterns [62], the question arises how can itemsets generated from an association miner also be used for classification purposes. In this chapter we investigate the use of association mining techniques as the basis for building efficient classifiers and describe a new classifier called Local Bayes (LB) that we presented in [60] [61] and which is unique in a number of ways:

- LB fits in between the eager and lazy learning paradigms [1]. Eager methods, like Decision Trees [74] and Bayesian Network Classifiers [32][70], infer a global model of the data during training. Classification requests are then answered by querying the learned model. Lazy learners such as nearest-neighbor classifiers, defer processing of the data until classification time, and classify cases by directly using the unprocessed data. In contrast to both, the learning phase of LB creates a condensed representation of the database in the form of itemsets with their class supports. Classification queries are then satisfied by constructing on the fly a query-specific local model using the evidence provided by the itemsets.

- This work sheds light on the relationship between descriptive (association) and classification mining. An association mining algorithm, which in this case is Apriori [AS94], is modified to discover frequent and interesting patterns from data and we show that this condensed representation can be used for effective classification. Moreover, the approach is flexible enough to include a variety of itemset mining algorithms like the recent FP-growth [37] algorithm. This flexibility provides the benefit that any advances in association mining techniques can be integrated into LB to enhance its efficiency. The association mining approach is particularly useful in domains where greedy search methods (such as decision tree induction algorithms) are unable to take into account all aspects of the underlying model as well as in domains where it is important to capture context-specific information about the relationships that exist in the data as explained in the sequel.

- LB is the first classifier, to the best of our knowledge, that builds local probabilistic models in order to classify unknown examples. All existing algorithms for the inference of Bayesian Networks (BN) or Bayesian Network classifiers from data rely on the detection of global dependencies and independencies among the variables to construct global models [32][70]. Our approach is completely different. During training we collect a set of interesting itemsets, i.e., sets of attribute-value pairs. A local network is only constructed during classification and the independence assumptions hold only in the context of the given classification query. This allows the creation of classifiers that are sensitive enough to model independencies that hold in specific contexts without building overly complex global Bayesian networks.

- Most real data sets contain only a small fraction of the possible attribute-value pairs. Moreover, a small subset of the possible variable assignments (itemsets) covers the majority of the probability space. Instead of trying to accurately estimate the probability of any possible set of attribute-value pairs (also called items) we rather focus on estimating the probabilities of those sets of items that occur frequently in the data. This leads to accurate and stable results while the size of the classifier is kept small.

- Experimental results on a large number of benchmark data sets show that Local Bayes consistently outperforms the widely used Naïve Bayes classifier. In many cases, Local Bayes is also superior to other state of the art classification methods such as C4.5, CBA (a recently proposed rule classifier built from association rules), and TAN (a Bayesian network extension of Naïve Bayes).

3.1 Overview of Local Bayesian Classification.

LB is derived from association mining, and this is apparent from the terminology we use to describe it. It considers each attribute-value pair as a distinct *item* and assumes that the training set is a set of transactions. During the learning phase, LB employs an association mining algorithm to discover *interesting* and *frequent labeled* itemsets of arbitrary size. In the context of classification, we define a labeled itemset l as a set of items together with its class-supports $l.sup_i$ for each possible class c_i. The class-support is a variation of the usual support notion and measures the frequency of the joint occurrence of a pattern with each class label. In other words a labeled itemset provides the observed probability distribution of the class variable given an assignment of values for the corresponding attributes: $l.sup_i = P(l,c_i)$. As Figure 5 illustrates, LB does not build a classification model during the training phase. The outcome of the training is the set of discovered itemsets.

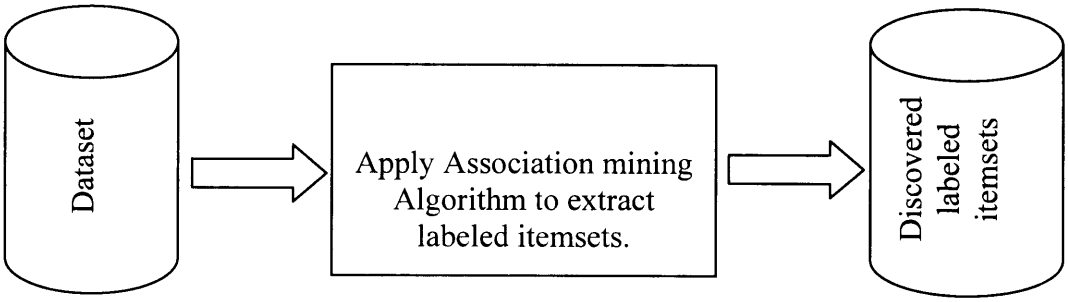


Figure 5: Illustration of the training process for Local Bayes Classifier

Upon arrival of a new case $A=\{a_1,a_2,...,a_n\}$ to be classified, a local classification model is built on the fly focusing on the context of this particular case only. 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 the new case A consists of a formula computing the conditional probability $P(c_i|A)=P(A,c_i)/P(A)$, i.e., the probability that the case belongs to class c_i given the evidence A. Since the denominator is constant with respect to c_i it can be ignored and the object is said to be in class c_i with the highest value $P(A,c_i)$.

As discussed in [52], the probability $P(A, c_i)$ can be estimated using different product approximations. Each product approximation implies different independence assumptions about the attributes. For example, $P(a_1,a_2,a_3,c_i)P(a_4,a_5|a_1,c_i)$ and $P(a_1,a_2,a_3,c_i)P(a_5|a_2,c_i)P(a_4|a_1,a_5,c_i)$ are both product approximations of $P(a_1,a_2,a_3,a_4,a_5,c_i)$. We show that selecting a certain product approximation is equivalent to selecting certain itemsets generated while training. The resulting classification process is illustrated in Figure 6,

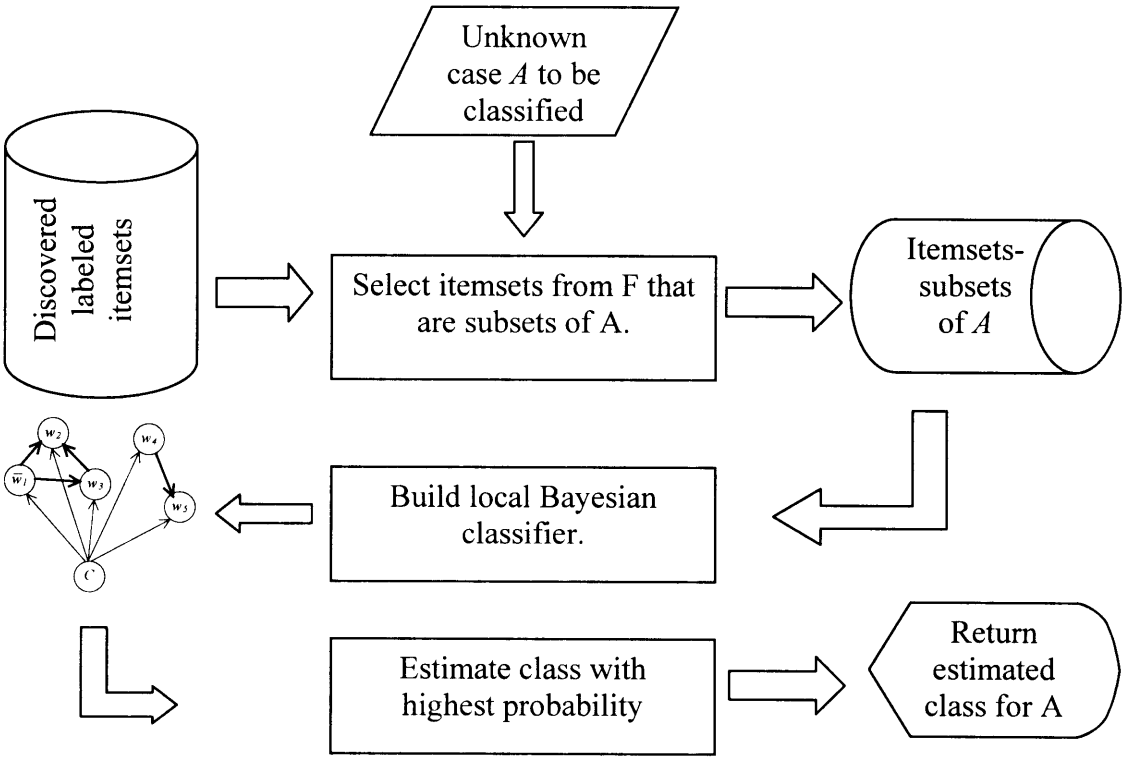


Figure 6: Illustration of the classification process for Local Bayes Classifier

The itemset selection strategy uses four underlying principles:

(i) Itemsets should be reliable, hence frequent; also called large in itemset terminology;

(ii) As many itemsets as possible should be used, i.e., the product approximations should contain as many factors as possible;

(iii) The individual itemsets or factors should be as large in size as possible; and

(iv) The itemsets chosen should be interesting in the sense that they indeed provide more information than their subsets.

In [60] and [61] we named the proposed classifier Large Bayes (LB). Keeping the same initials, we now feel that it is more appropriate to give it the name *Local Bayes*. This name seems justified as LB builds local models and in the extreme case, when the selected itemsets are all of size one, Local Bayes reduces to Naïve Bayes.

Already Naive Bayes (NB), described in section 2.3.1.3, is a surprisingly successful classification method that has outperformed much more complicated methods in many application domains [26][32]. NB assigns new cases, represented as a vector of attribute-values $\boldsymbol{A}=\{a_1,a_2,\ldots,a_n\}$, to the class c_i with the highest conditional probability $P(c_i|\boldsymbol{A})$. From the definition of conditional probability follows $P(c_i|\boldsymbol{A})=P(\boldsymbol{A},c_i)/P(\boldsymbol{A})$. Since the denominator is constant with respect to c_i it can be ignored and the object is said to be in class c_i with the highest value $P(\boldsymbol{A},c_i)=P(\boldsymbol{A}|c_i)P(c_i)$. To compute $P(\boldsymbol{A},c_i)$[2], NB assumes that all attributes are conditionally independent given the class c_i, hence:

Equation 1 $$P(a_1\ldots a_n c_i) = P(c_i)P(a_1 \mid c_i)\cdots P(a_n \mid c_i) = \frac{\prod_{j=1}^{n} P(a_j, c_i)}{P(c_i)^{n-1}}$$

Essentially, this means that the classification solely depends on the values of $P(a_j,c_i)$ and $P(c_i)$. LB can also be seen as a powerful extension of NB that uses itemsets of arbitrary size when estimating $P(a_1\ldots a_n c_i)$. This implicitly relaxes the strong independence assumptions implied by NB, but retains its strengths: superior performance in many cases over state of the art classification methods, and the important ability to handle missing attribute values.

---

[2] We use the notation $P(a_1\ldots a_n c_i)$ and $P(\boldsymbol{A},c_i)$ interchangeably.

The rest of this chapter is structured as follows. Section 3.2 reviews related work and motivates our approach. Section 3.3 provides an intuitive illustration of the algorithm and shows the relationship between the classification produced by LB on the one hand and Bayesian Networks on the other. Section 3.4 discusses the algorithm in detail, pinpoints various fine-tuning opportunities that can lead to even further improved classification accuracy and presents an extensive experimental evaluation of LB. For the comparison we use popular benchmark datasets and compare LB with C4.5, the standard decision-tree classifier, Naïve Bayes, the simple yet proven classifier, TAN which is a restricted Bayesian network extension of NB, and CBA, a recently proposed rule based classification method based on association rule mining. In section 3.5 we present an enhancement of LB that uses chi-square tests as a measure of the quality of itemsets, we describe some trade offs that can be used to establish a balance between simplicity and speed on one hand and training time on the other. The evaluation of these enhancements shows considerable improvement in terms of classification accuracy for LB. Section 3.6 draws our conclusions.

3.2 Related Work

The surprising success of NB has triggered the development of several extensions, most of which aim at relaxing its strong independence assumptions. Most of them fall within the more general framework of learning Bayesian networks from data [15][38]. *Tree Augmented Naive Bayesian classifier (TAN)* [32] extends NB taking into account additional dependencies among non-class attributes. TAN employs a modified version of a method proposed by [17] to learn a restricted tree-structured Bayesian network considering only the most important correlations between pairs of attributes. Several other methods follow the same direction. *KDB* [76] constructs a Bayesian classifier where each attribute may depend on at most k other attributes, where k is a given parameter. The *Selective Bayesian Classifier* [54] preprocesses data using a form of feature subset selection to delete strongly inter-related (and therefore redundant) attributes. *Semi-Naïve Bayesian Classifier* [48] iteratively joins pairs of attributes in what may be called *feature construction* [30] to relax the strongest independence assumptions. Instead of relaxing the independence assumptions in order to increase the accuracy of the probability estimations, A*djusted Probability NB* [81] infers a weight for each class, which is applied to derive an adjusted probability estimation used for the classification. Finally, *NBTree* [46] is a hybrid approach combining Naïve Bayes and decision-trees. A decision tree partitions the instance space into regions and a separate NB classifies cases within each region. NBTree is shown to frequently outperform both NB and decision-trees in terms of classification accuracy.

All these methods aim at relaxing the strong independence assumptions implied by NB. Variables x and y are said to be conditionally independent given variable z, denoted $I(x|z|y)$, if for all values of x,y and z: $P(x|y,z)=P(x|z)$, whenever $P(y,z)>0$. Consider now the extreme case where the condition $P(x|y,z)=P(x|z)$ is satisfied by all possible instantiations of x,y and z except by a few instantiations only, which badly violate it. Clearly, x and y are conditionally dependent given z. Algorithms like the ones above would consider to store all elements of the joint probability distribution $P(x,y,z)$ and use it during classification to obtain more accurate estimations. Such an approach is not only redundant but also error prone because there might not exist enough data to provide reliable probability estimations for each possible variable assignment. To alleviate this representational weakness, [11] introduces the notion of *context-specific independence,* which describes independence relations among variables that hold in certain contexts only. Similarly, *Bayesian Multinets* [39] model relationships among variables within each class separately. Local Bayes goes even further. Only relationships among instantiations of all variables are considered. These relationships are expressed using itemsets. As illustrated before, this representation accounts for additional flexibility, precision and increased representational ability.

Liu et al [53] follow an approach similar to ours introducing a method for the integration of association and classification mining. They use association rules to generate a complete classifier. All association rules with only the target class in the head are filtered out. This forms a set of classification rules. Using heuristics to prune this large rule set, they simplify it to finally obtain a smaller classifier. It is shown that this classifier, called *CBA,* outperforms C4.5 [74] in many cases. Earlier work on using association rules for classification includes [7] that discusses the problem of enhancing an association rule miner with additional pruning strategies to extract high-confidence (>90%) classification rules from data sets. [3] discusses the use of association rule mining for the discovery of models of the data that may not cover all classes or all examples.

3.3 Classification Using Local Patterns

LB approximates long marginals (support of itemsets with many items) using itemsets of arbitrary length as long as they are frequent and, therefore, presumably reliable. All attributes are assumed to be discrete. Continuous attributes are discretized into intervals using standard discretization algorithms [28]. The resulting intervals are considered as distinct attribute values. Each possible attribute-value pair is called an *item.* Thus, in a domain described by n attributes, each *training example* is represented by an *itemset* [2] $\{a_1,...,a_n\}$ containing the items which are true, and is labeled with a class c_i. Note that the labeled itemsets have size n (each of the n attributes has one value and they do not contain the class attribute) and this size is reduced by one for each missing attribute value.

We will briefly introduce some notations here. Let D denote the set of training examples. An itemset l has support s for class c_i in D (denoted by $l.sup_i = s$), if s% of the cases in D contain both c_i and l. We also define the *support* of l in D, denoted by $l.sup$, as the sum of supports $l.sup_i$ for all classes c_i. Note that $l.sup_i$ is the observed probability $P(l,c_i)$. Similarly, $l.sup$ is the observed probability of $P(l)$. An itemset is called *frequent* if its support matches at least a given minimum support *minsup*.

The learning phase of LB employs an extension of Apriori [2] to extract from D a set F of frequent and *interesting* itemsets. An itemset $\boldsymbol{A}$ is interesting in our context, if $P(\boldsymbol{A},c_i)$ cannot be accurately approximated by its direct subsets, i.e., subsets of $\boldsymbol{A}$ with one item missing. A quantitative measure of interestingness is presented in section 3.4.1. No complete model of the domain is built during the training phase; the result of the learning phase is a condensed representation of the database tailored for the purpose of classification.

Classification of new cases $\boldsymbol{A}=\{a_1,\ldots,a_n\}$ is done by combining the evidence provided by the subsets of $\boldsymbol{A}$ that are present in F (see procedure *classify*, section 3.4.2). Long itemsets (i.e., with many items) are obviously preferred for classification as they provide more information about the higher order interactions between attributes. This is realized by first selecting the *border of F with respect to $\boldsymbol{A}$*. The border consists of the longest possible itemsets of F that are subsets of $\boldsymbol{A}$. Only itemsets of the border B are used in the solution. The following example illustrates this and will be used in the sequel as the running example.

Suppose a request to classify $A=\{a_1,\ldots,a_5\}$ arrives. In the learning phase, the set F of all interesting and frequent itemsets has been computed. Figure 7 shows all subsets of A in F. The non frequent or uninteresting ones are marked gray and cannot be used as their support can either not be counted reliably (non frequent) or can be approximated using smaller itemsets (uninteresting). We first select the border of F with respect to $\{a_1,\ldots,a_5\}$. These itemsets, marked in bold in Figure 7, can be used to compute the desired estimates. Note that NB uses all supports in the first level of Figure 7, i.e., all single attributes, and TAN employs some itemsets of the second level, i.e., pairs of attributes.

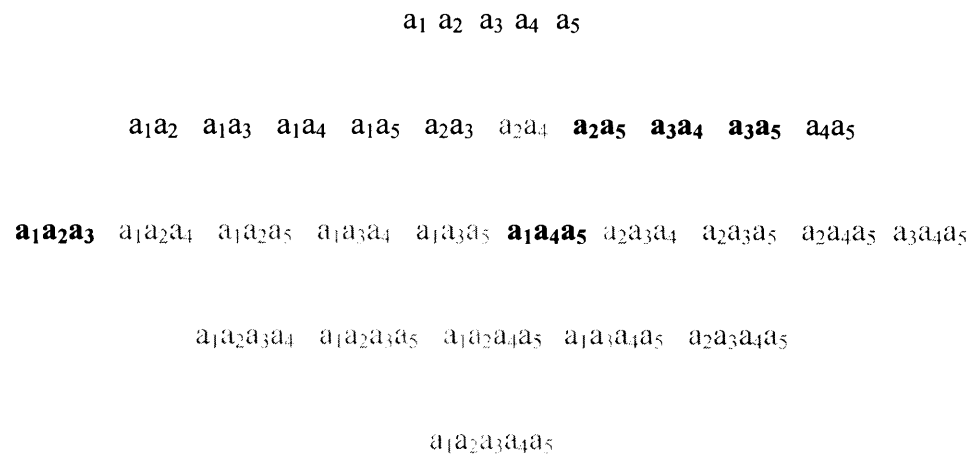


Figure 7: The border of F with respect to $\{a_1,\ldots,a_5\}$

LB uses the itemsets of B to derive a *product approximation* of $P(\boldsymbol{A},c_i)$ for all classes c_i. [L59] describes the concept of product approximation in the context of approximation of higher order probability distributions by their lower order components. We only consider the approximation of individual elements of a probability distribution. Following [52], the product approximation of the probability of an n-itemset $\boldsymbol{A}$ for a class c_i contains a sequence of at most n subsets of $\boldsymbol{A}$ such that each itemset contains at least one item not contained (or *covered*) in the previous itemsets. To derive the approximated value of $P(\boldsymbol{A},c_i)$ the itemsets are combined using the chain rule of probability while assuming that all necessary independence assumptions are true. The following are some valid product approximations of $\{a_1,\ldots,a_5\}$ using the border itemsets shown in Figure 7:,

i. $\{a_1,a_2,a_3\},\{a_1,a_4,a_5\} \Rightarrow P(a_1,a_2,a_3,c_i)\ P(a_4,a_5|a_1,c_i)$

ii. $\{a_1,a_2,a_3\},\{a_2,a_5\},\{a_1,a_4,a_5\} \Rightarrow P(a_1,a_2,a_3,c_i)P(a_5|a_2,c_i)\ P(a_4|a_1,a_5,c_i)$

iii. $\{a_1,a_2,a_3\},\{a_2,a_5\},\{a_3,a_4\} \Rightarrow P(a_1,a_2,a_3,c_i)\ P(a_5|a_2,c_i)\ P(a_4|a_3,c_i)$

iv. $\{a_2,a_5\},\{a_3,a_5\},\{a_1,a_2,a_3\},\{a_1,a_4,a_5\} \Rightarrow P(a_2,a_5,c_i)P(a_3|a_5,c_i)P(a_1|a_2,a_3,c_i)P(a_4|a_1,a_5,c_i)$

Note that $\{a_1,a_2,a_3\}$, $\{a_1,a_4,a_5\}$, $\{a_2,a_5\}$ (which contains the same itemsets as ii. but in a different order) cannot be written as a product approximation since all items of $\{a_2,a_5\}$ are already covered by the first two itemsets.

Clearly, more than one product approximation of a given itemset exists and not all border itemsets can be used in the solution. The product approximation of the query itemset $\boldsymbol{A}$ is created incrementally adding one itemset at a time until no more itemsets can be added. All items already included in the product approximation are said to be *covered*. An itemset l inserted in the solution should satisfy the following condition:

Condition 1) $|l\text{- }covered| \geq 1$

Condition 1 guarantees that the solution is indeed a product approximation. Selection among alternatives is done as follows. Itemset l_k is selected instead of l_j if the following conditions are satisfied in the given order of importance:

Condition 2) $|l_k\text{-}covered| \leq |l_j\text{-}covered|$

Condition 3) $|l_k| \geq |l_j|$

Condition 4) l_k is more interesting than l_j.

Condition 2 assures that each itemset in the sequence contains the smallest number of not covered items. This is equivalent to maximizing the number of itemsets used. This bias essentially ensures that as many higher order interactions as possible are taken into account (minimizes the conditional independence assumptions). Essentially, conditions 1 and 2 ensure that each additional itemset inserted will add exactly one new item in the product approximation. This is ensured by the fact that all 1-itemsets (itemsets containing one item) are included in the model. If no larger itemset introduces exactly one item (which is the minimum as required by condition 2) then one of the 1-itemsets will do so. Condition 3 ensures that long itemsets are considered if there are alternatives with the same minimal number of uncovered items (implicitly again minimizing the independence assumptions). Finally, condition 4 gives priority to interesting itemsets to resolve ties between itemsets that satisfy all the three first conditions.

#	Covered items	Itemset selected	Product approximation	Available itemsets
0	∅	∅	N/A	$\{a_1a_2a_3\},\{a_1a_4a_5\},\{a_2a_5\},$ $\{a_3a_4\},a_3a_5\}$
1	$\{a_2a_5\}$	$\{a_2a_5\}$	$P(a_2a_5c_i)$	$\{a_1a_2a_3\},\{a_1a_4a_5\},\{a_3a_4\},\{a_3a_5\}$
2	$\{a_2a_3a_5\}$	$\{a_3a_5\}$	$P(a_2a_5c_i)P(a_3\|a_5c_i)$	$\{a_1a_2a_3\},\{a_1a_4a_5\},\{a_3a_4\}$
3	$\{a_1a_2a_3a_5\}$	$\{a_1a_2a_3\}$	$P(a_2a_5c_i)P(a_3\|a_5c_i)P(a_1\|a_2a_3c_i)$	$\{a_1a_4a_5\},\{a_3a_4\}$
4	$\{a_1a_2a_3a_4a_5\}$	$\{a_1a_4a_5\}$	$P(a_2a_5c_i)P(a_3\|a_5c_i)$ $P(a_1\|a_2a_3c_i)P(a_4\|a_1a_5c_i)$	$\{a_3a_4\}$

Figure 8: Incremental construction of a product approximation for $P(a_1...a_5c_i)$.

Figure 8 displays the incremental construction of the product approximation of $P(a_1...a_5c_i)$, in our running example. Initially, the solution and the set of covered items are empty. While all itemsets satisfy condition 1, condition 2 is satisfied by $\{a_2,a_5\},\{a_3,a_4\}$ and $\{a_3,a_5\}$ and all three have the same size. Therefore, the most interesting one is selected. Assume this is $\{a_2,a_5\}$. It is inserted in the solution and items a_2 and a_5 are covered now. Among the remaining itemsets, $\{a_3,a_5\}$ introduces the fewest non-covered items, namely a_3 only. It is inserted in the solution and a_3 is marked as covered. In the third step, itemsets $\{a_3,a_4\}$ and $\{a_1,a_2,a_3\}$ both introduce one non-covered item (a_4 and a_1 respectively). Since $\{a_1,a_2,a_3\}$ is the longest it is preferred, as is $\{a_1,a_4,a_5\}$ preferred over $\{a_3,a_4\}$ in the fourth step. Itemset $\{a_3,a_4\}$ remains unused as all its items are already covered. As a result, $P(a_1...a_5c_i)$ is computed using the product approximation

$$P(a_2a_5c_i)P(a_3|a_5c_i)P(a_1|a_2a_3c_i)P(a_4|a_1a_5c_i).$$

This solution is a local model describing only the relationships among the specific attribute values of the query. All implied independence assumptions are only true in the context defined by $\boldsymbol{A}$. Consider for example the items a_2, a_3, a_5 and let v_2, v_3 and v_5 be their corresponding variables. In other words, a_2, a_3 and a_5 are instantiations of the variables v_2, v_3 and v_5. Our local model assumes that $P(a_3|a_2,a_5,c_i)=P(a_3|a_5,c_i)$. This is much weaker than assuming that $P(v_3|v_2,v_5,c_i)=P(v_3|v_5,c_i)$ for all possible instantiations of v_2, v_3 and v_5; a global model would make this stronger assumption.

An interesting property of LB is that the resulting models can always be represented as instantiations of a Bayesian Network classifier. In other words, the incremental construction of a product approximation can be seen as an incremental construction of a Bayesian Network classifier with specific values assigned to its variables. The Bayesian network equivalent to the approximation of our previous example is shown in Figure 9.

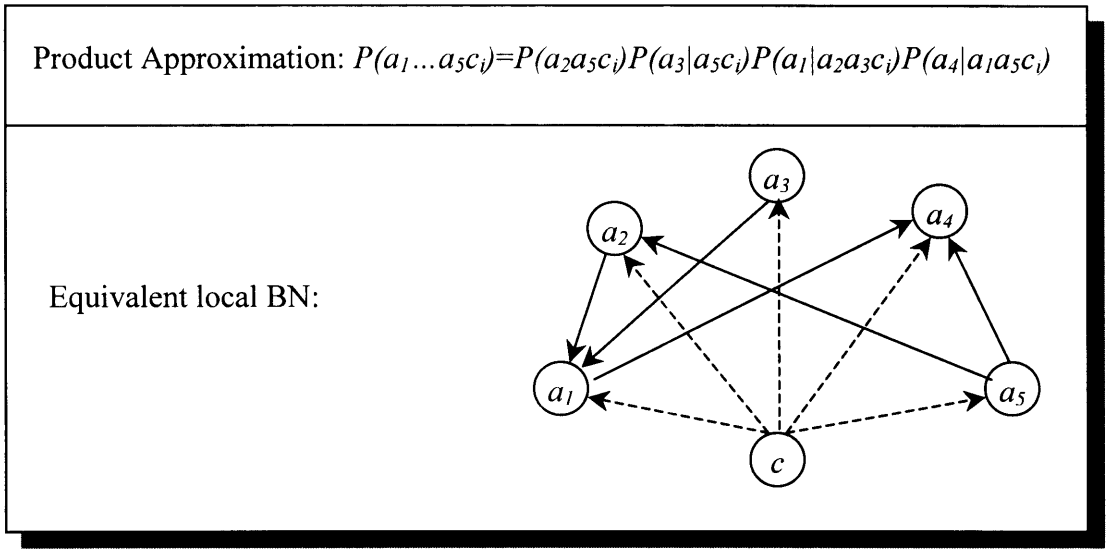


Figure 9: Product approximation constructed by LB and equivalent local Bayesian Network Classifier. Dotted lines represent the dependencies used by Naïve Bayes

The algorithm for the construction of the BN of Figure 9 is listed in Figure 10. A number of observations can be made for Figure 10. Each item corresponds to a node in the BN and the class is an additional node. For the first itemset inserted there are no independencies among the items or between items and the class. Therefore, the corresponding nodes are connected in a clique including the class. After the first insertion, every new itemset inserted in the product approximation will contain exactly one item that is not already present (line 10). This is guaranteed by conditions 1 and 2 as explained above. This new item will be conditionally independent of all other items already in the BN given the ones in the itemset where it belongs. This is indicated in the BN by drawing arcs from the items in the new itemset that already exist in the BN to the new item. We can show this using Figure 8 and Figure 9. When the itemset $\{a_1a_2a_3\}$ is inserted in the product approximation, items a_2 and a_3 exist already as nodes in the BN and a_1 is the newly introduced item. Therefore three arcs will be directed to a_1: One arc will originate from the class (all items are dependent on the class, and one arc from each a_2 and a_3.

1. Start with an empty network N.
2. Add a single node to N representing the class variable c
3. Let l_1 be the first itemset included in the product approximation.
4. For each item $l_{1,k}$ of l_1 {
5. Add a new node in N representing $l_{1,k}$
6. Add an arc from C to $l_{1,k}$
7. Add an arc from all previously inserted items of l_1 to $l_{1,k}$
8. }
9. For each additional itemset l_j added to the product approximation{
10. Add a new node in N representing the new item introduced by l_j
11. Add an arc from C to this new item.
12. Add an arc from all other items of l_j to this new item.
13. }

Figure 10: Procedure for the incremental creation of BN equivalent of a product approximation.

From the above we can prove the following lemma:

Lemma: Each model generated by LB during classification can be represented as a Directed Acyclic Graph generated with the procedure of Figure 10.

Proof: LB adds itemsets sequentially in such a way that in the corresponding graph one new node is inserted with each itemset insertion. This new node will only receive arcs from existing nodes in the BN. Therefore no cycle can be created because this would require at least one arc to be directed from a new node towards an older one, which never happens.

Now we are ready to show that the models created by LB are indeed *Local Bayesian Classifiers* in the sense that they are instantiations of Bayesian Networks where each variable apart from the class has been assigned a value.

Lemma: For each model M generated by LB there is a Bayesian Network B such that M can be derived from B by assigning a value to each of the variables in B.

Proof: The proof comes directly from the definition of Bayesian Networks: A BN B is a DAG with n nodes where each node represents a random variable and edges represent direct dependencies between variables. The graph structure encodes the following set of independence assumptions: Each node X_i is independent of its non-descendants given its parents. A Bayesian network represents a unique probability distribution over the n variables given by:

Equation 2: $$P_B(X_1, X_2, ..., X_n) = \prod_1^n P_B(X_i | parents(X_i))$$

The model M is a *DAG* as we showed in the previous lemma. Each node represents the random variable for which the corresponding item is an instantiation. Each itemset l inserted in M introduces a term of the form $P_B(X_i | parents(X_i))$ where X_i is the variable that corresponds to the item of l that does not already exist in M and $parents(X_i)$ is the set of variables that correspond to the items of l that already exist in M. Therefore there is exactly one Bayesian Network B with the same graph structure as M where the nodes are the variables that correspond to the items of M. The model M is the instantiation of this Bayesian Network B. End of proof.

In the sequel we describe the algorithm in detail and explain the itemset selection strategy.

3.4 Algorithm Local Bayes (LB)

Section 3.4.1 presents the training phase, which consists of discovering the set F of all interesting (according to a strict definition) and frequent (satisfying a minimum support threshold) itemsets with their class supports. Section 3.4.2 shows how these itemsets and the class supports are used to classify new cases and section 3.4.3 studies statistical fine-tunings. The experimental evaluation of this algorithm is presented and discussed in section 3.4.4.

3.4.1 Learning or Discovering Interesting Itemsets

Algorithm *genItemsets* (Figure 11) generates the interesting and frequent itemsets using a bottom-up approach based on Apriori [2]. The input of the algorithm is the database D and the output is the set of interesting and frequent itemsets F with their class counts. To facilitate the class counting, each itemset has an associated counter $count_i$ for each class c_i. Dividing a class counter by the number of tuples $|D|$ provides the *class support*, that is, $l.count_i/|D|=P(l,c_i)$.

First, all 1-itemsets $l=\{a_j\}$ are included in F_1 (a_j is a non-class attribute). Then the class count $l.count_i$ for each class c_i is determined by scanning the database D once (line 2). This assures that LB keeps at least as much information as Naïve-Bayes does. In general, the set F_k contains the set of frequent and interesting itemsets of size k.

genItemsets(D)

input: the database D of training examples

output: the set F of itemsets l and their class counts $l.count_i$

1. $F_1 = \{\{a_j\} | \ a_j$ is non class attribute$\}$
2. determine $l.count_i$ for all $l \in F_1$ and all classes i
3. for ($k=2$; $F_{k-1} \neq \emptyset$; k++) {
4. $C_k = genCandidates(F_{k-1})$
5. for all tuples $t \in D$ {
6. C_t = subsets(C_k, t);
7. i = class of t;
8. for all candidates $l \in C_t$ {
9. $l.count_i$++;
10. }
11. }
12. $F_k = selectF(C_k)$
13. }
14. return $F = \cup_k F_k$ and $l.count_i$ for all $l \in F$ and all I

Figure 11: Algorithm *genItemsets*.

Procedure *genCandidates* generates the candidate itemsets C_k, a superset of F_{k-1}. In lines 5-9, the database is scanned to calculate the class supports for all itemsets in C_k. For each tuple $t \in D$, all subsets of t that belong to C_k are selected (line 6) and their counters that correspond to the class of t are increased (lines 7-9). This determines the class counts $l.count_i$ of all candidate itemsets l. After the counting is completed, the interesting (see below) and frequent itemsets are selected by procedure *selectF* (line 12).

Procedure *genCandidates* is an extension of the candidate generation procedure suggested in Apriori [2]. It generates candidate itemsets l of size k such that each subset l' of l with size k-1 is frequent and interesting. Note that when calling *genCandidates*(F_{k-1}) for $k>2$, all itemsets in F_{k-1} satisfy the two requirements as this is ensured in line 12 by procedure *selectF*. Therefore, there is no need to explicitly check these two properties. When $k=2$, however, we explicitly check if the itemsets are frequent (since F_1 contains both frequent and infrequent itemsets), but we assume that all 1-itemsets are interesting.

Procedure *selectF* selects from C_k those itemsets that are frequent and interesting. An itemset is *frequent*, if its support is above the user defined threshold *minsup*,

$$\frac{\sum_i l.count_i}{|D|} \geq minsup\ .$$

We define the interestingness of an itemset l in terms of the error when estimating $P(l,c_i)$ using subsets of l. Let l be an itemset of size $|l|$ and l_j , l_k be two ($|l|$-1)-itemsets obtained from l by omitting the j^{th} and k^{th} item respectively. We can use l_j , l_k to produce an estimate $P_{j,k}(l,c_i)$ of $P(l,c_i)$:

Equation 3

$$P_{j,k}(l,c_i) = \frac{P(l_j,c_i) \times P(l_k,c_i)}{P(l_j \cap l_k,c_i)}$$

The *interestingness* $I(l \mid l_j,l_k)$ *of* l *with respect to* l_j and l_k is a measure of the accuracy of this estimate:

Equation 4

$$I(l \mid l_j,l_k) = \sum_{c_i} \left| P(l,c_i) \log \frac{P(l,c_i)}{P_{j,k}(l,c_i)} \right|$$

$I(l \mid l_j, l_k)$ becomes zero if $P_{j,k}(l, c_i) = P(l, c_i)$; its value increases with the difference between the estimated and the actual probability. The absolute value in Equation 4 is necessary as $P_{j,k}(l, c_i)$ can be greater or less than $P(l, c_i)$, thus yielding positive or negative values for the logarithm. Taking the absolute value prevents a positive and a negative diversion from canceling each other.

The *interestingness* of l is defined as the average over all its subsets l_j and l_k with size $|l|$-1. Since there are $\binom{|l|}{2} = \frac{|l| \cdot (|l|-1)}{2}$ pairs of different l_j and l_k, we have:

Equation 5

$$I(l) = \frac{2}{|l| \cdot (|l|-1)} \cdot \sum_{\substack{j,k \in \{1,..,|l|\} \\ j<k}} I(l \mid l_j, l_k)$$

The proposed interestingness measure $I(l)$ quantifies how accurately $P(l,c_i)$ can be approximated using any two subsets of size $|l|$-1. A high value of $I(l)$ means that l is interesting, since $P(l,c_i)$ cannot be accurately estimated using smaller itemsets. On the other hand, if $I(l)$ is below a certain threshold τ_I, then we can safely ignore l since it does not provide much additional information.

Our interestingness measure is a variation of *cross-entropy* $I_{P-P'}$ [52], a measure used in probability theory to estimate the distance between two probability distributions P and P':

Equation 6

$$I_{P-P'} = \sum_{\mathbf{x}} P(\mathbf{x}) \log \frac{P(\mathbf{x})}{P'(\mathbf{x})}$$

The difference in our case is that we estimate the difference between specific elements of P and P' rather that between the whole distributions.

3.4.2 Classifying, or equivalent, computing a Product Approximation

Once the set F of interesting and frequent itemsets is available, new unlabeled cases A are classified by procedure *classify* in Figure 12. The class supports $P(l,c_i)=l.count_i/|D|$ computed during the learning phase are used.

As described earlier, the goal is to incrementally build the product approximation of $\boldsymbol{A}$ by adding one itemset at a time until no more can be added. Line 4 selects the border B of F with respect to $\boldsymbol{A}$; B is the set of all subsets of $\boldsymbol{A}$ of maximal cardinality that belong to F. Procedure *pickNext* (Figure 13) then repeatedly inserts itemsets from the border in the solution marking their items as covered. The algorithm stops once all items in $\boldsymbol{A}$ have been covered.

classify(F,A)
Input: The set F of discovered itemsets a new instance $\boldsymbol{A}$
Output: The classification c_i of $\boldsymbol{A}$

1. $cov = \varnothing$ \\ the subset of $\boldsymbol{A}$ already covered
2. $nom = \varnothing$ \\ set of itemsets in nominator
3. $den = \varnothing$ \\ set of itemsets in denominator
4. $B = \{l \in F \mid l \subseteq \boldsymbol{A}$ and $\neg\exists l' \in F: l' \subseteq \boldsymbol{A}$ and $l \subset l'\}$.
5. for $(k=1;\ cov \subset A;\ k++)\{$
6. $\quad l_k = pickNext(cov, B)$
7. $\quad nom = nom \cup \{l_k\}$
8. $\quad den = den \cup \{l_k \cap \text{cov}\}$
9. $\quad cov = cov \cup l_k$
10. }
11. output that class c_i with maximal $P(A,c_i)$ computed as:

$$P(\mathbf{A}, c_i) = P(c_i) \cdot \frac{\prod_{l \in nom} P(l, c_i)}{\prod_{h \in den} P(h, c_i)}$$

Figure 12: Algorithm *classify*.

Each itemset l_k selected in line 6 contributes one factor to the product approximation, namely, the conditional probability of the non-covered items of l_k given the covered ones and the class:

Equation 7 $$P(l_k - \text{cov} \mid l_k \cap \text{cov}, c_i) = \frac{P(l_k, c_i)}{P(l_k \cap \text{cov}, c_i)}$$

In line 7 and 8 the nominator and denominator itemsets of Equation 7, l_k and $l_k \cap \text{cov}$ respectively, are stored in two separate sets *nom* and *den* which are used in line 11 to derive the value of the product approximation of $P(\mathbf{A}, c_i)$ for each class c_i. The most likely class is then returned.

pickNext(cov, B)
  $T = \{ l \in B: |l\text{- covered}| \geq 1 \}$;
  Return an itemset $l_k \in T$ such that for all other itemsets $l_j \in T$:
a) $|l_k\text{-covered}| < |l_j\text{-covered}|$, or

b) $|l_k\text{-covered}| = |l_j\text{-covered}|$ and $|l_k| > |l_j|$, or

c) $|l_k\text{-covered}| = |l_j\text{-covered}|$ and $|l_k| = |l_j|$ and $I(l_k) > I(l_j)$

Figure 13: Procedure *pickNext*.

Procedure *pickNext* selects from B the next itemset of the product approximation. This is uniquely determined from the selection conditions 1–4 of section 3.3. In the rare case where more than one itemsets qualify one is randomly chosen.

3.4.3 Zero Counts and Smoothing

The set of conditional probabilities used in line 11 of *classify* (Figure 12) can be unreliable for small data sets. A standard statistical technique is to incorporate a small-sample correction into the observed probabilities [68]. This is called *smoothing* in [32] and helps eliminate unreliable estimates and zero-counts. Let $P(\mathbf{x},\mathbf{y},c_i)$ and $P(\mathbf{y},c_i)$ be the observed frequencies in D of $\{\mathbf{x},\mathbf{y},c_i\}$ and $\{\mathbf{y},c_i\}$ respectively (**x** and **y** are itemsets). Instead of $P(\mathbf{x}|\mathbf{y},c_i)=P(\mathbf{x},\mathbf{y},c_i)/P(\mathbf{y},c_i)$ we use the *smoothed* conditional probability:

Equation 8
$$P^s(\mathbf{x} \mid \mathbf{y}, c_i) = \frac{|D| \cdot P(\mathbf{x},\mathbf{y},c_i) + n_0 \cdot P(\mathbf{x})}{|D| \cdot P(\mathbf{y},c_i) + n_0}$$

where n_0 is a small smoothing factor indicating the confidence in the observed probabilities. This estimate is based on the assumption that the conditional probabilities $P(\mathbf{x}|\mathbf{y})$ are usually close to the marginal probabilities $P(\mathbf{x})$. Intuitively, it states that the fewer the examples that support the observed conditional probability $P(\mathbf{x}|\mathbf{y},c_i)$, the closer the actual conditional probability is to $P(\mathbf{x})$. This way differences that can be attributed to very small samples are ignored.

In the special case where both $P(\mathbf{x},\mathbf{y},c_i)$ and $P(\mathbf{y},c_i)$ are zero, Equation 8 yields $P^s(\mathbf{x} \mid \mathbf{y},c_i) = P(\mathbf{x})$. In all our experiments n_0 is set to 5.

3.4.4 Experimental Evaluation

We use 21 data sets from UCI ML Repository [63] to evaluate the performance and the behavior of the Local Bayes (LB) algorithm. We compare LB with Naïve Bayes [26], C4.5 [74], TAN [32], and CBA [53], all of which have been briefly described in section 3.2.

In all experiments, accuracy is measured using the holdout method for large data sets (partition the data into a training and a testing set), and 10-fold cross-validation (CV-10) for the small data sets. For all classification methods, the same train/test set split and the same partitioning for cross-validation is employed. Since all methods, except C4.5, deal with discrete attributes, all attributes are discretized using entropy discretization [28] as implemented in the MLC++ system [47]. No discretization was applied for C4.5.

Table 4 provides detailed information on the data sets used and summarizes the accuracies achieved by the four algorithms on the 21 data sets. Keeping in mind that no classification method can outperform all others in all possible domains [W94], we can draw several conclusions from Table 4. Firstly, it is clear that TAN and LB are in general better than the others in terms of prediction accuracy. This is also supported by the fact that they win in 6 (TAN) and 8 (LB) cases, respectively. On the other hand, if the domain concept can naturally be expressed as a set of if-then rules (e.g., data sets Chess, Voting Records), then rule-based classifiers such as C4.5 and CBA still produce better results. TAN and LB are comparable in terms of accuracy. LB, however, has the additional advantage of handling missing values.

For TAN and CBA the parameters are set to the standard values as suggested in the literature. For example, CBA is used with minimum support 1% and minimum confidence 50% plus the pruning option. Since TAN does not handle missing attribute values, its accuracy for data sets containing missing attribute values cannot be reported. Its smoothing parameter is set to 5. For LB, we fix the minimum support threshold to 1% or 5 occurrences (needed for the very small data sets), whichever is larger. The interestingness threshold τ_I was set to 0.04 and the smoothing parameter n_0 is set to 5. This is further discussed and justified in the sequel.

	Data set Properties					Accuracy				
Data Set	**# Attributes**	**# Classes**	**Missing Values**	**# Train**	**# Test**	**NB**	**C4.5**	**TAN**	**CBA**	**LB π_1=0.04**
1 **Adult**	14	2	Yes	32561	16281	0.8412	0.854	N/A	**0.8567**	0.8511
2 **Australian**	14	2	No	690	CV-10	**0.8565**	0.8428	0.8522	0.8551	**0.8565**
3 **Breast**	10	2	Yes	699	CV-10	**0.97**	0.9542	N/A	0.9528	0.9686
4 **Chess**	36	2	No	2130	1066	0.8715	**0.995**	0.9212	0.9812	0.9024
5 **Cleve**	13	2	Yes	303	CV-10	**0.8278**	0.7229	N/A	0.7724	0.8219
6 **Diabetes**	8	2	No	768	CV-10	0.7513	0.7173	0.7656	0.729	**0.7669**
7 **Flare**	10	2	No	1066	CV-10	0.7946	0.8116	0.8264	**0.8311**	0.8152
8 **German**	20	2	No	999	CV-10	0.741	0.717	0.727	0.732	**0.748**
9 **Heart**	13	2	No	270	CV-10	0.8222	0.7669	**0.8333**	0.8187	0.8222
10 **Hepatitis**	19	2	Yes	155	CV-10	0.8392	0.8	N/A	0.802	**0.845**
11 **Letter**	16	26	No	15000	5000	0.7494	0.777	**0.8572**	0.5176	0.764
12 **Lymph**	18	4	No	148	CV-10	0.8186	0.7839	0.8376	0.7733	**0.8457**
13 **Pima**	8	2	No	768	CV-10	**0.759**	0.711	0.7577	0.7303	0.7577
14 **Satimage**	36	6	No	4435	2000	0.818	0.852	**0.872**	0.8485	0.839
15 **Segment**	19	7	No	1540	770	0.9182	**0.958**	0.9351	0.9351	0.9416
16 **Shuttle-small**	9	7	No	3866	1934	0.987	0.995	**0.9964**	0.9948	0.9938
17 **Splice**	59	3	No	2126	1064	**0.9464**	0.933	**0.9463**	0.7003	**0.9464**
18 **Vehicle**	18	4	No	846	CV-10	0.6112	0.6982	**0.7092**	0.6878	0.688
19 **Voting Records**	16	2	No	435	CV-10	0.9034	**0.9566**	0.9332	0.9354	0.9472
20 **Waveform-21**	21	3	No	300	4700	0.7851	0.704	0.7913	0.7534	**0.7943**
21 **Yeast**	8	10	No	1484	CV-10	0.5805	0.5573	0.5721	0.551	**0.5816**

Table 4: Description of data sets and summary of results.

Varying the minimum support threshold helps somewhat to prune the number of itemsets generated. However, as classification data sets often contain a huge number of associations, minsup is not very effective in preventing the combinatorial explosion in the number of generated itemsets; otherwise many useful ones are omitted. Indeed, the prediction accuracy of LB is very stable with respect to the minimum support threshold. In most datasets we observed that the accuracy slowly decreases with increasing minimum support. As long as the minimum support is low enough to enable the presence of important local patterns, the interestingness threshold is the key factor influencing the number of itemsets generated and the classification accuracy.

Table 5 and Table 6 provide more insights into the behavior of LB. Table 5 shows the accuracy and Table 6 the number of itemsets used by LB for values of the interestingness threshold τ_I varying from 0.02 to 0.06. The last column of Table 6 also provides the number of itemsets used by NB; this is the minimum number of itemsets LB can possibly use. It is apparent that LB is relatively stable with respect to τ_I. Nevertheless, its performance can be boosted even further if the specific aspects of each data set are taken into account.

	Interestingness threshold τ_I								
Data Set	**0.02**	**0.025**	**0.03**	**0.035**	**0.04**	**0.045**	**0.05**	**0.055**	**0.06**
1 **Adult**	**0.8608**	**0.86**	**0.8577**	0.8547	0.8511	0.8499	0.8499	0.8496	0.8496
2 **Australian**	0.8449	0.8507	**0.8609**	0.8551	**0.8565**	0.8478	**0.8594**	**0.8638**	**0.8725**
3 **Breast**	0.9557	0.9628	0.9642	0.9686	0.9686	0.9671	0.9642	0.9657	0.9671
4 **Chess**	0.9212	0.9053	0.9071	0.9184	0.9024	0.9034	0.878	0.8893	0.8812
5 **Cleve**	0.8217	0.7922	0.7991	0.8024	0.8219	0.8253	**0.8319**	**0.8285**	0.8252
6 **Diabetes**	0.7604	0.7617	0.7604	**0.7682**	**0.7669**	0.7643	0.7643	0.763	0.7604
7 **Flare**	0.8274	0.8227	0.8199	0.8208	0.8152	0.8161	0.8161	0.818	0.818
8 **German**	0.738	**0.752**	**0.75**	**0.747**	**0.748**	**0.746**	**0.742**	**0.742**	**0.746**
9 **Heart-**	0.8185	0.8185	0.8222	0.8296	0.8222	**0.837**	0.8333	0.8333	0.8333
10 **Hepatitis**	0.82	**0.8775**	**0.8708**	**0.8579**	**0.845**	**0.8708**	**0.865**	**0.8583**	**0.8588**
11 **Letter**	0.794	0.7902	0.776	0.7692	0.764	0.7564	0.7576	0.7554	0.7538
12 **Lymph**	0.8114	0.8243	0.8186	**0.8519**	**0.8457**	**0.8452**	**0.8586**	**0.8652**	**0.8719**
13 **Pima**	0.7421	0.7499	0.7512	0.7538	0.7577	0.7525	0.7512	0.7525	0.7525
14 **Satimage**	0.844	0.8465	0.8415	0.8395	0.839	0.8405	0.8325	0.8255	0.821
15 **Segment**	0.9416	0.9403	0.9416	0.9351	0.9416	0.9416	0.9364	0.9299	0.926
16 **Shuttle-small**	0.9938	0.9943	0.9943	0.9943	0.9938	0.9938	0.9938	0.9943	0.9943
17 **Splice**	0.9258	0.9417	**0.9568**	**0.953**	**0.9464**	0.9445	0.9436	0.9436	0.9436
18 **Vehicle**	0.6786	0.6962	0.6821	0.6822	0.688	0.6809	0.6844	0.6856	0.688
19 **Voting Records**	**0.9564**	0.9495	0.9379	0.9449	0.9472	0.9403	0.9449	0.9425	0.9495
20 **Waveform-21**	0.7851	0.7719	0.784	**0.7915**	**0.7943**	**0.7972**	**0.8009**	**0.797**	**0.7945**
21 **Yeast**	0.5749	**0.5829**	**0.5809**	**0.5809**	**0.5816**	**0.5823**	**0.5836**	**0.5836**	**0.5829**

Table 5: Accuracy of LB for various values of τ_I; values which outperform all other methods in Table 4 are in bold.

	Interestingness threshold τ_I									
Data Set	**0.02**	**0.025**	**0.03**	**0.035**	**0.04**	**0.045**	**0.05**	**0.055**	**0.06**	**NB**
1 **Adult**	291	248	214	199	193	186	179	176	175	149
2 **Australian**	305	181	137	110	89	76	68	63	60	50
3 **Breast**	133	92	70	58	52	46	41	37	35	30
4 **Chess**	2801	1340	812	622	476	371	308	259	233	75
5 **Cleve**	329	202	135	97	78	67	60	55	49	29
6 **Diabetes**	65	52	42	35	30	27	26	25	24	16
7 **Flare**	164	134	109	96	82	73	71	66	58	29
8 **German**	326	216	166	131	116	100	92	84	79	62
9 **Heart**	251	151	101	76	62	52	47	42	40	19
10 **Hepatitis**	1625	803	475	324	238	180	147	123	107	34
11 **Letter**	782	495	360	301	266	241	222	211	205	172
12 **Lymph**	2297	1233	760	503	356	277	226	189	162	53
13 **Pima**	66	50	40	34	30	28	27	26	25	17
14 **Satimage**	6677	4790	3547	2294	1774	1773	1445	1219	1035	390
15 **Segment**	1087	878	714	602	524	454	391	341	310	154
16 **Shuttle-small**	185	155	143	126	109	99	96	91	88	58
17 **Splice**	1675	767	501	381	320	294	290	290	290	290
18 **Vehicle**	4535	2812	1907	1371	1055	850	697	596	525	73
19 **Voting Records**	1908	1185	780	534	377	290	238	208	185	48
20 **Waveform-21**	3088	1570	890	632	478	383	304	248	204	47
21 **Yeast**	44	35	33	32	32	31	31	30	29	28

Table 6: Number of itemsets used by LB for various values of τ_I; the last column displays the number of itemsets used by NB.

Figure 14 illustrates the effect of varying the interestingness threshold τ_I on two different data sets. It displays the accuracy and the number of itemsets used by LB as a function of τ_I for the two extreme cases, Lymph and Adult data set. Varying τ_I has clearly the opposite effect for these two data sets. The increased number of itemsets produced by lowering τ_I, causes a 7.5% reduction of the error rate in Adult (from 15.05% to 13.92%), whereas in Lymph the error rate increases by almost 47% (from 12.81% to 18.86%). The reason is the size of the data sets. Adult is the largest and Lymph the smallest data set.

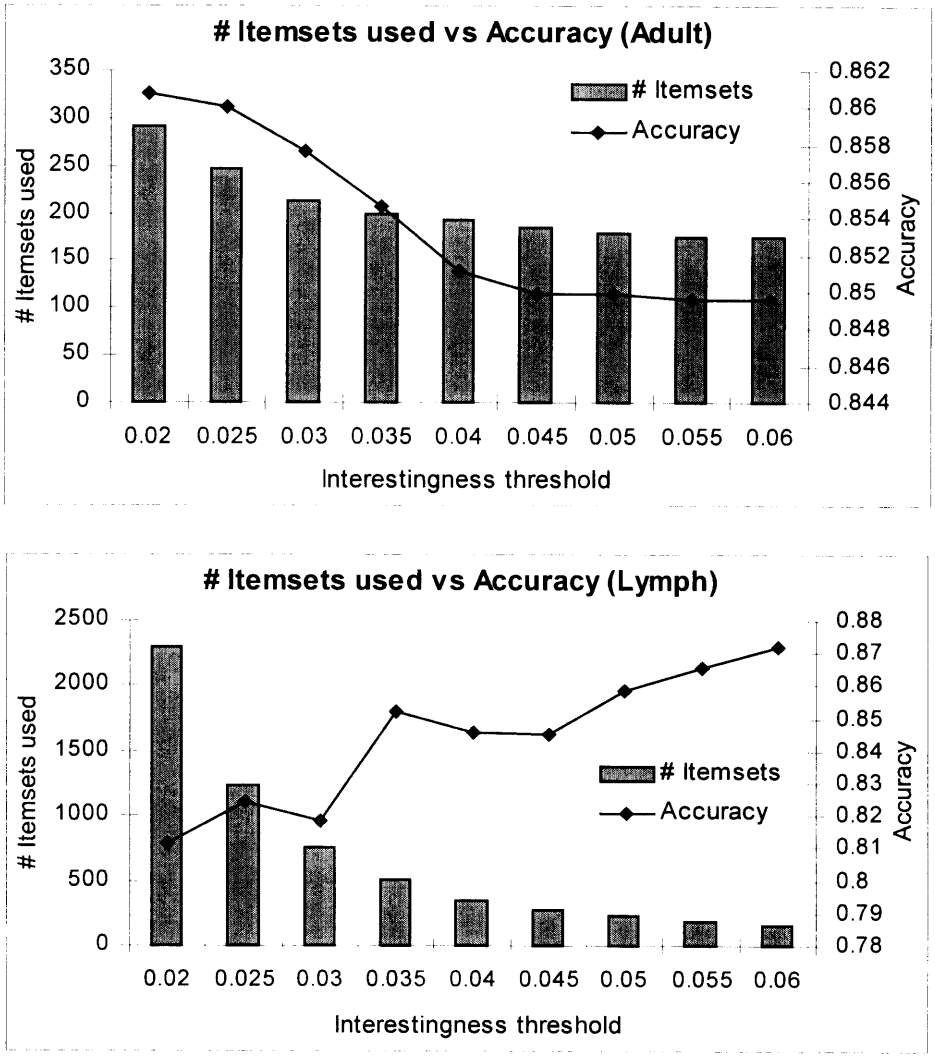


Figure 14. Accuracy and the number of itemsets used by LB for the Adult and Lymph data sets as a function of τ_I.

The data set size in general is responsible for this effect. Lowering τ_I from 0.06 to 0.02 contributes to a reduction of the error rate in five out of the seven largest data sets used in our experiments, whereas in 12 out of the 14 smallest data sets the error rate increases. This is expected since lower values of τ_I allow more and longer itemsets to be discovered, which tend to cause overfitting of the training data and unreliable estimates if the data set is small.

Figure 15 further supports this claim. It shows the difference in the accuracy of LB when lowering τ_I from 0.06 to 0.02 as a function of the data set size. Figure 15 indicates that lowering τ_I in order to discover a large number of itemsets is beneficial only if the data set is large enough to provide reliable estimates. This is further supported by experimental results in [25] where NB is found to outperform more complex models (like C4.5) if the training sample size is small. However, the performance of NB cannot scale up if more data are available.

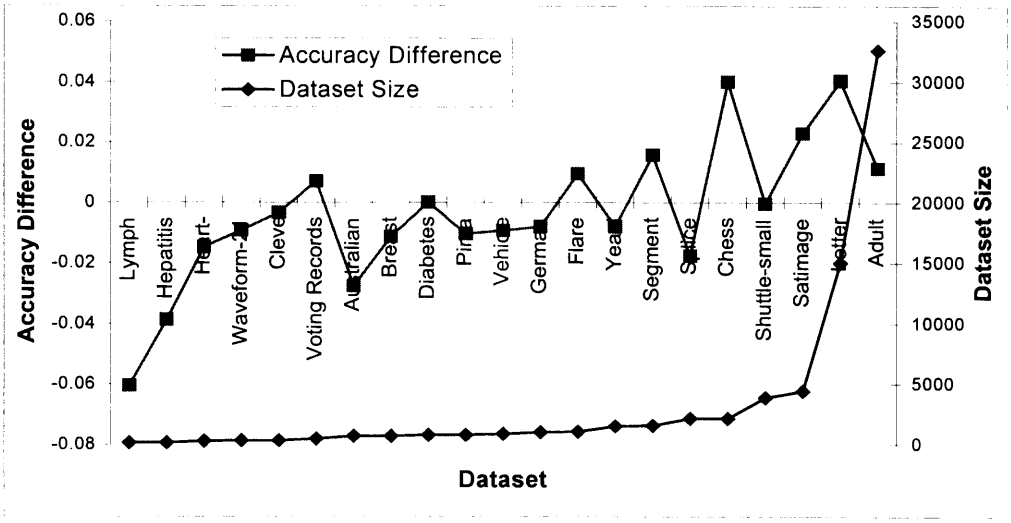


Figure 15. Accuracy change when decreasing τ_I from 0.06 to 0.02 with *minsup* set to 1%. Domains are sorted in increasing data set size. Positive change indicates accuracy improvement when τ_I decreases.

In contrast to NB, LB can take advantage of the largest samples by adjusting τ_I and allowing more complex models to be built. In order to directly compare LB with the other methods (see Table 4) we simply fixed τ_I to 0.04, the middle of the range of values we used in our experiments to balance performance with large and small data sets. However, making τ_I a function of the size of the training set would boost the accuracy of LB even more. For example, from Table 5 it follows that taking τ_I as 0.02 instead of 0.04 for the two largest data sets, would result in LB being also the best classification method for Adult and being second for Letter.

The classification time of LB mainly depends on the number of discovered itemsets. As shown in Table 6, for most data sets a fairly small number of itemsets is generated and is sufficient to produce accurate classifications. Table 7 shows the CPU time in seconds required for training and testing of LB with τ_I=0.02 and τ_I=0.04 respectively. The experiments were carried out on a 400 MHz Pentium II Windows NT workstation. For the 14 small data sets, the average time over the 10 CV-folds is reported. It is apparent from Table 7 that LB is fast, though our implementation has not focused on optimizing speed. Our itemset mining routine, *genItemsets*, could, for instance, be extended by using the additional pruning strategies proposed in [7] and classification speed can be further improved by indexing the set F of itemsets.

		τ_l =0.02		τ_l =0.04	
	Data Set	**Train**	**Test**	**Train**	**Test**
1	**Adult**	12.7	5.9	12.7	5.5
2	**Australian**	0.37	0.06	0.27	0.03
3	**Breast**	0.13	0.03	0.1	0.02
4	**Chess**	7.9	9.1	3.9	2.2
5	**Cleve**	0.17	0.05	0.09	0.02
6	**Diabetes**	0.09	0.02	0.07	0.01
7	**Flare**	0.17	0.04	0.14	0.03
8	**German**	0.67	0.08	0.6	0.05
9	**Heart**	0.12	0.04	0.07	0.01
10	**Hepatitis**	0.6	0.21	0.12	0.03
11	**Letter**	14.9	4.8	11.4	3.6
12	**Lymph**	0.97	0.33	0.26	0.05
13	**Pima**	0.08	0.03	0.08	0.01
14	**Satimage**	140.9	16.7	57.1	6.9
15	**Segment**	3.7	0.77	2.8	0.5
16	**Shuttle-small**	0.8	0.4	0.7	0.4
17	**Splice**	18.2	5.3	17.9	2.8
18	**Vehicle**	4.3	1.1	1.6	0.27
19	**Voting Records**	1.2	0.31	0.49	0.07
20	**Waveform-21**	2.1	21.5	0.6	4.8
21	**Yeast**	0.13	0.04	0.12	0.03

Table 7: CPU time in seconds for training and testing LB.

3.5 Using Chi-square Tests to measure the interestingness of itemsets

A key factor affecting the performance of Local Bayes is the accurate identification of interesting itemsets. The interestingness of an itemset l is defined in terms of the error when estimating $P(l,c_i)$ using subsets of l. Let l be an itemset of size $|l|$ and l_j, l_k be two ($|l|$-1)-itemsets obtained from l by omitting the j^{th} and k^{th} item respectively. We can use l_j, l_k to produce an estimate $P_{j,k}(l,c_i)$ of $P(l,c_i)$:

Equation 9 $$P_{est}(l,c_i) = P_{j,k}(l,c_i) = \frac{P(l_j,c_i)\cdot P(l_k,c_i)}{P(l_j \cap l_k,c_i)}$$

Our goal is to keep those itemsets only for which the corresponding observed probabilities differ a great deal from the estimated ones. Information-theoretic metrics, such as the cross-entropy (or Kullback-Leibler distance), are widely used [32] as a measure of the distance between the observed and the estimated probability distributions:

Equation 10 $$D_{KL}(P,P_{est}) = \sum_{l,c_i} P(l,c_i)\log\frac{P(l,c_i)}{P_{est}(l,c_i)}$$

In our case, however, the goal is to measure the distance between specific elements of the probability distribution. Consider for example a case with two variables A_1 and A_2 and $|val(A_1)| = |val(A_2)| = 4$. The corresponding sixteen 2-itemsets define the complete observed joint probability distribution $P(A_1,A_2,C)$. A high value for such metrics suggests that the class-supports of the corresponding itemsets cannot be accurately approximated *on average* by the class-supports of their subsets. To measure the accuracy of the approximation for individual itemsets in section 3.4.1 we defined the interestingness $I(l|l_j,l_k)$ of l with respect to its subsets l_j and l_k as:

Equation 11 $$I(l \mid l_j, l_k) = \sum_{c_i} \left| P(l, c_i) \log \frac{P(l, c_i)}{P_{est}(l, c_i)} \right|$$

This ad-hoc measure presents certain drawbacks with respect to its ability to identify interesting local patterns. Consider for example a domain with two classes and an itemset l for which $P(l,c_1) = 0$, $P(l,c_2)=0.15$ and $P_{est}(l,c_1) = 0.1$, $P_{est}(l,c_2)=0.15$. Although this is indeed a very interesting itemset, since the estimated probability for c_1 greatly differs from the observed one, $I(l|l_j,l_k) = 0$ and l is discarded as non-interesting. In addition, our interestingness measure (but also every information-theoretic measure) suffers from the fact that it ignores the sample size and assumes that the sample probability distribution is equal to the population probability distribution thus ignoring the possibility that the differences occurred purely because of chance.

In the sequel we describe the application of chi-square (χ^2) tests to overcome these problems. We reduce the problem of deciding whether an itemset is interesting to applying a hypothesis-testing procedure on the following hypotheses:

H_0: $P(l,c_i) = P_{est}(l,c_i)$, i.e., l is not-interesting

H_1: $P(l,c_i) \neq P_{est}(l,c_i)$, i.e., l is interesting

To test the hypotheses we calculate the χ^2 test statistic with $|C|$ degrees of freedom. ($|D|$ is the database size, |C| the number of classes):

Equation 12 $$\chi_l^2 = \sum_{j=1}^{|C|} \frac{\left(P(l,c_i)\cdot |D| - P_{est}(l,c_i)\cdot |D|\right)^2}{P_{est}(l,c_i)\cdot |D|} = \sum_{j=1}^{|C|} \frac{\left(P(l,c_i) - P_{est}(l,c_i)\right)^2}{P_{est}(l,c_i)} * |D|$$

If $\chi_l^2 > \chi^2_{p,|c|}$ the null hypothesis H_0 is rejected and l is considered interesting. The statistical-significance threshold p is user-defined but should in general be high i.e., $p<0.05$, since discovering non-interesting itemsets does not improve the accuracy and unnecessarily increases the complexity of the resulting classifier. The degrees of freedom for the test are $|C|$ since the sums of the expected and the observed frequencies of an itemset are generally different [72]. If the degrees of freedom are two or less, Yates correction is applied (subtracting 0.5 from the absolute difference in Equation 12 before squaring, when this difference exceeds 0.5).

A problem associated with χ^2 tests is that the estimated frequencies $P_{est}(l,c_i)\cdot|D|$ in each term of Equation 12 should not be too small; otherwise the test is sensitive to errors. To overcome this problem we apply a merging step before calculating the χ^2-statistic. During this step, the class with the smallest frequency is merged with the immediate larger class to form a composite class containing the sum of the frequencies. The corresponding observed frequencies are merged also and the degrees of freedom (*df*) are reduced by one. The merging phase stops when all expected frequencies are large enough or when all class-frequencies are merged. Following standard statistical practice we set the minimum value of an expected frequency to 5 if $df = 2$ and 3 if $df>2$; otherwise it is merged.

As a result of the merging step, each itemset l has a value χ_l^2 that refers to different degrees of freedom df_l. To compare these values with the minimum required threshold $\chi^2_{p,|c|}$ we need a degrees-of-freedom-independent test. For that reason we take advantage of the fact that the value $t = \sqrt{2 \cdot \chi^2} - \sqrt{2 \cdot df - 1}$ approximately follows the normal distribution and therefore the modified requirement for an itemset to be interesting becomes:

Equation 13 $$t = \sqrt{2 \cdot \chi_l^2} - \sqrt{2 \cdot df_l - 1} > \sqrt{2 \cdot \chi^2_{p,|C|}} - \sqrt{2 \cdot |C| - 1}$$

Note that although t can now take negative values as well, the requirement for an itemset to be interesting remains that its t value is bigger than the threshold of Equation 13 which is determined by the required statistical significance level p and the number of classes $|C|$.

3.5.1 Experimental evaluation of the effectiveness of Chi-Square tests as a measure of interestingness

To evaluate the performance of LB with the χ^2 tests (LB-chi2), we use 23 data sets from the UCI ML Repository [63] with a special preference on the largest and more challenging ones in terms of achievable classification accuracy. We compared LB-chi2 with the originally proposed version of LB, the Naïve Bayes classifier [26] (since in the extreme case if only 1-itemsets are used LB reduces to NB), Quinlan's Decision Tree classifier C4.5 [74], and TAN [32]; a Bayesian Network classifier that relaxes the independence assumptions of NB by using some pairs of attributes.

Accuracy was measured either using 10-fold cross validation (CV-10) for small data sets or the holdout method (training and testing set split) for the larger ones. The train and test set splits and the cv-folds were the same for all results reported. Since all methods except C4.5 only deal with discrete attributes, we used entropy-based [28] discretization for all continuous attributes. No discretization was applied for C4.5.

The factor most affecting the results is the p-value of the χ^2 tests. We experimented with 0.01, 0.025, 0.005, 0.001 and 0.0005, and selected 0.005 as the most effective one. Higher p-values slowly deteriorated the accuracy and tended to produce more complex, larger and slower classifiers. This is natural since high p-values cause more itemsets to be characterized as interesting. On the other hand, values below p=0.005 caused most of the itemsets to be rejected as non-interesting and generated simplistic classifiers with poor accuracy. The effects of the varying p-values on the average accuracy and classifier size can be seen in Figure 16.

		NB	C4.5	TAN	LB	LB-chi2
1	**Average Accuracy**	0.8187	0.8147	0.8376	0.8332	0.8434
2	**Average Rank**	3.696	3.739	2.739	2.652	1.913
3	**No wins of LB-chi2 vs.:**	19 – 4	19 - 4	16 - 6	15-7	--
4	**1-side Paired t-test**	0.9995	0.9991	0.9940	0.9828	--
5	**Wilcoxon paired signed rank test**	>0.995	>0.995	>0.99	>0.975	--

Table 8: Comparison of the classifiers according to various criteria.

	Data set Properties						Accuracy				
Data Set	# Attributes	# Items	# Classes	Missing Val.	# Train	# Test	NB	C4.5	TAN	LB	LB-chi2
1 **Adult**	14	147	2	Yes	32561	16281	0.8412	0.854	0.8571	0.8511	**0.8668**
2 **Australian**	14	48	2	No	690	CV-10	0.8565	0.8428	0.8522	0.8565	**0.8609**
3 **Breast**	10	28	2	Yes	699	CV-10	0.97	0.9542	0.9671	0.9686	**0.9714**
4 **Chess**	36	73	2	No	2130	1066	0.8715	**0.995**	0.9212	0.9024	0.9418
5 **Cleve**	13	27	2	Yes	303	CV-10	**0.8278**	0.7229	0.8122	0.8219	0.8255
6 **Flare**	10	27	2	No	1066	CV-10	0.7946	0.8116	**0.8264**	0.8152	0.818
7 **German**	20	60	2	No	999	CV-10	0.741	0.717	0.727	0.748	**0.75**
8 **Heart**	13	17	2	No	270	CV-10	0.8222	0.7669	**0.8333**	0.8222	0.8185
9 **Hepatitis**	19	32	2	Yes	155	CV-10	0.8392	0.8	0.8188	**0.845**	0.8446
10 **Letter**	16	146	26	No	15000	5000	0.7494	0.777	0.8572	0.764	**0.8594**
11 **Lymph**	18	49	4	No	148	CV-10	0.8186	0.7839	0.8376	0.8457	**0.8524**
12 **Pendigits**	16	151	10	No	7494	3499	0.8350	0.923	0.9360	0.9182	**0.9403**
13 **Pima**	8	15	2	No	768	CV-10	**0.759**	0.711	0.7577	0.7577	0.7564
14 **Pima Diabetes**	8	14	2	No	768	CV-10	0.7513	0.7173	0.7656	**0.7669**	0.763
15 **Satimage**	36	384	6	No	4435	2000	0.818	0.852	0.872	0.839	**0.8785**
16 **Segment**	19	147	7	No	1540	770	0.9182	**0.958**	0.9351	0.9416	0.9429
17 **Shuttle-small**	9	50	7	No	3866	1934	0.987	0.995	**0.9964**	0.9938	0.9948
18 **Sleep**	13	113	6	No	70606	35305	0.6781	0.7310	0.7306	0.7195	**0.7353**
19 **Splice**	59	287	3	No	2126	1064	**0.9464**	0.933	0.9463	**0.9464**	0.9408
20 **Vehicle**	18	69	4	No	846	CV-10	0.6112	0.6982	0.7092	0.688	**0.7187**
21 **Vote Records**	16	48	2	No	435	CV-10	0.9034	**0.9566**	0.9332	0.9472	0.9334
22 **Waveform-21**	21	44	3	No	300	4700	0.7851	0.704	0.7913	**0.7943**	0.7913
23 **Yeast**	8	18	10	No	1484	CV-10	0.5805	0.5573	0.5721	**0.5816**	**0.5816**

Table 9: Summary Table of datasets and results.

Table 8 provides a comparison of the algorithms in the 23 data sets according to five criteria. LB-chi2 outperforms all others according to all criteria indicating that it is indeed a very accurate classifier. The criteria used are: (1) Average Accuracy of the classifiers, (2) Average Rank (smallest values indicate better performance on average), (3) The number of wins–losses of LB-chi2 against other algorithms, and the statistical significance of the improvement of LB-chi2 against each algorithm using (4) a one-sided paired t-test and (5) a Wilcoxon paired, signed, one-sided, rank test.

Table 9 provides information about the data sets and lists the accuracies of the classifiers on all data sets. Table 10 shows the training and testing time of LB-chi2, measured on a 400MHz Pentium WinNT PC. Noticeably, the biggest improvements in accuracy against the original LB came mostly from the largest data sets; this indicates the inefficiency of the originally used interestingness metric in such cases.

	Data Set	Time LB-chi2			Data Set	Time LB-chi2	
		Train	Test			Train	Test
1	**Adult**	48.81	37.04	13	**Pima**	0.18	0.03
2	**Australian**	0.18	0.03	14	**Pima-Diabetes**	0.07	0.02
3	**Breast**	0.11	0.02	15	**Satimage**	392.60	83.75
4	**Chess**	1.99	2.20	16	**Segment**	2.28	1.16
5	**Cleve**	0.07	0.01	17	**Shuttle-small**	1.40	0.78
6	**Flare**	0.42	0.07	18	**Splice**	476.71	621.97
7	**German**	0.05	0.01	19	**Sleep**	3.24	3.07
8	**Heart**	0.05	0.01	20	**Vehicle**	1.24	0.23
9	**Hepatitis**	109.29	56.90	21	**Voting Records**	0.13	0.04
10	**Letter**	0.07	0.02	22	**Waveform-21**	0.1	2.724
11	**Lymph**	44.23	22.58	23	**Yeast**	0.15	0.04
12	**Pendigits**	0.06	0.02				

Table 10: Training and testing time of LB-chi2 for the 23 datasets.

The p-value for chi2-LB was set to 0.005 and to facilitate more accurate χ^2 tests the minimum support was set to *max{10, 2*|c|}*. Although this is a minimum requirement in order for the test statistic to be accurate, this can be further increased in order to reduce both the training time and the size of the classifier as discussed below.

3.5.2 Pruning criteria: Trading off accuracy for simplicity and speed.

A difficult challenge in the design of classifiers is preventing overfitting and generating simple models. Simple models are not only easily interpretable but also generalize better on unseen data and are faster to build and evaluate. In LB overfitting translates to the discovery of "too many itemsets" and this is particularly true in domains with many multi-valued attributes, where the search space is huge. χ^2 tests significantly reduce the number of discovered itemsets to a tractable amount. However, some other pruning criteria can potentially reduce the number of itemsets and accelerate both the learning and classification phase.

Support-based pruning is used by many classification methods including decision trees, where a leaf is not expanded if it contains less than a minimum number of cases. In section 3.5.3 we evaluate the effect of support pruning on the accuracy of LB.

A somewhat more effective pruning criterion is the conditional entropy of the class C given an itemset l:

Equation 14 $$H(C|l) = \sum_{j=1}^{|C|} P(c_i|l) \cdot \log P(c_i|l)$$

Conditional entropy takes values ranging from zero (if *l* only appears with a single class) to *log(|C|)*, if *l*'s appearances are uniformly distributed among the classes. If *H(C|l)* is very small, *l* bears almost certainty about a class and therefore need not be expanded. In the next section we show that the introduction of a relatively low conditional entropy threshold often reduces the size of the classifier without significantly affecting its accuracy.

3.5.3 Experimental evaluation of the pruning criteria

Figure 16 illustrates the effect of conditional entropy pruning on the average accuracy (10a) and size (10b) of LB. Since the number of classes is different among the datasets, the minimum threshold *minH* is expressed as a percentage of the maximum conditional entropy $\log |C|$. In a 4-class domain, for example, a value of 0.2 implies that $minH = 0.2 \cdot \log 4 = 0.4$. Values for *minH* of up to $0.3 \cdot \log |C|$ have little impact on the accuracy while at the same time reducing the size of the classifier. The rightmost values of the graph correspond to maximum pruning where only 1-itemsets are used and therefore represent the accuracy and size of a Naïve Bayes classifier.

Support pruning has a more drastic effect on the size of the classifier as can be seen in Figure 17. This is particularly true on large data sets like "Sleep" where 10 occurrences for an itemset *l* represent a probability *P(l)=0.0001*. Increasing the minsup threshold to 0.005 in this data set reduced the number of itemsets discovered from 31000 to 7500 while the accuracy fell only slightly, from 0.7336 to 0.727.

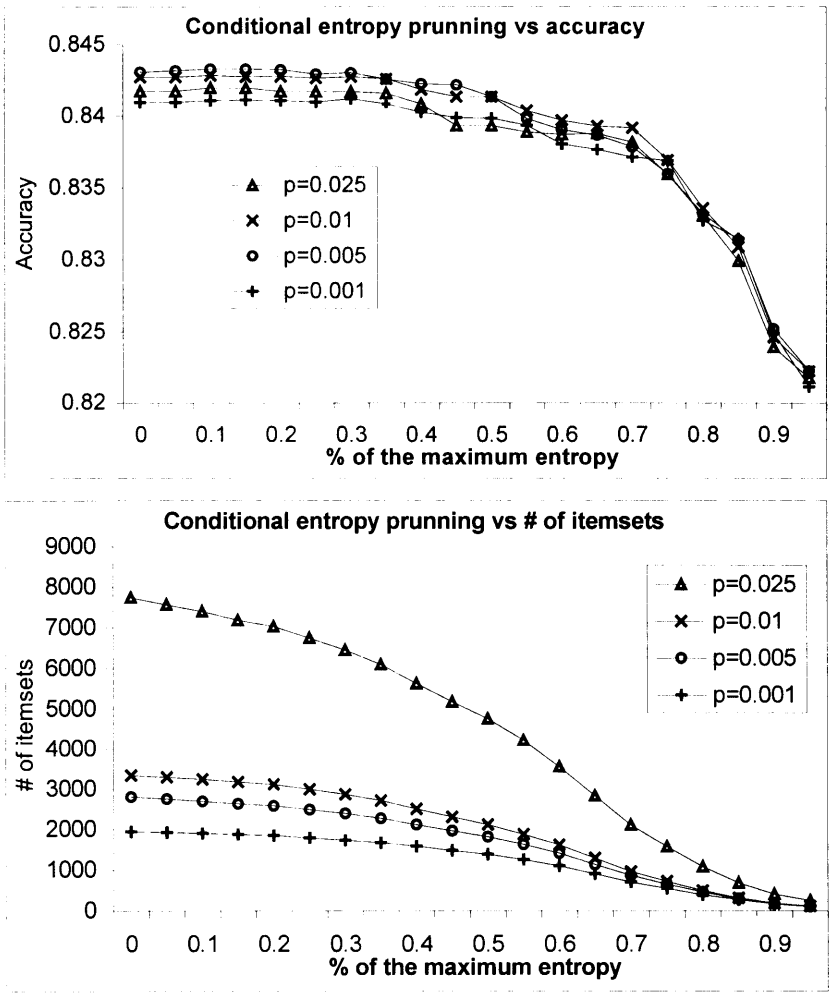


Figure 16: Effect of conditional entropy pruning on (a) accuracy and (b) size of classifier for four different p-value thresholds. x-axis contains the threshold as a percentage of the maximum conditional entropy $log|C|$

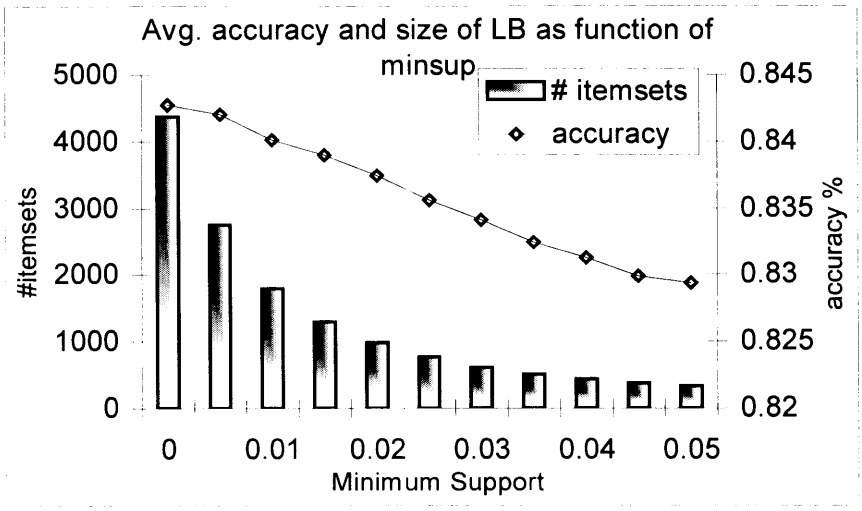


Figure 17: Effect of support pruning on average accuracy and size of LB.

3.6 Conclusions

In this chapter, a new classification algorithm, called Local Bayes (LB), has been introduced. In the training phase, LB employs an Apriori-like frequent pattern mining algorithm to discover frequent and interesting itemsets of arbitrary size together with their class supports. The class support is an estimate of the probability that the pattern occurs with a certain class label. Upon arrival of a new case to classify, a local classification model is built on the fly depending on the evidence of the new case.

We prove that the models constructed by LB are local Bayesian networks. In the extreme case, where all the discovered itemsets are of size one only, LB reduces to Naïve Bayes. Local Bayes is especially suitable for large and complex data sets and is shown to consistently outperform Naïve Bayes. Naïve Bayes (NB) is more accurate on only three of twenty one data sets; and NB's accuracy is on average only 0.28% higher than LB's in these three cases. LB is also shown to be very competitive compared with other state of the art classification algorithms. It is the most accurate in eight out of twenty one cases while still being very efficient. Moreover, it has been shown that the accuracy of LB can be easily improved by exploring data specific properties such as the size of the data.

CHAPTER 4. LOCAL BAYESIAN CLASSIFICATION FOR TEXT CATEGORIZATION

The Naïve Bayes (NB) classifier has long been considered a core methodology in text classification mainly due to its simplicity and computational efficiency. There is an increasing need however for methods that can achieve higher classification accuracy while maintaining the ability to process large document collections. In this chapter, we present our work on the use of local Bayesian Classification for text categorization [59]. We examine text categorization methods from a perspective that considers the tradeoff between accuracy and scalability to large data sets and large feature sizes. We start from the observation that Support Vector Machines, one of the best text categorization methods, cannot scale up to handle the large document collections involved in many real word problems. We then consider Bayesian extensions to NB that achieve higher accuracy by relaxing its strong independence assumptions. Our experimental results show that Local Bayes, the association-based partially lazy classifier described in the previous chapter can achieve a good tradeoff between high classification accuracy and scalability to large document collections and large feature set sizes.

4.1 INTRODUCTION

Text classification is the supervised learning task of assigning natural language text documents to one or more predefined categories or classes according to their content. While it is a task with a history of almost forty years of research [51], it has recently attracted an increasing amount of attention due to the ever-expanding amount of text documents available in digital form on the World Wide Web, electronic mail, net news and digital libraries. Its applications span a number of areas including sorting email, filtering junk email, cataloguing news articles, providing relevance feedback and reorganizing large document collections.

While the quality of classification is the primary goal of competitive methods, the explosive growth in the amounts of textual data collected and stored introduces the need for methods that can scale up to handle large textual databases. In contrast to relational data mining, where scalability is a main issue, the trade-off between quality and scalability has been largely ignored by a number of studies for text classification. Strikingly, the Naïve Bayes (NB) classifier [26], probably the most well studied and longest used method for text classification and Information Retrieval [51], attributes its success and longevity not only to its accuracy but also to its simplicity and low computational requirements. It scales up linearly both with the number of features and with the number of documents.

Although NB was used in the majority of supervised IR and classification for a long time [51], a number of recent studies have shown that alternative, more complex algorithms often outperform NB in terms of classification performance. Such algorithms originate from a variety of research areas such as Nearest Neighbor [56], neural networks [66], regression [83], rule-induction [4],[20] and Support Vector Machines [41]. The accuracy improvements are often significant in a variety of real-world data sets (see also [84] for a comparative study of many algorithms).

Support Vector Machines (SVM) have been recognized as one of the most effective text classification methods. Unfortunately, they cannot be applied to large volumes of data since their training time is quadratic in the number of training examples [42]. In this work, we investigate whether extensions of NB can achieve higher accuracy performance without significant sacrifices in terms of scalability.

A careful examination of successful text classification methods at some level of abstraction suggests that their efficiency is to a large extent attributed to some common properties that NB does not posses and of which we focus on the following two: First, most methods deviate from the independence assumption and capture relationships among sets of words. The relaxation of the independence assumptions allows more complex models to be built. Such models are expected to capture more aspects of the reality. Second, a number of methods focus on the use of context which seems to be very important in text classification [20].

Context-insensitive classifiers build global models of the data, where the influence of each word to the final decision is independent of the presence/absence of other words in the same document. In natural languages, however, the meaning of words is clearly affected by the context in which they are used. The word "adopt" for example has a very different meaning in the presence of the word "child" than in the presence of the word "strategy". Context-sensitive learning methods aim at higher performance by capturing such context-specific properties of the data. It is noteworthy that these two properties are *orthogonal,* i.e., a classifier may or may not posses one independently of the other.

Extensions of NB along these directions have been shown to yield higher classification accuracy, but often at the cost of excessive computational requirements [27]. In this chapter, we focus on the use of two such extensions, namely Tree Augmented Naïve Bayes classifier (TAN) [32] and LB or Local Bayes Classifier [60], [61]. TAN is a global Bayesian classifier that relaxes the independence assumptions of NB by taking into account dependencies between pairs of features. LB is a very competitive local/lazy Bayesian classifier constructed from associations among feature-values, also called itemsets. Our experimental results show that LB, with appropriate text representation, can achieve a very good tradeoff between accuracy and scalability.

The rest of this chapter is structured as follows. The next section contains our classification framework and discusses alternative forms of text representation. Section 4.3 provides an overview of NB and its two extensions TAN and LB as well as SVMs and section 4.4 describes our experimental evaluation and discussion. Our conclusions are stated in section 4.5.

4.2 Text Representation and Classification Framework

For simplicity, we chose the *multivariate Bernoulli model* or *binary independence event model* [57] as the basis for the representation of text documents. Documents are represented as vectors of binary features and each feature corresponds to the presence or absence of a single word. This model does not capture a large amount of information. More specifically, both the order of words and the number of their appearances in a document are ignored; documents are represented as "bags of words". An alternative representation is the *multinomial model* where the number of occurrences of the words in a document is also captured. The latter provides more information and indeed it is reported to achieve higher performance for NB in a recent study over many data sets [57]. Its main drawback is that it assumes independence between the occurrences of the same word in a document, an assumption that is regularly violated in real text documents. For our experiments, we chose the multivariate-Bernoulli model for the following reasons:

i. Relaxing the independence assumptions is simpler if the multivariate Bernoulli model is used [57]. This is particularly true when adapting algorithms from the relational machine learning paradigm, where the simple feature-vector representation is dominant. We are currently experimenting with the multinomial model and other models when the independence assumptions are relaxed.

ii. The multinomial model is usually more accurate with a large vocabulary. Although NB scales up linearly in the number of features, the performance of LB and TAN degrades faster as the number of features increases. Moreover, [57] reports that in the Reuters-21578 corpus a small feature size achieves highest performance (and the two models have similar performance). Similarly, [27] reports some of the highest accuracy results for SVMs on the Reuters-21578 dataset, (which we use for our experiments) using only 300 features and the multivariate Bernoulli model.

The training database is a set of N documents $D=\{d_1, d_2, ..., d_N\}$. A vocabulary $V=\{w_1, w_2, ... w_n\}$ contains all n words used as features. This is constructed in the preprocessing stage. Each document d_j is then represented as a binary feature vector $d_j=<W_1, W_2, ..., W_n>$ where the value of feature W_l is one if word w_l is present in d_j and zero otherwise. Finally, each document is labeled with none, one or more than one class from a set of M classes $C=\{c_1, c_2, ..., c_M\}$. The data set is therefore represented as a relational N×n table (if we ignore the classes). This *relational* representation is very common for traditional machine learning algorithms. It is, however, inefficient in text classification since the number of features can be very large.

A large improvement in space requirements is achieved if the data set is stored as a *transactional database* [2]. This requires that words are considered as *items* and each document is stored as a transaction consisting of the items it contains. Each transaction therefore contains exactly as much items as the words that appear in the corresponding document, which can be orders of magnitude less than the vocabulary size [3]. It is not clear to us how TAN can be extended to handle transactional databases. However, LB, which has its roots in association mining, inherently uses this representation and therefore can be easily applied. This is at the cost of ignoring information about words that do not appear in a document. On the other hand transforming the relational database into a transactional one allows absence of words to be captured by LB also. This is discussed in more detail later.

4.3 An overview of the learning methods

In the following sections we describe the classifiers involved in this study namely Naïve Bayes, TAN, LB and Support Vector Machines. We have described NB, TAN and LB in the previous chapters. Nevertheless, the text classification perspective has sufficient differences from classification of relational data, the context and the major difficulties lie in slightly different areas and the terminology used is considerably different to justify a brief introduction of the algorithms from a text classification perspective. This will also enable a smoother sequence of thoughts throughout the chapter.

---

[3] Note the similarity of this representation with the one used by RIPPER [20]where it is called "learning with set-valued features" (see also [19]).

In order to illustrate the properties of the three algorithms, consider the following toy example. A database of documents is available and the vocabulary consists of 5 words $V=\{w_1,...,w_5\}$. In the training phase, NB and TAN will build the global models of Figure 18a and Figure 18b respectively (W_j are the word features that can take values 0 or 1). Note that in addition to the class dependencies induced by NB, TAN enhances the dependency graph with a tree of dependencies among pairs of features (the bold arrows). LB does not build any model but instead discovers a set of itemsets such as $\{\overline{w_1},w_2,w_3\}$ (standing for $\{W_1=0,W_2=1,W_3=1\}$) and $\{w_4,w_5\}$.

Note that if a transactional representation is used, LB can capture information only about the presence of words in a document. However, if supplied with the relational representation both the presence and absence of words are taken into account (absent words are considered distinct items). As discussed in Section 4.4, there is significant difference in the behavior of the former version of LB which we will call LB+ and the latter one which we will name LB+-.

Assume now a query to classify a document d that contains words w_2, w_3, w_4 and w_5, but not w_1. TAN and NB will estimate $P(d,c_i)$ using the induced models and will assign the class with the highest probability:

a) NB: $P(d,c_i) = P(c_i)P(\overline{w_1} \mid c_i)P(w_2 \mid c_i)P(w_3 \mid c_i)P(w_4 \mid c_i)P(w_5 \mid c_i)$

b) TAN: $P(d,c_i) = P(c_i)P(\overline{w_1}, w_2 \mid c_i)P(w_4 \mid w_2,c_i)P(w_3 \mid w_4,c_i)P(w_5 \mid w_4,c_i)$

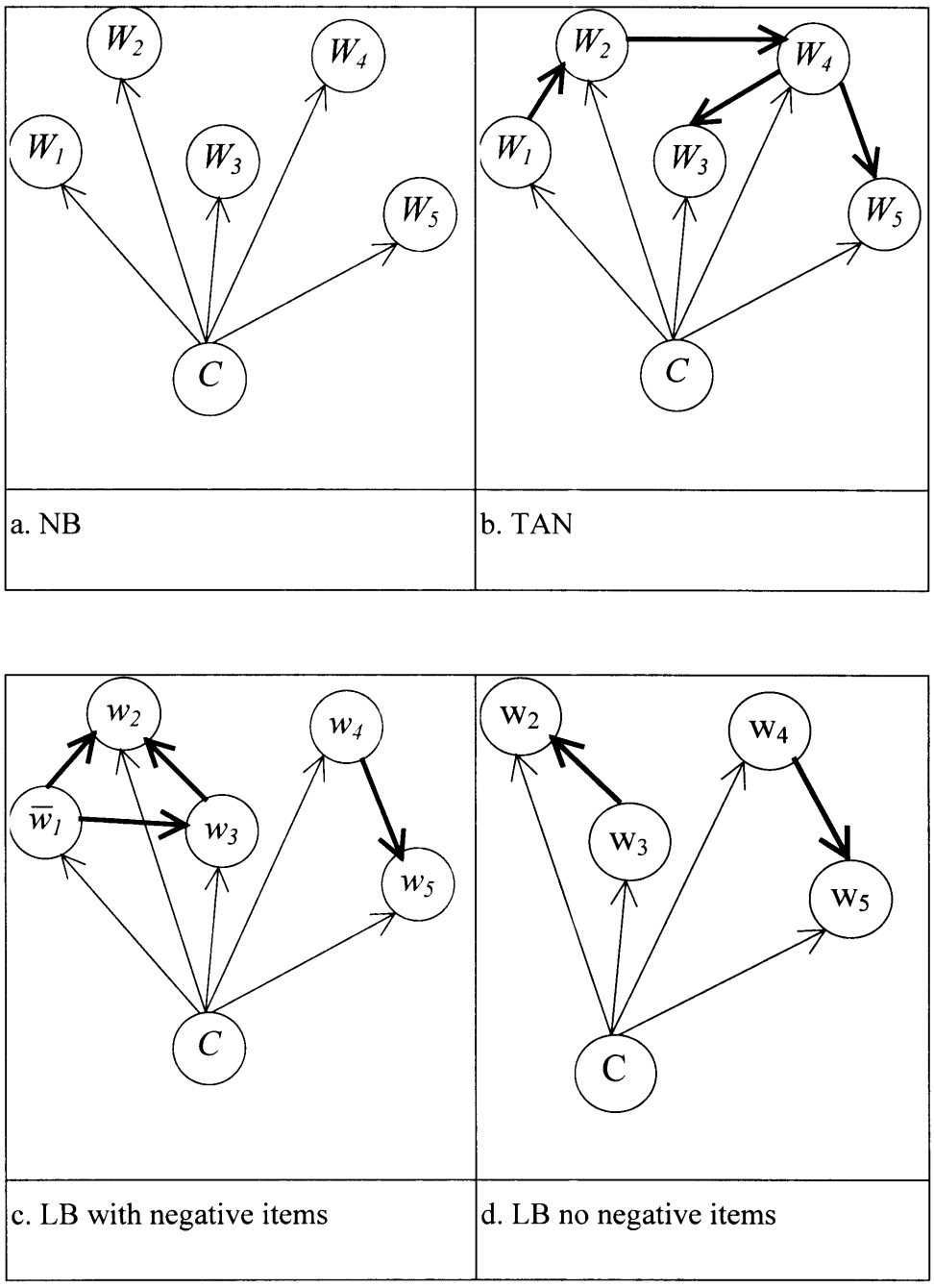


Figure 18: Global (NB and TAN) and local (LB) models built from the three classifiers in the example of section 4.3 for the classification of a document that contains words w_2, w_3, w_4 and w_5 but not w_1.

LB will first select the longest itemsets that are subsets of the query document *d* and then it will build a local model in order to derive an estimation for $P(d,c_i)$. If negative items (i.e., absence of words) are used, the model of Figure 18c is constructed, otherwise item $\overline{w_1}$ is ignored and the model of Figure 18d is created. Computation of $P(d,c_i)$ in the two cases is as follows:

c) LB,with negative items: $P(d,c_i) = P(c_i)P(\overline{w_1}, w_2, w_3 \mid c_i)P(w_4 \mid c_i)P(w_5 \mid w_4, c_i)$

d) LB, no negative items: $P(d,c_i) = P(c_i)P(w_2, w_3 \mid c_i)P(w_4, w_5 \mid c_i)$

4.3.1 Support Vector Machines

Support Vector Machines (SVM) are a relatively new approach for binary classification problems that stems from statistical learning theory. In their simplest form, *linear SVMs* treat examples as points in a high-dimensional space. During training, their goal is to find hyperplanes that maximize the margin between positive and negative examples. The task is treated as a quadratic optimization problem and the resulting hyperplanes are used to classify new examples. SVMs are conceptually not related to the methods described above. Nevertheless, they have recently been shown to be highly effective for the task of text classification achieving the highest classification scores in standard text collections. While classification time is linear in the number of features, a major issue on the application of SVMs on large datasets is that their running time is quadratic in the number of training examples. In our experimental section, we study the behavior of SVMs both in terms of accuracy and in terms of scalability using the *SVMlight* package [41], an efficient implementation that has been used in several text classification studies.

4.4 Experimental evaluation

The goal of this section is to study the performance of the methods above along two directions:

I. Quality of classification measured in terms of recall and precision.

II. Scalability in the number of features and the number of documents.

We used the Reuters-21578 Distribution 1.0 that is available from [6]. This data set consists of 21,578 stories from the 1987 Reuters newswire, each one pre-assigned to one or more of a list of 135 topics. We used the '*Mod Apte*' training-test split that contains 9,603 training and 3,299 test examples and considered the 93 categories that are assigned to at least one example in the training and testing data. All words were converted to lower case, punctuation marks were removed, numbers were ignored and words from a common list of stop-words were removed. Finally, a feature selection step was applied. For each class C we created a local dictionary consisting of the K words w_t with the highest average mutual information with C:

$$MI(C;W_t) = H(C) - H(C \mid W_t) = \sum_{c \in C} \sum_{w_t \in \{0,1\}} P(c, w_t) \log \frac{P(c, w_t)}{P(c) \cdot P(w_t)}$$

The classification task requires that a document may be assigned in none, one or more than one categories. We followed standard practice and treated the problem as a series of binary classification problems where the task is to decide whether the document belongs to a specific class or not. This procedure is repeated for all classes and the set of "on" classes is assigned to the document.

We evaluated performance using the standard Information Retrieval measures *recall* and *precision* defined as:

$$recall = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive examples}}$$

$$precision = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive predictions}}$$

To combine recall and precision with a single-value metric that can be used to derive a total order on the classifiers we use the F_1 measure, defined as follows:

$$F_1(recall, precision) = \frac{2 \cdot recall \cdot precision}{recall + precision}$$

The algorithms were optimized to yield maximum F_1 scores using a validation test consisting of the last 2,456 training stories. Algorithms were trained on the first 7,147 documents, optimized on the 2,456 validation documents and finally tested on the 3,299 test stories.

An interesting peculiarity of the data that influences the interpretation of the results is that the category distribution is extremely skewed. There are 93 categories in total, but as Figure 19 shows, only a few categories appear relatively frequently in the training set. Since the performance of many classifiers appears to vary significantly with the number of positive examples in a class [84], we obtained separate results for the 10 largest classes (appearing more than 150 times each in the data) and for all the classes together.

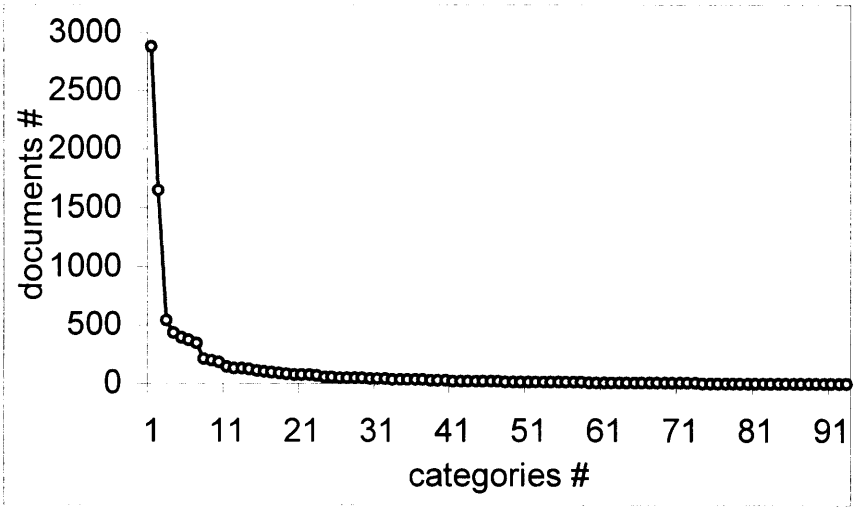


Figure 19: Category frequency distribution in Reuters-21578.

4.4.1 Classification quality

Classification accuracy results are presented in Table 1, which presents the recall, precision and F1 measures of the five algorithms on all classes for vocabulary sizes of 20, 50, 100, 200, 300 and 400 words. In order to summarize the recall, precision and F_1 measures over a set of different classes we used *micro-averaging*, that is cell-wise summing up of the confusion matrices over all the classes and then applying the measures on the resulting confusion matrix. The feature set size has different influence on each algorithm so the feature set size used for each result is also reported.

The effect of the feature size on the classification accuracy is shown in Figure 20 which plots the value of the F_1 score as a function of the feature set size.

Table 11 and Figure 20 confirm results from other studies showing that SVMs are the most accurate methods for text classification. SVMs achieve higher performance both in the 10 most frequent classes and for the 93 classes in total. However, the difference from the other methods is not as large as in other studies. NB sets the base level for comparison and is easily beaten by all other methods, however LB+ and LB+- offer consistently higher scores than TAN with LB+- being very close to SVMs.

	10 most frequent categories				**All 93 categories**			
	m-recall	**m-precision**	**m-F_1**	**|V|**	**m-recall**	**m-precision**	**m-F_1**	**|V|**
NB	0.840	0.813	0.826	100	0.712	0.798	0.752	20
TAN	0.841	0.876	0.858	200	0.765	0.815	0.789	20
LB+	0.850	0.882	0.866	100	0.760	0.82	0.790	50
LB+-	0.877	0.871	0.874	300	0.783	0.814	0.798	50
SVM	0.861	0.916	0.888	300	0.767	0.904	0.830	100

Table 11: Best micro-averaged performance of classifiers and the corresponding vocabulary size on the 10 largest categories (left) and on all 93 (right).

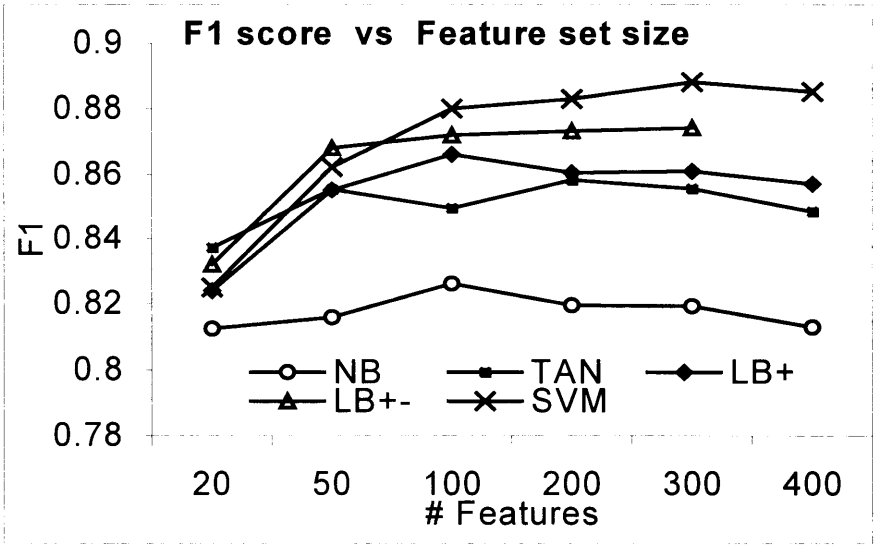


Figure 20: F_1 scores for NB, TAN, LB+, LB+- and SVMs as a function of the vocabulary size.

It is interesting to note that the information available to LB+ is less than that available for the other methods as it only uses information about the presence of words in a document. This is more apparent by a comparison between LB+ and LB+- where the only difference is that LB+- also considers information about the absence of words. Since the two methods differ only in the input representation, we conclude that information about absence of words may also have predictive value. This is, however, at a very expensive price in terms of computational effort needed to handle it as shown in the next section.

We attribute the improvement of both LB versions over TAN to a) the use of local models that capture context specific properties of the data and b) the capture of dependencies among more than two words (dependencies of up to five words were used by LB in the experiments above). The benefits are so great that even LB+ outperforms TAN, although it uses a poorer representation model.

We also measured the relative contribution of factors a) and b) above to the performance gains. We choose LB+- and restricted it to discover itemsets with two items only thus eliminating the influence of factor b). The F_1 score fell but remained above that of TAN therefore we believe that the strength of LB comes mainly from its context-sensitive properties.

4.4.2 Scalability

In Figure 21 we show the training time needed by SVM and LB+ (on a 400MHz PentiumII WinNT PC) when applied in databases of up to nine times the size of the original one to learn the most common class ("earn"). The high accuracy of SVMs does not come for free. Their main drawback is that they require training time quadratic in the number of examples. In contrast, all other methods scale up linear with the data set size. This is an inherent property of SVMs that is caused by their attempt to treat the learning as a quadratic optimization problem and hinders their application to big datasets.

Scalability over the feature set size has remained a major problem for most data mining/machine learning applications. Figure 22 shows the time to train and test the algorithms for the class "earn" as the feature set varies from 20 to 400 words. SVMs and NB scale up linearly with the number of features in both the learning and testing phase. TAN's training time increases quadraticaly with the feature size, but its testing time is linear in the number of features. The graphs also show the significant differences in the performance of LB for the two different representation models.

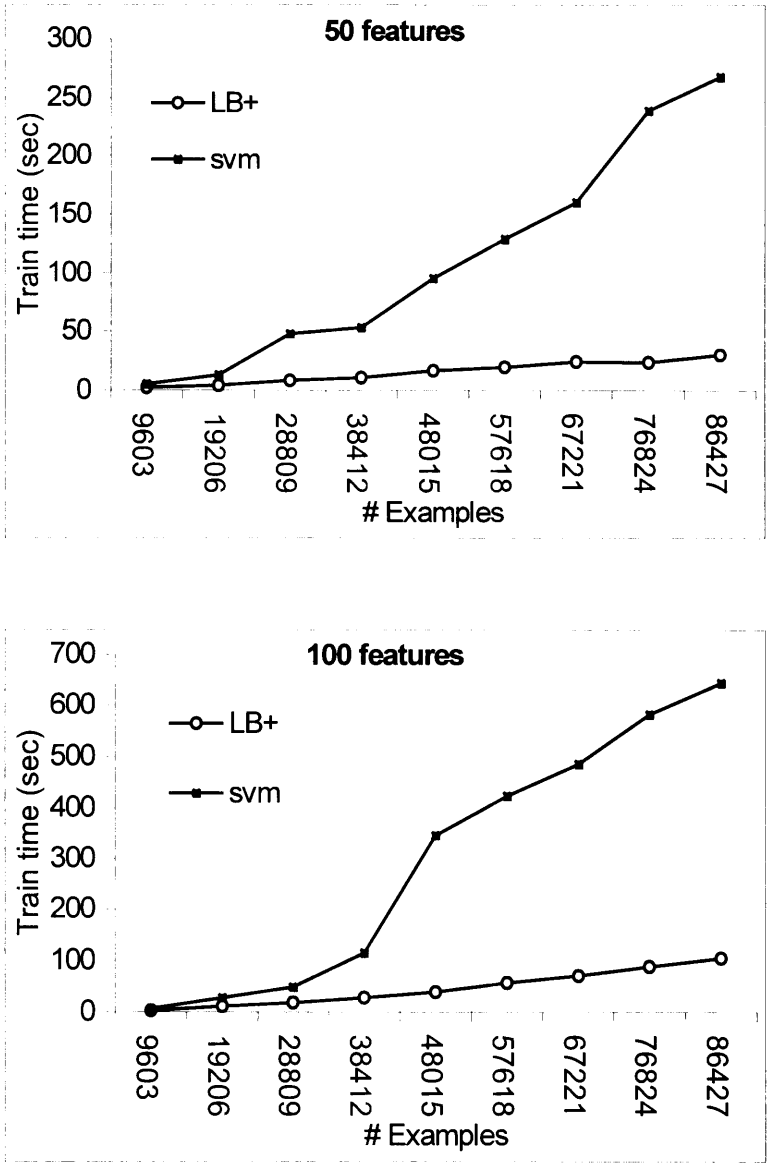


Figure 21: Training time of LB+ and SVM for the biggest class ("earn") and for multiples of the original dataset when the feature set size is 50 (upper) and 100 (lower) features.

In principle, the training time of LB using an Apriori-based itemset discovery engine is exponential in the number of features. Therefore, when both presence and absence of words is considered the combinatorial explosion prevents LB+- from using more than a few hundred features. However, when only word presence is taken into

account the input space is very sparse since most documents contain only a few words from the vocabulary. With these conditions, LB+ scales up very well to many hundreds of features. This is particularly important in comparison with TAN because LB+ outperforms TAN in terms of accuracy and also scales up much better when the dimensionality increases.

An interesting finding of our experiments is that LB particularly benefits from stemming. In our datasets we did not use stemming and this caused the discovery of many unnecessary itemsets. For example, the words "japan" and "japanese" are among the most informative words for class trade. The probability of co-occurrence of these two words in a document labeled as "trade" is very high as they are not independent. Therefore LB will consider the itemset {japan, japanese} as interesting. Also if a word x is associated with "japan" to form itemset {"japan",x} the duplicate itemset {"japanese", x"} will most likely also be identified. This causes unnecessary increase in the size and the complexity of the model without apparent improvement in the predictive power and could have been avoided if stemming was used.

LB is also the only algorithm in this study that does not scale up linearly during the testing phase. This is the price to be paid because LB builds a different model for each classification query. Observations for LB are similar as for the training phase with LB+- scaling up poorly whereas LB+ can scale up reasonably well. We believe that the classification speed of LB can be speeded up if special data structures for storing the itemsets are used or if some caching method is applied. This is the focus of our ongoing work.

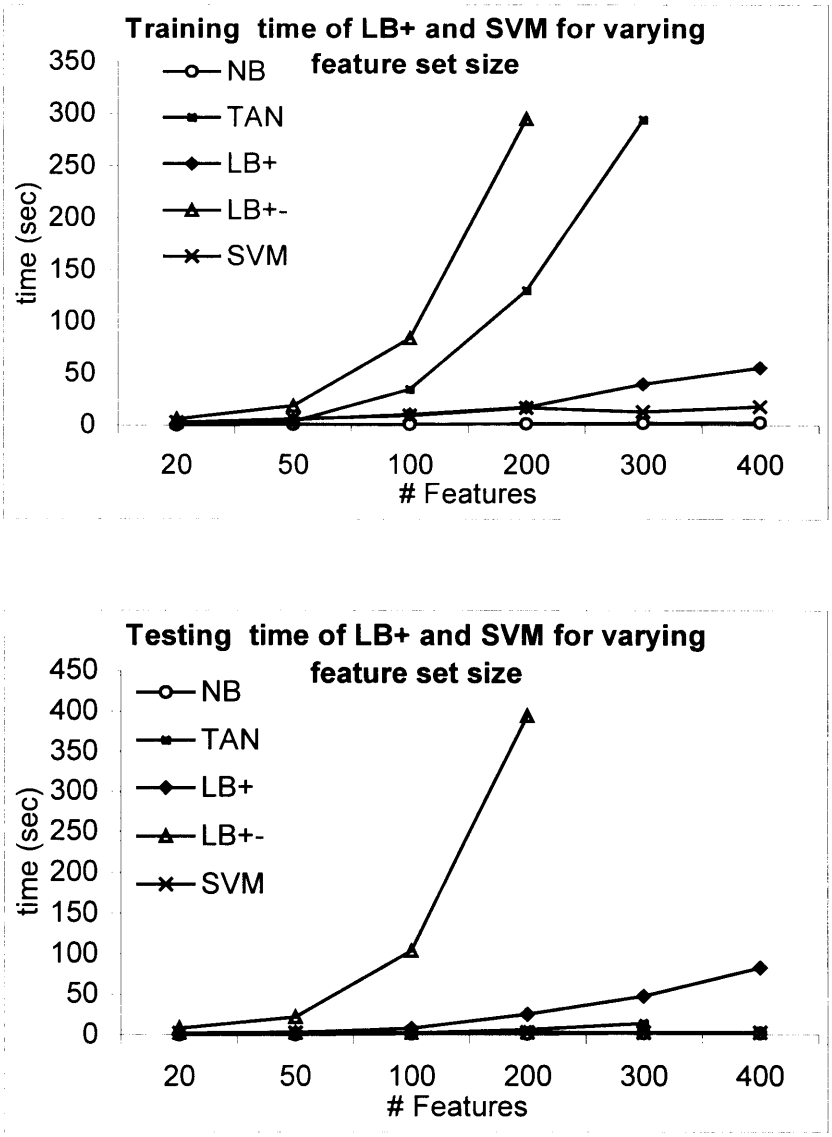


Figure 22: Train (upper graph) and test (lower graph) time for class "earn" for constant database size (9,603 examples) and varying feature size from 20 to 400 features.

4.5 CONCLUSIONS

There are considerable tradeoffs in choosing an appropriate classification method for a given task. As already shown in previous studies, SVM is definitely the most effective in terms of classification performance. Its inability to handle large data sets, however, requires the development of alternative methods that can handle the large document collections encountered in many real problems. In this chapter, we considered Bayesian extensions of NB as alternatives.

Global methods, such as Bayesian Networks, provide good results, but scale up very poorly as the number of features increases. On the other hand, LB, which follows a lazy probabilistic approach, is a promising alternative. LB+-, which uses a dense data representation capturing information about the presence and absence of words, provides very competitive results, but does not scale up to large feature spaces. Its alternative LB+ sacrifices the use of word absence information without loosing much quality to gain significant improvement in terms of speed and scalability. We conclude that LB+, using this sparse data representation, provides a good alternative that achieves high accuracy, scales up linear in the number of documents and handles large vocabulary sizes.

Several research directions remain unexplored. We are interested in applying more complex document representation models for Bayesian approaches. Accuracy of NB is shown to substantially improve when the multinomial model is used. One interesting topic is to investigate if extensions of NB will peoduce similar gains. The higher performance of LB+- compared to LB+ raises the question whether the use of word absence information can be used selectively to yield accuracy gains without significantly decreasing the performance. Results like those reported in [37] are promising and a successful integration with LB would translate to faster running times and better scalability when the dimensionality is increased.

CHAPTER 5. INTEGRATING FEATURE AND INSTANCE SELECTION

Text databases have certain properties, which separate them from traditional relational databases as far as the application of data mining algorithms is concerned. The research area of Information Retrieval has been considered significantly different from the area of Machine Learning (and of Data Mining in recent years) and it has been treated as a separate research field. The nature of text databases is certainly a key factor that contributed to this result. One of the most important differences between text and relational databases has been the number of features, which for text databases can easily reach to the tens of thousands [9][50] compared to orders of magnitude smaller feature set sizes in relational databases. This raises big hurdles in porting machine learning algorithms to the information retrieval arena because such algorithms have been traditionally designed having in mind scalability in the number of instances rather than in the number of features. Feature selection, therefore, has been a major research area within IR since the reduction of the features used for the representation of documents is an absolute requirement for the use of any but the simplest machine learning algorithms [9].

Feature selection methods reduce the dimensionality of datasets by removing features that are considered irrelevant for the classification. This transformation procedure has been shown to present a number of advantages such as:

- Smaller dataset size

- Less computational requirements for the classification algorithms (especially those that do not scale well with the feature set size)
- Considerable shrinking of the search space for the classification algorithm
- Improved classification accuracy

While reducing the number of features has been a very active research topic, historical reasons have shifted the focus of attention of the IR research community away from the number of instances. The main reason is that the amount of unstructured information available in electronic form has been very small until very recently. This has, however, changed recently with the advent of the Internet, the increase in the number of email users, the World Wide Web and very recently the interest of big corporations in Knowledge management solutions to speed up the exchange of information among employees. Collections of thousands, even millions of documents are common nowadays; there are several billion pages freely available on the web and big corporations see an exploding number of electronic documents produced and archived. This increases the demand for more efficient management of text information. Various strategies exist to cope with the increase in the dataset size. They all fall under the umbrella of *instance selection* [9].

As Figure 23 illustrates, instance selection works orthogonal to feature selection. The aim here is the reduction of the number of instances presented to the classifier during the training phase. The motives are similar to the ones considered in feature selection and include smaller dataset size, less computational requirements for the classification algorithms (especially those that do not scale well with the dataset size) and improvement of the input quality by removing noise introduced by inconsistent examples and by examples that do not provide information useful for the classification.

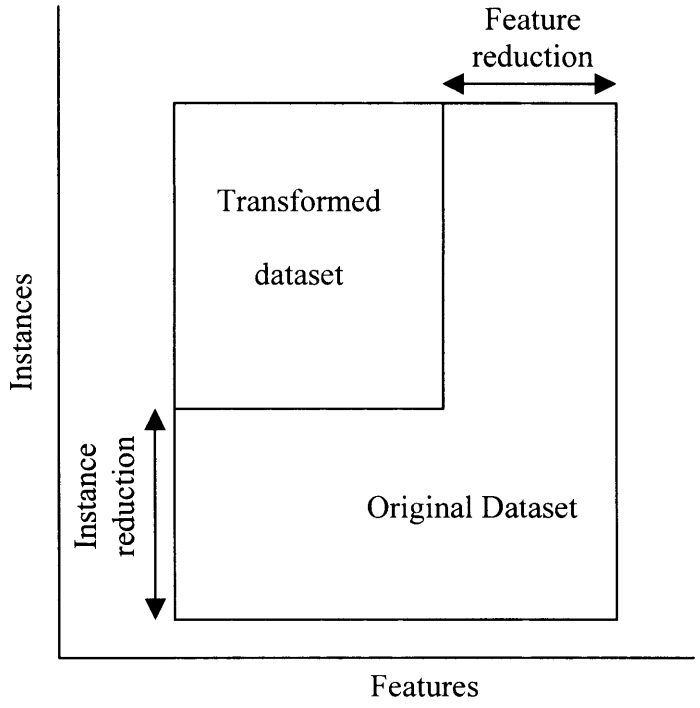


Figure 23: Feature and instance selection as orthogonal tasks.

To the best of our knowledge, feature selection and instance selection have so far been considered only independently of each other. In this chapter, we present our innovative approach [31] that deals with both problems simultaneously. Our algorithm, which we call *FIS* (**F**eature and **I**nstance **S**election) targets the feature and instance selection problem in the context of text classification. At a very abstract level, FIS works in two steps. In the first step, it sequentially selects features that have high precision in predicting the target class. All documents that do not contain at least one such feature are dropped from the training set. In the second step, FIS searches within this subset of the initial dataset for a set of features that tend to predict the complement of the target class and these features are also selected. The sum of the features selected during these two steps is the new feature set and the documents selected from the first step comprise the training set.

Our experimental results with standard benchmark datasets show that FIS achieves a considerable reduction in both the number of features and the number of instances. Equally important, the accuracy of a range of classifiers including Naïve Bayes, TAN and LB is considerably increased when the resulting training sets are used instead of the original ones. In some cases, when the transformed datasets are used for the training, these classifiers prove to be more accurate than the Support Vector Machines, which is currently considered one of the best text classification methods.

5.1 Problem setting, motivation and algorithm outline

Assume a document collection $D = \{d_1, d_2, \ldots d_{|D|}\}$, where each document d_k, k *in* $\{1, 2, \ldots, |D|\}$, contains one or more words from a vocabulary $W = \{W_1, W_2, \ldots W_{|W|}\}$. Each word W_i is associated with a binary variable w_i where $w_i = 1$, if W_i is present one or more times in d_k and $w_i = 0$, if W_i is not present in d_k. In addition, each document d_k is associated with a class label c, which indicates whether d_k belongs to the target class C ($c=1$) or not ($c=0$) in which case we will say that it belongs to class C'. For the sake of simplicity, we ignore the number of occurrences of a particular word W_i in a document d_k as well as their relative position in the document. Instead, we use the so-called "Bag of words" [57] document representation. Additionally, we consider only the binary class problem; multi-class problems have to be split into a sequence of binary class problems. Although in some cases this is not very convenient, it is the case in a variety of problems where a document may belong to more than one possible class.

Our research is based on the hypothesis that neither all documents nor all words are useful for classification. This hypothesis is both intuitive and is supported by experimental evidence (see for example [4][5][9][27][41][50][59][75][84][85][83]). As the vocabulary is typically very large, we expect that only a subset F of features influence the class likelihood of a document. All other features can be safely ignored without affecting the classification.

Similarly, the number of documents available in typical text classification problems may be very large (i.e., the set of pages on the WWW). Only a small subset T of documents, however, bears information that can be used to distinguish between the two classes.

Our Feature and Instance Selection method aims at the simultaneous discovery of the set of relevant features F and the set of relevant documents T. Since both of them are a small percentage of the total set of features and documents our method can achieve considerable reduction in the dataset size. Below we outline the main ideas of the algorithm.

One of the two goals of *FIS* is to select a small number of relevant features. Our generic definition of relevance is that "A feature is relevant if it tends to predict either that a document belongs to C or that it belongs to C' ". Not all relevant features are needed, however, since our objective is the accurate classification of documents based on the features and not the discovery of the relevant features per se. FIS first discovers the subset F_P that contains the *positive features* of W. i.e., the features that indicate the positive class C. We claim that positive features are the words with the highest precision, i.e., those that tend to co-occur with C–labeled documents whereas they are absent in documents belonging to C'.

With a proper selection procedure, F_P will contain enough positive features to ensure that most documents of class C will contain at least one feature from F_P. While most C-labeled documents contain at least one feature from F_P, only a small subset D_{F_P} of documents have non-empty projections on F_P (i.e., contain words from F_P). This is because:

a) Documents usually contain a small portion of the total vocabulary W and

b) Most of the documents that do not belong to C also do not contain features from F_P (note that F_P contains only high-precision features).

The previous argument allows us to connect the feature selection and the instance selection task: If a document d does not contain any words from F_P then it most likely does not belong to class C. Therefore, we can drop all such documents from the training set and achieve a considerable reduction in the number of instances. During classification, if a document's projection of F_P is empty it is automatically assigned to C'; otherwise it is fed to the classifier to estimate its class.

Note that while we can ensure that most documents of C will fall in D_{F_P}, the reverse is not true. This means that D_{F_P} will also contain many documents from C'. To address this problem we search *within* the documents of D_{F_P} for the set F_N of words that predict class C' with high precision. We call these words *negative features*. Note that the accuracy of these features in predicting C' is measured only within the much smaller set D_{F_P} rather than within the original set D.

The process above produces the subset $F = F_P \cup F_N$ of W which is the result of the feature selection and the subset T of D which is the result of the instance selection. The set T contains the documents of D_{F_P} represented only with features from F. In other words, we can say that T is the projection of D_{F_P} on F.

5.2 Related work and supporting evidence

Feature selection for Information Retrieval and more specifically for text classification has received much attention over a long time span and the issue has been addressed from the various perspectives of different research communities including linguistics, natural language processing, information retrieval, machine learning and more recently data mining. This is easily credited to the fact that text collections often have feature set sizes (also called vocabularies) that can reach up to tens or even hundreds of thousands depending on the text representation.

On the other hand, Instance Selection has not enjoyed much attention and we are not aware of any work addressing this issue in the context of text classification. We can name two reasons for this. First, the size of the text collections available for classification has been small up to very recently. Second, in many cases even if the number of documents available is large, the number of labeled documents is small and labeling can be expensive since it is a human-based process. Nevertheless, during recent years we see an expansion in the size of text collections that render unrealistic the assumption that the training set is small.

A very good survey on feature and instance selection as two independent problems in the context of machine learning is presented in [9]. In the context of information retrieval and text classification, several works have indicated that effective feature selection can enhance the performance of classifiers: [27] reports good results in the well known Reuters-21578 collection using 50 to 300 features out of several tens of thousands. Similar results are reported in a very interesting comparison of event models for classification presented in [57] for the Reuters and some additional datasets. The focus was the Naïve Bayes classifier and there was a comparison of its performance when the number of word occurrences in a document is taken into account (multinomial model) versus when words are considered as binary variables that either are or are not present in a document (multivariate Bernoulli model). The findings suggest that, in the multivariate Bernoulli model, NB performs best with a feature set size of between 100 and 1000 words out of a total vocabulary size that varies from 20,000 to 70,000. The multinomial model achieves higher accuracy than the multivariate Bernoulli model, but also requires a much larger number of features (which can be one order of magnitude more). Another study that reinforces the previous results used the Reuters-21578 as well as the OHSUMED collections and the kNN and Linear Least Squares Fit (LLSF) [85]. Varying the number of words from 2% of the total vocabulary to 100%, [85] reports that the performance of both classifiers would peak for a feature set size that was around 2%-10% of the original. Similar results are reported in [50][75] .

In addition to the classification quality, big feature set sizes, even if they contain useful and non-redundant information, make the application of good classification algorithms infeasible since most algorithms have problems scaling up as the feature set size increases (SVMs are a notable exception, but they do not scale up in the dataset size). Therefore, reducing the feature set size enables the application of more sophisticated algorithms, grasping more complex relationships inherent in the dataset.

Another important point can be transferred from the machine learning literature on feature selection. John et. al in their interesting work on using the induction algorithm itself to evaluate the quality of the feature selection (the wrapper approach) [44] use a three-level definition of feature relevance, which we briefly present here:

- Irrelevant features: These features can be ignored without information loss or degradation in the classifier performance.

- Strongly relevant features: Features that contain useful information such that if removed the classification accuracy will degrade.

- Weakly relevant features: Features that contain information useful for the classification, but their presence is unnecessary given that some other words are present in an instance.

In the design of FIS, we took into account this definition of relevance, which is very intuitive. Some words may contribute to the distinguishing of the class, but they might be redundant, as they tend to co-occur with other words, which are also good predictors. Therefore, we can reduce the redundancies hoping that we will not reduce the amount of information in the dataset.

As mentioned earlier there is not much research on Instance Selection for text classification. The issue is mostly addressed in the Machine Learning community either with the traditional statistical approach of sampling or by more elaborate, but sometimes heuristic, approaches. Most of the work refers to Instance-based or lazy algorithms [1]. This is because such algorithms defer all processing until a classification query arrives. Therefore, a large number of instances has a worse impact on lazy learners such as kNN compared to eager learners such as decision trees and Bayesian networks. In [82] the problem is addressed using a distance measure. In essence instances that are "closer" to each other tend to bear overlapping information, therefore, some of them can be discarded.

We are not aware of any work that combines instance and feature selection. In our work, we assess the quality of the documents in terms of the quality of the features they contain. Given a measure of goodness of the features, a document that does not contain good features is itself not useful as a training example for a classifier. It can even be a bad example adding noise instead of additional information. This is the link between feature selection and document selection.

5.3 Algorithm *FIS* description

In this section we describe in detail our algorithm for feature and instance selection, called *FIS* (**F**eature and **I**nstance **S**election), and state the objectives behind our choices for the various steps of the algorithm. The detailed algorithm as well as implementation details is discussed in the next section.

The FIS algorithm operates in two steps:

As explained in section 5.1, during the first step FIS searches for a subset F_P of the original vocabulary W that contains the words of W that are the best predictors of the given class C. As a convenient side effect of this selection, D is pruned and only the documents with non-empty projection on F_P are kept. The resulting dataset D_{F_P} is the set of documents that contain at least one word from F_P. Our goal is that the set D_{F_P} contains the majority of the C-labeled documents and only a small portion of the documents from C' while at the same time the documents outside D_{F_P} will mostly belong to C'. Taking this argument to the extreme, in the ideal case D_{F_P} would contain *all* documents of class C and *only* these documents. In reality this is not the case and the second step aims at refining the results of the first step by discovering words that are good predictors of C'.

In the second step *FIS* searches within D_{F_P} for the set of negative features F_N. This step is exactly symmetrical to the first one and the resulting set contains the words of W that are the best predictors of class C' *within* D_{F_P}. The result of the algorithm is the feature set $F = F_P \cup F_N$ and the instance set $T = D_{F_P}$ that contains the documents in D_{F_P} represented using the features in F.

In Figure 24 we list FIS0, a procedure that is called twice, the first time to discover F_P from the total set of documents D and the second time to discover F_N from the documents in D_{F_P}.

FIS0(C, W, D, min_score, c_support, F, D_F)
Input:

A target class *C*
A vocabulary *W*
A dataset *D* of class-labelled documents containing words from *W*
User defined thresholds: *min_score, c_support.*

Output:

A subset *F* of *W* that are best predictors of class *C*
A subset D_F of *D* with the documents that have non-empty projection on *F*

Algorithm:

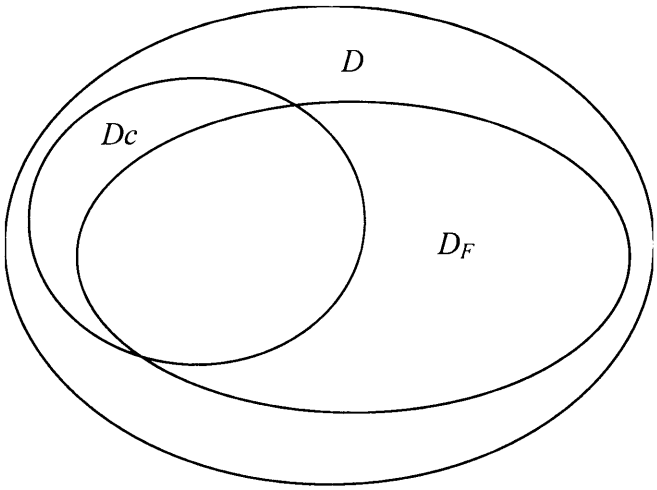
1. $D_F = \{\}$, $F = \{\}$, $D_C^{'} = D - D_C$
2. *repeat{*
3. *for each feature* f_j *in* $W - F$ *calculate the following{*
4. D_{f_j} = {All documents that contain f_j}
5. $F_1 = (D_{f_j} \cap D_C) - D_F$
6. $F_2 = (D_{f_j} \cap D_C^{'}) - D_F$
7. $S_{f_j} = score(f_j) = \dfrac{|F_1|}{|F_2| + \frac{1}{2}}$
8. }
9. *if there* exists *a feature f* that meets the following two criteria:
10. a. $S_f >$ min_*score*
11. b. $S_f \geq S_{f_k}$ for each other feature f_k in $W - F$
12. c. $|D_{f_j}| \geq c\_support \cdot |Dc|$
13. *then{*
14. $F = F \cup \{f\}$
15. $D_F = D_F \cup F_1 \cup F_2$
16. *}*
17. }until no new feature is added to *F*

Figure 24: Procedure FIS0

The sequential feature selection procedure employed by FIS0 works as follows. It starts with an empty set of relevant features F and an empty set of relevant documents D_F. At each iteration it selects the best new feature f to be added to the set of relevant features, where "best" is measured with the function *score(f)* in line 7 of Figure 24. This feature is added to F and all documents that contain it are added to D_F if they are not already there. This procedure repeats until some stopping criteria are met.

Figure 25 illustrates the sets involved in the operation of FIS0. At some point in its operation FIS0 will have created a set D_F of selected relevant documents. The aim of FIS0 is to ensure that D_F will contain most, and ideally all, C-labelled documents without containing many (and ideally containing none) C'-labelled documents. In other words, the objective of FIS0 is to maximize $|D_F \cap D_C|$ while keeping $|D_F - D_C|$ as small as possible. Note that in the ideal case the set D_F would contain *all* and *only* the documents that belong to class C, i.e., the feature selection would have built a perfect classifier.

Let us now examine the criteria for the selection of the next feature to be inserted in the set of relevant features F. For this, assume that the feature f is currently evaluated and D_f, shown in Figure 26, is the set of all documents that contain f. The word f will be inserted into F if its score, measured in line 7 of FIS0, is the highest among all other candidate words. Sets F_1 and F_2, also shown in Figure 26, are used to derive a value for the function *score(f)*.

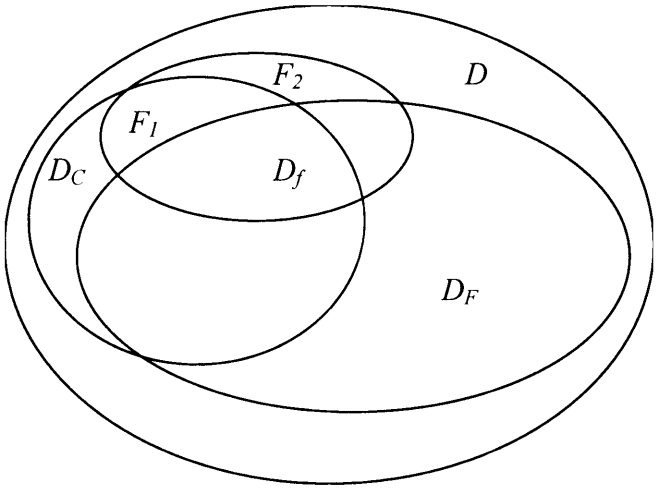


D: The full set of documents

D_C: The documents in *D* that belong to class *C*

D_F: The documents in *D* selected by FIS0

Figure 25: The document sets involved in the operation of FIS0



D: The full set of documents

D_C: The documents in *D* that belong to class *C*

D_F: The documents in *D* selected by FIS0

Df: Documents containing word *f*.

D'_C: The documents in *D* that do not belong to class *C*

$F1 = (D_f \cap D_C) - D_F$

$F2 = (D_f \cap D'_C) - D_F$

Figure 26: The document sets involved in evaluating the relevancy of a word *f*

The values of $|F_1|$ and $|F_2|$ used in line 7 of FIS0 represent the increase in the number of positive and negative examples in D_F respectively, if all the documents that contain the word f are added to D_F. This means that the scoring function $score(f)$ is the traditional Laplace-corrected precision of the word f as a predictor of C in the set $D_f - D_F$ of documents that contain word f but do not contain any of the previously selected words.

The set F generated by FIS0 contains words that also exist in negative examples and consequently D_F contains negative examples. As new features are added to F, D_F grows and along with D_F, $|D_F \cap D_C|$ and $|D_F - D_C|$ grow as well. As the *score* of the new features inserted into F in later iterations decreases, the $|D_F \cap D_C|$ growth rate decreases while the $|D_F - D_C|$ growth rate increases. This means that each newly added feature adds fewer positive and more negative documents in D_F than its predecessors do. In the extreme case, if all features are added to F D_F will contain all documents in D. The final size of D_F, however, as well as the proportions of $|D_F \cap D_C|$ and $|D_F - D_C|$, are controlled by the two stopping criteria in lines 10 and 12 of FIS0:

1. $S_f >$ min_score, where *min_score* is a user define constant.
2. $|D_f| \geq c\_support \cdot |Dc|$, where *c_support* is also a user defined constant.

For example a value of *min_score=1* means that FIS0 will stop if all candidate words introduce more C'-labeled than C-labeled documents to D_F. We fixed the second constant to a value of *c_support=0.01*. This requires the frequency of f in the dataset to be higher than a minimum threshold of $0.01 \cdot |Dc|$ to prevent overfitting.

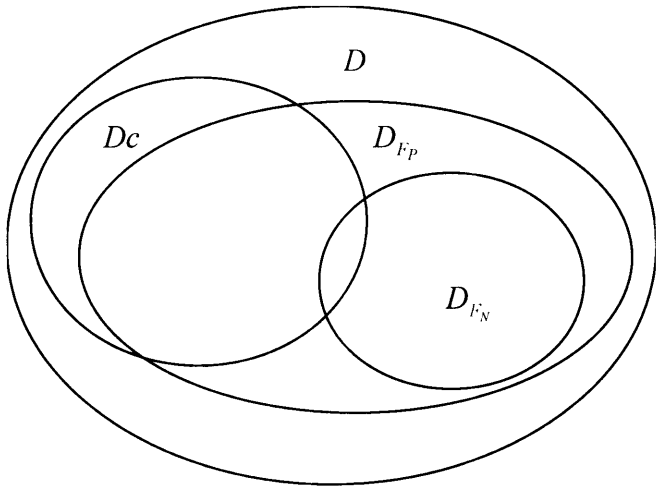
The FIS algorithm described in Figure 27 is a wrapper algorithm that calls FIS0 twice. In the first step, it uses FIS0 to extract F_P and D_{F_P}. F_P is the set of words that are best predictors for class C in D, and D_{F_P} contains all documents from D that have at least one word from F_P. D_{F_P} may be regarded as the union of two sets: $D_{F_P} \cap D_C$ and $D_{F_P} - D_C$. This means that it contains both documents that belong to C and documents that belong to C'. The relative size of $D_{F_P} \cap D_C$ and $D_{F_P} - D_C$ depends on the quality of the words in F_P and the threshold *min_score_pos*. It is obvious that the higher the precision of the words in F_P as predictors of C, the smaller the size of $D_{F_P} - D_C$. In the ideal case, $D_{F_P} \cap D_C = Dc$ and $D_{F_P} - D_C = \{\}$. In the real world, however, $D_{F_P} - D_C$ usually contains many documents that might drive a classification system to make wrong decisions. This is the reason why FIS calls FIS0 again to perform a second step of feature extraction.

FIS(*W, D, min_score_pos, min_score_neg, c_support, F, D_F*)

Input:
A vocabulary *W*
A dataset *D* of class-labelled documents containing words from *W*
User defined thresholds: ***min_score_pos, min_score_neg, c_support***
Output:
$F = F_P \cup F_N$
$D_F = D_{F_P}$
Algorithm:
1. FIS0 (*C, W, D, min_score_pos,* **, c_support,** F_P, D_{F_P}).
2. FIS0 (*C', W,* D_{F_P}, *min_score_neg,* **, c_support,** F_N, D_{F_N}).

Figure 27: Algorithm FIS

In the second step FIS0 looks only within D_{F_P} for a set of words that may accurately discriminate documents that belong to C' from documents that belong to C. This is illustrated in Figure 28.



D: The full set of documents (Training set)

D_C: The documents in D that belong to class C

D_{F_P} : The documents in D with non-empty projection on F_P (i.e., contain at least one feature from F_P)

D_{F_N} : The documents in D_{F_P} with non-empty projection on $F_P \cup F_N$ (i.e., contain at least one feature from $F_P \cup F_N$)

Figure 28: The document sets involved in the operation of FIS0

Figure 28 illustrates the sets generated by the two calls of FIS0. In the first call, FIS0 operates on D and it returns D_{F_P}, which is the set of documents that contain at least one feature from F_P. D_{F_P} contains a substantial part of D_C, but, unfortunately, it also contains many documents outside C. The number $\left|D_{F_P} - D_C\right|$ of documents in D_{F_P} that belong in class C' is, however, very small compared to $|C'|$. Therefore if we restrict our training set to D_{F_P} we expect not to loose much information while greatly reducing the number of training examples.

Reducing the dataset size, however, is not the only aim of FIS. Our experimental results show that if we represent the new dataset with the features of F_P alone and feed them to a classifier, the classifier will not be able to accurately learn class C. To overcome this problem, FIS0 is applied on D_{F_P} and returns F_N, that is the set of words that best predict that a document belongs to C' assuming that it contains at least one word from F_P. The corresponding set of documents is D_{F_N} and it contains documents with at least one word from both F_P and F_N. The set D_{F_N} contains a substantial part of $D_{F_P} - D_C$ and the words in F_N are very accurate in predicting class C'.

As a result of the feature selection, FIS produces the transformed data set D_{F_P} where documents are represented with the features from $F_P \cup F_N$. Typical values for the parameters *min_score_pos* and *min_score_neg* are 0.01 and 1, respectively. In our experiments we show that this transformed dataset leads to superior results by a variety of classification algorithms.

5.4 Experimental evaluation

For our experiments, we used two standard benchmark document collections, the Reuters-21578 and the 20 Newsgroups [6] collection. The classification algorithms we used for the evaluation were NB, LB, TAN and SVM, as described in previous chapters. In the following sections, we briefly present the algorithms used in the experiments. Then, we describe the datasets and the experimental setup. We next provide an overview of the experimental results and finally we drill down discussing the performance of the algorithms in more detail.

5.4.1 Algorithms used

Our experiments evaluated the competitiveness of FIS as a feature and instance selection method. We chose Mutual Information (MI) based feature selection as the alternative for the comparison with FIS. The MI-based feature selection algorithm selects for each class C a local dictionary consisting of the K words w_t with the highest average mutual information with C:

$$MI(C;W_t) = H(C) - H(C \mid W_t) = \sum_{c \in C} \sum_{w_t \in \{0,1\}} P(c, w_t) \log \frac{P(c, w_t)}{P(c) \cdot P(w_t)}$$

The classification task requires that a document may be assigned in none, one or more than one categories.

Feature selection methods influence the performance of classification algorithms in a different way. In our experimental evaluation, we used four algorithms: Naïve Bayes (NB), Tree augmented Naïve Bayes Classifier (TAN), Local Bayes Classifier (LB) and Support Vector Machines (SVM). Section 4.3 introduced these algorithms in the context of text classification.

5.4.2 Dataset description and preprocessing

5.4.2.1 20 Newsgroups dataset

The 20 Newsgroups was collected by Ken Lang and has become one of the standard text collections for evaluation of classification algorithms. It contains about 20,000 newsgroup postings from 20 different UseNet groups. Each document may belong to one or more of the groups. The documents are evenly divided among the classes, i.e., each class contains around 1,000 documents.

We extracted word tokens from the data and removed words that occur in the whole collection (hapax legomena). In addition, all headers from the postings (including the newsgroup header of course) were removed and only the body was used for training or testing. No stemming was used. To reduce the size of the vocabulary we kept only features that occurred in at least five documents. The resulting vocabulary contained 12,357 words.

For each class we created a training set consisting of the first 80% of the documents and a testing set containing the last 20% of the data set. We ran the feature selection and the subsequent classification 20 times, each time considering the binary problem of whether a document belongs to the target class or not.

5.4.2.2 The Reuters-21578 dataset

The Reuters-21578 Distribution 1.0 (available from [6]) consists of 21,578 stories from the 1987 Reuters newswire, each one pre-assigned to one or more of a list of 135 topics. We used the '*Mod Apte*' training-test split that contains 9,603 training and 3,299 test examples. All words were converted to lower case, punctuation marks were removed, numbers were ignored and words from a common list of stop-words were removed. Although Reuters contains 135 classes in total, there are only 93 classes with at least one document in the training and testing set and the frequency of the classes is highly skewed as shown in Figure 19 on page 104. Following a practice popular in the literature, we only used the 10 most popular classes for our experiments.

The classification task requires that a document may be assigned in none, one or more than one categories. We followed standard practice and treated the problem as a series of binary classification problems where the task is to decide whether the document belongs to a specific class or not. This procedure is repeated for all classes and the set of "on" classes is assigned to the document.

5.4.3 Performance measures

We evaluated performance using the standard Information Retrieval measures *recall*, *precision* and *accuracy*. For reference they are provided below:

$$recall = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive examples}}$$

$$precision = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive predictions}}$$

$$accuracy = \frac{\#\text{ of correct (positive or negative) predictions}}{\#\text{ of predictions}}$$

In contrast to accuracy, recall and precision provide a partial order of the quality of classification/feature selection methods. To combine recall and precision with a single-value metric that can be used to derive a total order on the classifiers we use the F_1 measure, defined as follows:

$$F_1(recall, precision) = \frac{2 \cdot recall \cdot precision}{recall + precision}$$

The algorithms were optimized to yield maximum F_1 scores using a validation test consisting of the last 25 per cent of the training stories.

5.4.4 Results discussion

The positive influence of FIS in the performance of classification algorithms is summarized in Table 12 and Table 13 that list the F1-measure, accuracy and train/test time of the four classifiers described above in the 20-newsgroups (Table 12) and Reuters (Table 13) datasets. The two tables compare FIS as a feature and instance selection method with Mutual Information (MI) based feature selection. MI is a commonly used method for feature selection and it has been shown to yield very competitive results compared to other feature selection methods [85].

	NB		TAN		LB		SVM	
	MI	FIS	MI	FIS	MI	FIS	MI	FIS
F1	57,5	69,52	62,6	67,98	59,2	68,63	64,99	66,23
Accuracy	95,5	96,98	96,4	96,89	95,5	97,02	97,01	96,89
Train Time	2,5	0,66	226,2	87,75	1,25	2,78	56	10,1
Test Time	1,36	0,36	2,74	0,71	4,9	0,54	1,65	0,62

Table 12: Classification Accuracy and Time for "20-newsgroups"

	NB		TAN		LB		SVM	
	MI	FIS	MI	FIS	MI	FIS	MI	FIS
F1	83,02	89,21	86,75	89,34	86,6	89,29	88,88	89,72
Accuracy	97,14	98,14	97,79	98,21	97,72	98,20	98,16	98,25
Train Time	1,25	0,07	138,7	8,96	126,75	6,3	16,9	4,9
Test Time	0,92	0,1	1,64	0,16	165,9	4,69	1,25	0,8

Table 13: Classification Accuracy and Time for "Reuters 21578"

In both datasets, there is a significant improvement in the classification quality of all four algorithms as measured by the F1 measure when FIS is used as the feature selection method compared to MI-based feature selection. Interestingly, the performance improvement with FIS is higher for the methods that have the lowest performance when MI is used. We believe that this is attributed to the following two reasons:

a. There are fewer complex relationships between the features returned by FIS compared to the features returned by MI, and

b. FIS significantly reduces the number of examples retained, compared to the total dataset size.

NB is known to reach its peak performance under such conditions, whereas classifiers that build more complex models such as TAN, LB and SVM tend to under-perform when the complexity of the dataset is lower.

The simplicity and small size of the datasets after the application of FIS compared to MI is also apparent from the training time of the algorithms. In most cases, there is a significant decrease in the running time, which can reach to an order of magnitude (e.g., for TAN and LB in the Reuters dataset).

In the results of Table 12 and Table 13, we also include the Accuracy next to the F1 measure as a measure of classification quality mainly because several papers use it as a measure in algorithm comparisons involving the 20-Newsgroups dataset. Our results, however, show that accuracy is not as appropriate as the F1 measure even in the 20-Newsgroups dataset because of the small number of documents that belong to each class. In the sequel, we will only refer to F1 as a measure of classification quality.

The effect of FIS on the four classifiers varies considerably. Overall, NB achieves the highest average F_1 score of 79.37 on the two datasets, followed by LB with 78.96 and TAN with 78.66. SVM is last with an F_1 score of 77.98.

In Reuters-21578, NB with FIS is more accurate than SVM with MI, while in 20-Newsgroups NB achieves about 1.3 per cent higher accuracy than the TAN algorithm. Furthermore, training and testing times for NB remain very low, making NB the best choice. It is interesting that SVM and TAN, two methods that build complex classification models, do not benefit from the use of FIS. While their performance improves when FIS is used, overall NB and LB enjoy stronger performance improvements from the use of FIS. We attribute this to the fact that the feature/instance selection step with the use of FIS produces a feature and instance set that do not contain complex feature relationships. Naïve Bayes is known to perform best in such situations, whereas other more complex methods perform better when there are more complexities inherent in the dataset. LB has the ability to adapt its model to the complexities of the dataset and it reduces to NB in the extreme case. This may explain the fact that its performance is close to the performance of NB.

Table 14 provides a comparison of FIS and MI in terms of their direct effects on the dataset rather than on the performance of the algorithms that use the dataset. Both methods reduce the feature set size to such an extent, that only a handful of words are used to represent an average document. The main benefit of FIS, however, is its ability to reduce the dataset size by around a significant 80% in these cases

	20-Newsgroups		Reuters	
	MI	FIS	MI	FIS
Average # of features per document	4,13	5,1	7,31	3,92
Average # of examples per class	11110	2515	8913	1471
Dataset size in KBs	305,53	79,57	390,1	34,3
Feature selection time per class in secs	4,87	7,78	1,76	1,95

Table 14: Effect of FIS and MI on "20 Newsgroups" and "Reuters-21578"

An additional advantage of FIS over MI is that, for MI-based feature selection, a fixed total number of features has to be specified in advance, a task which requires experimentation from the user on a trial and error basis and can have a significant impact on the results obtained. In contrast, FIS automatically determines the number of features needed to optimize both dimensionality/size reduction and increased accuracy and removes the need for an additional user-defined threshold.

A unique characteristic of FIS is that to the best of our knowledge it is the first algorithm that performs simultaneous feature and instance selection. In contrast, MI and other feature selection methods reduce the number of features, but not the number of instances. The joint instance and feature selection by FIS results in both reduced number of examples in the training dataset (2,515 instead of 11,110 in 20-Newsgroups and 1,471 instead of 8,913 in Reuters) and significantly lower storage space for the dataset (four-fold reduction in 20-Newsgroups and more than ten-fold reduction in Reuters). Notably, all these improvements come at a very low cost; the running time of FIS is actually only slightly higher that the time of MI.

Table 15 and Table 16 illustrate the per class performance of FIS and MI for both 20-Newsgroups and Reuters-21578 document collections. It is clear that in the majority of cases the classification accuracy increased significantly, especially in the 20-Newsgroups collection.

F1	NB		TAN		LB		SVM	
class	MI	FIS	MI	FIS	MI	FIS	MI	FIS
1	40,44	65,03	47,80	59,85	49,88	62,75	60,82	55,79
2	49,18	57,51	52,02	57,54	43,9	59,15	42,07	53,17
3	48,43	47,72	49,88	47,92	44,74	47,45	47,06	51,35
4	49,18	55,43	53,55	57,49	51,46	55,86	53,25	54,18
5	55,31	71,96	63,18	70,49	60,51	71,11	66,47	66,67
6	53,92	70,79	60,76	67,77	62,95	69,01	66,87	67,03
7	70,21	69,68	69,12	65,66	63,5	67,53	72,14	65,91
8	62,29	74,75	65,73	70,91	60,71	73,90	63,58	70,50
9	82,71	85,45	84,55	83,41	75	83,73	88,28	85,29
10	67,89	89,06	80,51	88,56	75,18	87,78	82,82	87,27
11	79,32	89,69	85,86	89,69	78	89,69	85,18	90,16
12	70,41	88,89	77,60	87,37	75,86	87,17	84,62	84,91
13	42,65	52,94	46,01	52,57	44,16	49,32	38,85	42,33
14	60,59	72,73	66,67	71,23	55,67	72,68	68,15	65,72
15	70,03	76,57	73,12	74,73	66,81	77,16	73,60	74,61
16	58,99	77,51	64,56	76,70	64,22	76,44	70,24	71,83
17	48,53	64,48	49,63	62,10	53,5	62,68	52,57	60,57
18	76,76	82,12	79,26	75,82	77,69	78,71	78,31	77,26
19	35,40	43,64	38,03	44,57	43,77	38,85	40,42	40,99
20	36,36	48,83	39,11	45,20	42,68	44,81	38,87	44,51

Table 15: Per class classification quality for 20-Newsgroups

F1	NB		TAN		LB		SVM	
class	MI	FIS	MI	FIS	MI	FIS	MI	FIS
Earn	96,90	96,60	96,71	96,75	97,4	97,21	97,66	97,67
Acq	87,33	92,01	90,99	91,93	89,66	91,94	91,47	92,67
money-fx	57,45	73,25	63,70	69,36	68,28	72,77	65,06	75,86
Grain	75,08	92,31	84,21	92,26	83,44	91,56	91,75	90,55
Crude	80,11	83,82	82,89	83,06	82,48	81,84	80,87	82,04
Trade	54,98	65,44	57,73	66,67	61,33	66,37	70,20	66,67
Interest	50,90	70,08	61,03	69,32	62,91	67,91	62,50	68,33
Wheat	69,66	89,61	78,20	88,89	71,19	89,47	84,29	86,45
Ship	81,08	78,82	82,96	79,76	83,16	73,20	77,22	79,04
Corn	52,48	90,32	78	90,16	72,44	90,16	87,72	88,52

Table 16: Per class classification quality for Reuters-21578

Table 17 and Table 18 illustrate the attributes of the datasets that FIS and MI built for the 20-Newsgroups (Table 17) and the Reuters-21578 (Table 18) collections. An interesting observation can be done here, namely that the Reuters-21578 is a much easier dataset to learn than the 20-Newsgroups dataset. This is explained below.

In Reuters-21578 the F_P feature set consists of an average of only 27 features per class. These 27 features are enough to cover 99.6 per cent of the positive training examples while they are present in only 8.5 per cent of the negative training examples. It is really impressive that just 39 features cover 2,869 of 2,877 positive training examples of the "earn" class. FIS also extracted an average of 62 additional features per class for the F_N feature set. Thus, just 99 features led all the four-classification systems to superior classification accuracy.

On the other hand, 20-Newsgroups is a much tougher domain. An average of 118 features per class are contained in the F_P feature set compared to just 27 features in Reuters-21578. This is more than four times larger. The size of F_P, is not the only indication of the complexity of this collection. The 118 features of F_P in Newsgroups cover only 92.6 per cent of the positive training examples (99.6 percent in Reuters) and are present in 12.15 per cent of the negative training examples (8.5 percent in Reuters).

Summarizing, in the 20-Newsgroups collection FIS extracted 4 times more features, which cover 7 per cent less of the positive training examples and 4 per cent more of the negative training examples. Furthermore, FIS extracted 159 more features for the F_N feature set, per class on the average. Therefore, although the accuracy of NB with FIS is 21% higher than the accuracy of NB with MI, still the overall performance is low, compared with the one achieved in the Reuters-21578 collection.

class	FIS							MI	
	|Fp|	|Fn|	|Dc|	|Dc^Dfp|	|Dfp|	|Dc^Dfn|	|Dfn|	|Dc^Df|	|Df|
1	110	180	797	768	2333	9	1254	793	13345
2	141	204	790	715	3486	49	2098	746	12918
3	66	121	787	622	1343	10	578	754	11906
4	123	222	794	759	4320	66	2744	777	13645
5	113	215	790	736	3836	10	2247	756	13268
6	128	163	790	711	2201	4	1154	765	12998
7	98	232	773	753	6440	30	4706	750	14766
8	128	193	788	746	2705	6	1639	743	10195
9	103	113	791	760	1450	1	585	772	9992
10	120	113	789	735	1635	2	843	752	8857
11	82	30	793	725	820	0	86	775	7108
12	94	116	798	752	1543	4	680	774	10716
13	147	213	793	693	3720	7	2385	706	11880
14	146	130	791	716	1718	1	868	730	8255
15	141	157	795	746	1941	5	992	748	6354
16	135	167	796	765	2403	43	1438	783	11219
17	127	157	798	760	2105	7	1084	783	11732
18	93	74	793	736	989	0	230	755	8003
19	135	180	797	715	2352	50	1447	776	12071
20	142	211	795	736	3157	89	2000	775	12969
Average	118	159	792	733	2525	20	1453	761	11110

Table 17: Attributes of the FIS-built 20-Newsgroups dataset

class	FIS							MI	
	|Fp|	|Fn|	|Dc|	|Dc^Dfp|	|Dfp|	|Dc^Dfn|	|Dfn|	|Dc^Df|	|Df|
Earn	39	99	2877	2869	4577	117	1647	2871	9394
Acq	65	108	1650	1636	4096	347	2672	1626	9270
money-fx	36	75	538	536	1470	91	954	535	9145
Grain	19	35	433	433	532	4	94	433	9046
Crude	19	82	389	388	935	68	586	388	7773
Trade	16	83	369	363	1141	50	769	368	9070
Interest	29	82	347	346	1200	28	743	347	9213
Wheat	6	23	212	212	282	4	66	212	8942
Ship	31	18	197	192	255	5	61	197	8305
Corn	5	14	181	181	225	5	46	181	8972
Average	27	62	719	716	1471	72	764	716	8913

Table 18: Attributes of the FIS-built Reuters-21578 dataset

CHAPTER 6. CONCLUSIONS AND RESEARCH DIRECTIONS

The focus of this thesis is on four areas of data mining research. In this chapter, we summarize our results in each of these areas and illustrate possible research directions that we believe can deliver significant contributions.

The first topic of interest for us is the conception of a framework that is able to describe a variety of classification methods. In CHAPTER 2 we explain how we perceive a variety of classification methods to be variations of a more generic model where building classifiers equates to the search for an appropriate set of itemsets and classifying can be done by consulting the discovered itemsets. This different perception has been the cornerstone for the majority of our research with itemset-based Bayesian classification. Its biggest value lies in the fact that it uncovers similarities among classifiers that belong to apparently very different categories. We can now identify underlying similarities between lazy learners such as k-NN, eager Bayesian classifiers such as NB and Decision Tree classifiers such as C4.5.

The benefit of such an approach can be multifold. Having created a new perspective on the building blocks of classification methods prepares the ground for significant cross-fertilization. Combining association mining with Bayesian classification as described in CHAPTER 3 is just one instance. The idea of creating hybrid methods with features from k-NN and Decision Trees sounds ad-hoc without a proper common ground. Under our framework, it might be easier to combine techniques from different domains to build methods that are more efficient.

Our concept of classification as discovery and use of itemsets is by no means a solid methodology. An interesting research direction is to solidify this concept by formalizing the framework and creating the mapping between the framework and known classification methods. Defining the semantics behind this mapping is of particular interest.

Another research direction is to use the framework to create new algorithms that combine characteristics from several existing methods. It is well known, for example, that decision trees perform particularly well in domains where the classification decision relies on the information of a few variables only. Bayesian methods such as NB on the other hand perform better when the result of classification depends on the contribution of a larger number of variables. Adaptive methods that behave more like a Decision Tree in some contexts and more like NB in some others could be produced by combining features both from Decision Trees and from NB classification methods.

A number of hybrid classification methods already exist. NBTrees [46] are hybrids combining Decision Trees and Naïve Bayes classifiers. Our framework can be used to put in context such hybrid methods, compare them with existing methods and discover their advantages and disadvantages.

Itemset based Bayesian classification is our second topic of interest in this thesis. Local Bayes is a classifier built from itemsets discovered using association mining techniques. The unique characteristic of LB is that it builds local Bayesian Networks during the classification.

We can identify several research directions based on our experience with LB. One direction is towards faster learning and classification. Currently LB uses Apriori as the association-mining engine. There are several advances in the field of association mining and it would be interesting to adapt newer and faster association mining methods to increase the speed of LB.

A more fundamental question that still waits to be addressed is the relationship between local and global Bayesian classification. LB builds classification models that are local to each classification query. There is one model built per query though, independent of the number of classes. *Bayesian Multinets* [39] on the other hand build global models with respect to the classification queries, but produce one model per class. It is interesting to investigate an alternative version of LB that builds a local model per classification model *and* per class.

Generalizing the above, it is interesting to find the relationship between LB and global Bayesian classifiers. There are several questions to be answered such as the following: When is a local method better and when a global one? Is it more effective to build one model for all classes or class-specific models? Can we build a method that creates local models when locality is a characteristic of the dataset, but is also able to discover global models when such models exist?

Text classification with its particularities and special requirements is the third topic of interest in this thesis. The main open issues here are the use of a richer model for the representation of documents, when Bayesian classifiers are used. The multinomial model for text representation, that takes into account the frequency of word occurrence has been shown to present certain advantages over the multivariate Bernoulli model that considers bare word presence or absence [57]. While the use of the multinomial model with NB has been tested, it has not been used in combination with more complex Bayesian methods. To our opinion, this is due to the increased complexity introduced by the word dependencies in such methods. It is, however, of interest to evaluate the performance of both models above with more complex Bayesian classification methods such as TAN and LB.

The final piece of research presented in this thesis deals with feature and instance selection for text classification. Our Feature and Instance Selection method, called FIS, selects features that are good in predicting the target class while at the same time prunes documents that clearly do not belong to the target class. While the original reason for pruning documents is to reduce the dataset size, FIS sometimes is so effective in pruning documents that it keeps almost all and almost only the documents that belong to the target class. From a different point of view, *FIS* at its best behaves close to a classifier. It is very interesting to further investigate this link between feature selection, instance selection and classification as well as to define the point where feature selection stops and classification starts.

Other interesting topics related to FIS include the use of the multinomial model for feature and instance selection (currently we use the multivariate Bernoulli), the extension of FIS to handle multiple classes for feature selection, and the use of FIS on non-text, relational datasets.

BIBLIOGRAPHY

[1] D. W. Aha, *Lazy Learning*, (Reprinted from Artificial Intelligence Review 11), Kluwer Academic Publishers, 1997.

[2] R. Agrawal, R. Srikant, "Fast algorithms for mining association rules", *in proceedings of the VLDB-94*, 1994.

[3] K. Ali, S. Manganaris, R. Srikant, "Partial Classification using Association Rules", *in proceedings* of the *3rd International Conference on Knowledge Discovery & Data Mining*, 1997.

[4] C. Apte, F. Damerau, and S. M. Weiss, "Automated learning of Decision Rules for Text Categorization", *ACM TOIS*, Vol.12, No.3, pp.233 – 251, 1994.

[5] L. D.Baker, A. K. McCalum, "Distributional Clustering of Words for Text Classification", *in proceedings of the 21st ACM SIGIR (SIGIR'98)*, 1998.

[6] S. D. Bay, *The UCI KDD Archive* [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science, 1999.

[7] R. J. Bayardo, "Brute-Force Mining of High Confidence Classification ", *in proceedings of the Third International Conference on Knowledge Discovery & Data Mining*, 123-126, 1997.

[8] R. J. Bayardo, "Efficiently Mining Long Patterns from Databases", *in proceedings of ACM SIGMOD-98*, 1998

[9] A. L. Blum and P. Langley. "Selection of relevant features and examples in machine learning", *Artificial Intelligence, 97:245--271*, 1997

[10] R.R. Bouckaert, "Properties of Bayesian Network Learning Algorithms", in Proc. of the *10-th Conf. on Uncertainty in AI* (UAI-94), de Mantarnas R.L., Poole D. (eds.) Morgan Kaufmann, San Franciso 1994, pp. 102-109.

[11] C. Boutilier, N. Friedman, M. Goldszmidt, D. .Koller, "Context - Specific Independence in Bayesian Networks", in p*roceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1996.

[12] L. A. Breslow, D. Aha, "Simplifying Decision Trees: A Survey", *NCARAI Technical Report No. AIC-96-014*, 1996.

[13] L. Breiman, J. Friedman, R. Olshen, C. Stone, *Classification and Regression Trees.* Wadsworth, 1984.

[14] C. Brodley and P. Utgoff, "Multivariate Decision Trees", *Machine Learning*, 19, 1995.

[15] W. Buntine, "A guide to the literature on learning probabilistic networks from data", *IEEE Transactions on Knowledge and Data Engineering*, 8:195--210, 1996.

[16] B. Cestnik, "Estimating Probabilities: A crucial task in machine learning", *in proceedings of ECAI-90, 1990.*

[17] C. K. Chow, C. N. Liu, "Approximating discrete probability distributions with dependence trees", *IEEE Trans. on Information Theory,14*, 462-467, 1968.

[18] P. Clark, R. Niblett, "The CN2 induction algorithm", *Machine Learning*, 3, 261-284, 1989.

[19] W. Cohen, "Learning with set-valued features", *in proceedings* of the *13th National Conference on Artificial Intelligence (AAAI-96)*, 1996.

[20] W. Cohen and Y. Singer, "Context-sensitive learning methods for text categorization", *ACM TOIS, Vol.17, No.2, pp.141 – 173*, 1999.

[21] S. Cost and S. Salzberg, "A weighted nearest neighbor algorithm for learning with symbolic features", *Machine Learning, 10*, 57-78, 1993.

[22] B. Dasarathy (Ed.), *Nearest Neighbor (NN) Norms: NN pattern classification techniques*, IEEE Computer Society Press, 1991.

[23] P. Dawid, "Conditional Independence in Statistical Theory", in *Journal of Royal Statistical Society, Ser.B, vol.41, no.1, pp.1-31*, 1979.

[24] P. Domingos, "Context-Sensitive Feature Selection for Lazy Learners", *Artificial Intelligence Review*, 11, 227-253

[25] P. Domingos, M. Pazzani, "On the optimality of the Simple Bayesian Classifier under Zero-One Loss", *Machine Learning*, 29, 103-130, 1997.

[26] R. Duda, P. Hart, *Pattern Classification and Scene Analysis,* New York: John Wiley & Sons, 1973.

[27] S. Dumais, J. Platt, D. Heckerman and M. Sahami, "Inductive Learning Algorithms and Representations for Text Categorization", *in proceedings of the 7th ACM CIKM Conference (CIKM98)*, 1998.

[28] U. M. Fayyad, K. B. Irani, "Multi-Interval discretization of continuous-valued attributes for classification learning", *in proceedings* of the *13th International Joint Conference on Artificial Intelligence (IJCAI)*, 1022-1027, 1993.

[29] U. Fayyad, G. Piatetsky-Shapiro & P. Smyth, "From Data Mining to knowledge discovery", in U. Fayyad, G. Piatetsky-Shapiro & P. Smyth, eds, *From Data Mining to Knowledge Discovery*, 1996.

[30] P. Flach and N. Lavrac, "The role of feature construction in inductive rule learning", *in proceedings of the* ICML-2000 Workshop on Attribute-Value and Relational Learning, *June 2000*

[31] D. Fragoudis, D. Meretakis and S. Likothanassis, "Integrating Feature and Instance selection for Text Classification", *Submitted for publication*, 2001.

[32] N. Friedman, D. Geiger, M. Goldszmidt, "Bayesian Network Classifiers", *Machine Learning*, 29, 131-163, 1997.

[33] N. Friedman and M. Goldszmidt, "Learning Bayesian Networks with Local Structure", *in proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI' 96)*, 1996.

[34] J. Friedman, R. Kohavi, Y. Yun, "Lazy Decision Trees", in *Proc. of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[35] D. Geiger, D. Heckerman, "Knowledge representation and inference in similarity networks and Bayesian multinets", *Artificial Intelligence*, 82, 45-74, 1996.

[36] G. Graefe, U. Fayyad, S. Chaudhuri, "On the Efficient Gathering of Sufficient Statistics for Classification from Large SQL Databases", *in proceedings of the Fourth International Conference on Knowledge Discovery & Data Mining*, 1998.

[37] J. Han, J. Pei and Y. Yin, "Mining Frequent Patterns without Candidate Generation", *in proceedings of the ACM SIGMOD'2000*, 2000.

[38] D. Heckerman, "A tutorial on Learning with Bayesian Networks", *Technical Report MSR-TR-95-06*, Microsoft Research, Advanced Technology Division, 1996.

[39] D. Heckerman, *Probabilistic Similarity Networks*, Cambridge, MA: MIT Press, 1991.

[40] J. Hipp, U. Güntzer and G. Nakhaeizadeh, "Algorithms for Association Rule Mining- A general Survey and Comparison", *SIGKDD Explorations, vol.2, issue 1*, pp.58-64, ACM, 2000.

[41] T. Joachims, "Text Categorization with Support Vector Machines: Learning with many Relevant Features", *in proceedings of the ECML'98*, 1998.

[42] T. Joachims, "Making Large Scale SVM-Learning Practical", in *Advances in Kernel Methods Support Vector Learning,* MIT Press, 1999.

[43] G. John, *Enhancements to the data mining process*, PhD Dissertation, Stanford University, USA, 1997.

[44] G. John, R. Kohavi and K. Pfleger, "Irrelevant Features and the Subset Selection Problem", *in proceedings of the 11th International Conference on Machine Learning (ICML-94)*, San Francisco, USA, pp 121-129, 1997.

[45] S. Kasif, S. Salzberg, D. Waltz, J. Rachlin, D. Aha, "A probabilistic framework for memory-based reasoning", *Artificial Intelligence, 104*, 287-311, 1998.

[46] R. Kohavi, "Scaling up the accuracy of naïve-Bayes classifiers: A decision-tree hybrid", *in proceedings of the second International Conference on Knowledge Discovery & Data Mining*, 202-207, 1996.

[47] R. Kohavi, G. John, R. Long, D. Manley, K. Pfleger, "MLC++: a machine learning library in C++", *Tools with Artificial Intelligence*, 740-743, 1994.

[48] I. Kononenko, "Semi-Naïve Bayesian Classifier", *in proceedings of the 6th European Working Session on Learning*, 206-219, 1991.

[49] S. Kullback,R. A. Leibler, "On information and sufficiency", *Annals of Mathematical Statistics,* 22, 76-86, 1951.

[50] D. D. Lewis, "Feature Selection and Feature Extraction for Text Categorization", *Speech and Natural Language: in proceedings of a workshop held at Harriman, NY,* Morgan Kaufmann, San Mateo, CA, pp. 212-217, 1992.

[51] D. D. Lewis, "Naïve (Bayes) at Forty: The Independence Assumption in Information Retrieval", *in proceedings of the ECML-98*, 1998.

[52] P. M. Lewis, "Approximating Probability Distributions to Reduce Storage Requirements", *Information and Control*, 2:214-225, 1959.

[53] B. Liu, W. Hsu, Y. Ma, "Integrating Classification and Association Rule Mining", *in proceedings of the Fourth International Conference on Knowledge Discovery & Data Mining* (KDD-98), 1998.

[54] P. Langley, S. Sage, "Induction of selective Bayesian classifiers", *in proceedings of the 10th Conf. On Uncertainty in Artificial Intelligence, 399-406*, 1994.

[55] H. Mannila, H. Toivonen, "Multiple uses of frequent sets and condensed representations", *in proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.

[56] B. Masand, G. Linoff, and D. Waltz, "Classifying news stories using memory based reasoning", *in proceedings of the 15th ACM SIGIR (SIGIR'92)*, pp. 59-64, 1992.

[57] A. McCallum and K. Nigam. "A Comparison of Event Models for Naive Bayes Text Classification", *in proceedings of the AAAI/ICML-98 Workshop on Learning for Text Categorization*, pp. 41-48, AAAI Press, 1998.

[58] M. Mehta, R. Agrawal, J. Rissanen, "SLIQ: A fast Scalable Classifier for Data Mining", *Proc. of the 22nd Int'l Conference on Very Large Databases*, 1996.

[59] D. Meretakis, D. Fragoudis, H. Lu and S. Likothanassis, "Scalable Association Based Text Classification", *in proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM 2000)*, McLean, VA, USA, 2000.

[60] D. Meretakis, H. Lu and B. Wuthrich, "A study on the performance of Large Bayes Classifier", *in proceedings of the European Conference on Machine Learning (ECML 2000)*, Barcelona, Spain, 2000.

[61] D. Meretakis, B. Wüthrich, "Extending Naïve Bayes Classifiers Using Long Itemsets", i*n proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99), pp 165-174*, San Diego, CA, USA, 1999.

[62] D. Meretakis, B. Wüthrich, "Classification as Mining and Use of Labeled Itemsets", *in proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'99)*, Philadelphia, USA, 1999.

[63] C. J. Merz, P. Murphy, UCI repository of machine learning databases, 1996 (http:://www.cs.uci.edu/~mlearn/MLRepository.html)

[64] T. M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.

[65] R. Motwani, E. Cohen, M. Datar, S. Fujiware, A. Gionis, P. Indyk, J. D. Ulman and C. Yang, "Finding interesting associations without support pruning", *in proceedings of the 16th International Conference on Data Engineering (ICDE), IEEE, 2000.*

[66] T. H. Ng, W. B. Goh, and K.L. Low, "Feature selection, perceptron learning and a usability case study for text categorization", *in proceedings of the 20th ACM SIGIR Conference (SIGIR 97)*, pp. 67-73, 1997.

[67] R. Ng, L. Lakshmanan, J. Han, A. Pang. "Exploratory Mining and Pruning Optimizations of Constrained Association Rules", *In Proc. ACM SIGMOD Conference on Management of Data*, pp. 13-24, 1998.

[68] T. Niblett, "Constructing Decision trees in noisy domains", *in proceedings of the Second European Working Session on Learning, (pp 67-78),* 1987.

[69] J. S. Park, M. S. Chen, P.S. Yu, "An effective hash-based algorithm for mining association rules", In *Proc. ACM SIGMOD Conference on Management of Data*, pp. 175-186, 1995.

[70] J. Pearl, *Probabilistic Reasoning in Intelligent* Systems, Networks of Plausible Inference, San Mateo, CA: Morgan Kaufmann, 1991.

[71] J. Pearl, "Bayesian Networks", in *MIT Encyclopedia of the Cognitive Sciences*, 1997.

[72] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B.P. Flannery, *Numerical Recipes in C*, second Ed, Cambridge University Press, 1992.

[73] J. R. Quinlan, "Simplifying decision trees", *Intl. J. Man-Machine Studies,* 27,221-234, 1987.

[74] J. R. Quinlan, *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann, 1993.

[75] E. Riloff, "Little words can make a big difference for Text Classification", *in proceedings of the 18th ACM SIGIR Conference (SIGIR 95)*, pp. 130-136, 1995.

[76] M. Sahami, "Learning limited dependence Bayesian Classifiers", *in proceedings of the Second Internnational Conference on Knowledge Discovery and Data Mining*, 1996.

[77] A. Savasere, E. Omiecinski, S. Navathe, "An efficient algorithm for mining association rules in large databases", i*n proeedings of the 21st Intl'l Conf. on Very Large Data Bases (VLDB95),* 1995.

[78] R. Srikant, Q. Vu, R. Agrawal, "Mining Association Rules with Item Constraints", *Proc. of the 3rd Int'l Conference on Knowledge Discovery and Data Mining (KDD-97)*, 1997.

[79] P. D. Turney, "The Identification of Context-Sensitive Features: A formal Definition of Context for Concept Learning", *in proceedings of the ICML-96 workshop on Learning in Context-Sensitive Domains*, 1996.

[80] D. H. Wolpert, " The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework", In D. H. Wolpert (ed.), *The Mathematics of Generalization*, 1994.

[81] G. Webb and M. Pazzani, "Adjusted probability naïve Bayesian induction", *in proceedings of the Tenth Australian Joint Conference on Artificial Intelligence*, 1998.

[82] D. R. Wilson and T. R. Martinez, "Instance Pruning Techniques", *in proceedings of the Fourteenth International Conference on Machine Learning (ICML-97)*, San Francisco, USA, pp 404-411, 1997.

[83] Y. Yang and C. G. Chute, "An example-based mapping method for text categorization and retrieval", *ACM TOIS*, Vol. 12, No 3, pp 252-277, 1994.

[84] Y. Yang and X. Liu, "An re-examination of text categorization", *in proceedings* of the *22nd ACM SIGIR Conference (SIGIR' 99)*, 1999.

[85] Y. Yang and J. O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization", *in proceedings of the 14th International Conference on Machine Learning*, pages 412-420. Morgan Kaufmann, 1997.

[86] N. L. Zhang, "Irrelevance and parameter learning in Bayesian networks", *Artificial Intelligence*, 88, pp. 359-373, Elsevier Science B.V., 1996

[87] N. L. Zhang, D. Poole, "On the Role of Context-Specific Independence in Probabilistic Inference", *in proceedings* of the *19th International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.

[88] Z. Zheng, G. Webb, "Lazy Bayesian Rules", *Technical Report (TRC98/17),* Deaking University, Australia, *1998.*