Extending Naïve Bayes Classifiers Using Long Itemsets

Dimitris Meretakis
Computer Science Department
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
meretaks@cs.ust.hk

Beat Wüthrich
Computer Science Department
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
beat@cs.ust.hk

ABSTRACT

Itemsets provide local descriptions of the data. This work proposes to use itemsets as basic means for classification purposes too. To enable this, the concept of class support sup_i of an itemset is introduced, i.e., how many times an itemset occurs when a specific class c_i is present. Class supports of frequent itemsets are computed in the training phase. Upon arrival of a new case to be classified, some of the generated itemsets are selected and their class supports sup_i are used to compute the probability that the case belongs to class c_i. The result is the class c_i with highest such probability. We show that selecting and combining many and long itemsets providing new evidence (interesting) is an effective strategy for computing the class probabilities. The proposed classification technique is called Large Bayes as it happens to reduce to Naïve Bayes classifier when all itemsets selected are of size one only. Experimental results on a large number of benchmark data sets show that Large Bayes consistently outperforms the widely used Naïve Bayes classifier. In many cases, Large 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).

Keywords

Classification, Association Mining, Bayesian learning, Lazy Learning.

1. Introduction

Condensing data to make it more manageable and comprehensible has been recognized as a key topic in data mining [GFC98,MT96]. One popular way to condense data is to employ any of the successful itemset generation algorithms [AS94]. An itemset with its support provides information on the frequency of a certain pattern and can be thought of as a local descriptor of the data. Since training a classifier can be thought of as finding a condensed model of the data [W94], the question arises whether itemsets can also be used for classification purposes. This paper provides a positive answer by showing: (i) how itemsets can be used to construct a *partially-lazy classifier* [A97], and (ii) that the resulting classifier leads to superior accuracy in many cases. In the training phase, an Apriori-like frequent pattern mining algorithm is employed to discover frequent itemsets of arbitrary size together with their class supports. The class support is a variation of the usual support notion and estimates the probability that the pattern occurs under a certain class label. Upon arrival of a new case to be classified, a local classification model is built on the fly focussing on the context of this particular case. This approach is in between lazy learning (no work done during training) and eager learning (full construction of a global model while training).

The actual classification model for a new case with attribute values $A=\{a_1,a_2,...,a_n\}$ consists of a formula computing the conditional probability $P(c_i|A)$, i.e. the probability that the case belongs to class c_i given the evidence. The result is the class with highest such conditional probability. As discussed in [L59], the probability $P(c_i|A)$ can be estimated using different product approximations. Each product approximation implies different independence assumptions about the attributes. For example, $P(a_1a_2a_3c_i)P(a_4a_5|a_1c_i)$ and $P(a_1a_2a_3c_i)P(a_5|a_2c_i)P(a_4|a_1a_5c_i)$ are both product approximations of $P(a_1a_2a_3a_4a_5c_i)$. We show that selecting a certain product approximation is equivalent to selecting certain itemsets generated while training. The selection strategy for the itemsets 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. Consequently, the proposed classifier is called *Large Bayes* (*LB*). This name seems justified as in the extreme case, when the selected itemsets are all of size one, Large Bayes collapses to Naïve Bayes.

Already Naive Bayes (NB) is a surprisingly successful classification method that has outperformed much more complicated methods in many application domains [FGG97,DP97]. NB assigns new cases, represented as a vector of attribute-values $A=\{a_1,a_2,...,a_n\}$, to the class c_i with the highest conditional probability $P(c_i|A)$. From the definition of conditional probability follows $P(c_i|A)=P(A,c_i)/P(A)$. 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 $P(A,c_i)=P(A|c_i)P(c_i)$. To compute $P(A,c_i)$[1], NB makes the assumption that all attributes are conditionally independent given the class c_i, hence:

KDD-99 San Diego CA USA

[1] We use the notation $P(a_1...a_nc_i)$ and $P(A,c_i)$ interchangeably.

Eq. 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.

Our approach fundamentally differs from existing classification methods in the following aspects:

- LB fits in between the eager and lazy learning paradigms [A97]. Eager methods, like Decision Trees [Q93] and Bayesian Networks [P91], 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 a query-specific local model using the evidence provided by the itemsets.
- We shed light on the relationship between descriptive (association) and classification mining. A well-known and successful method, 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. Such an 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.
- In contrast to probabilistic inference methods, LB bypasses the conditional independence testing methodology as investigated in Bayesian Network literature [P91]. Most real data sets contain only a small fraction of the possible attribute-value pairs. 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.

The rest of the paper is structured as follows. Section 2 reviews related work and motivates our approach. Section 3 provides an overview and the intuition behind LB. Section 4 discusses the algorithm in detail while pinpointing various fine-tuning opportunities that can lead to even further improved classification accuracy. Section 5 presents an extensive experimental evaluation of LB on popular benchmark datasets and compares its performance with C4.5, the standard decision-tree classifier, Naïve Bayes, a restricted Bayesian network extension called TAN, and CBA, a recently proposed rule based classification method based on association rule mining. Section 6 draws conclusions and highlights future work.

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. *Tree Augmented Naive Bayesian classifier (TAN)* [FGG97] extends NB taking into account additional dependencies among non-class attributes. TAN employs a modified version of a method proposed by [CL68] to learn a restricted tree-structured Bayesian network [P91] considering only the most important correlations between pairs of attributes. Several other methods follow the same direction. *KDB* [S96] constructs a Bayesian classifier where each attribute may depend on at most k other attributes, where k is a given parameter. *Selective Bayesian Classifier* [LS94] preprocesses data using a form of feature subset selection to delete strongly inter-related (and therefore redundant) attributes. *Semi-Naïve Bayesian Classifier* [K91] iteratively joins pairs of attributes to relax the strongest independence assumptions. Following a different approach, *Adjusted Probability NB* [WP98] infers a weight for each class, which is applied to derive an adjusted probability estimation used for the classification. Finally, *NBTree* [K96] 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, [B+96] introduces the notion of *context-specific independence,* which describes independence relations among variables that hold in certain contexts only. Similarly, *Bayesian Multinets* [H91] model relationships among variables within each class separately. Large 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 [LHM98] 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 [Q93] in many cases. Earlier work on using association rules for classification includes [B97] that discusses the problem of enhancing an association rule miner with additional pruning strategies to extract high-confidence (>90%)

classification rules from data sets. [AMS97] 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. Classification Using Large Itemsets

LB approximates long marginals (support of itemsets with many items) using itemsets of arbitrary length as long as they are frequent and, therefore, reliable. All attributes are assumed to be discrete. Continuous attributes are discretized into intervals using standard discretization algorithms [FI93]. 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* [AS94] $\{a_1,...,a_n\}$ 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.

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 $l.sup_l$ 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 [AS94] to extract from D a set F of frequent and *interesting* itemsets. An itemset A is interesting in our context, if $P(A,c_i)$ cannot be accurately approximated by its direct subsets, i.e. subsets of A with one item missing. A quantitative measure of interestingness is presented in Section 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 $A=\{a_1,...,a_n\}$ is done by combining the evidence provided by the subsets of A that are present in F (see procedure *classify*, Section 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 A*. The border consists of the longest possible itemsets of F that are subsets of 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 $A=\{a_1,...,a_5\}$ arrives. In the learning phase the set of all interesting and frequent itemsets F has been computed. Figure 1 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 $\{a_1,...,a_5\}$. These itemsets, marked in bold in Figure 1, can be used to compute the desired estimates. Note that NB uses all supports in the first level of Figure 1, i.e. all single attributes, and TAN employs some itemsets of the second level, i.e. pairs of attributes.

LB uses the itemsets of B to derive a *product approximation* of $P(A,c_i)$ for all classes c_i. [L59] describes the concept of product approximation in the context of approximation of higher order

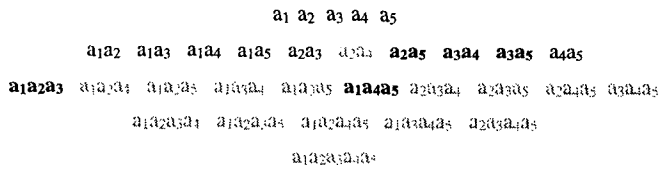

Figure 1: The border of F with respect to $\{a_1,...,a_5\}$

probability distributions by their lower order components. We only consider the approximation of individual elements of a probability distribution. Following [L59], the product approximation of the probability of an n-itemset A for a class c_i contains a sequence of at most n subsets of A such that each itemset contains at least one item not contained (or *covered*) in the previous itemsets. To derive the approximated value of $P(A,c_i)$ 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 $\{a_1,...,a_5\}$ using the border itemsets shown in Figure 1:

i. $\{a_1a_2a_3\},\{a_1a_4a_5\} \Rightarrow P(a_1a_2a_3c_i)\,P(a_4a_5|a_1c_i)$

ii. $\{a_1a_2a_3\},\{a_2a_5\},\{a_1a_4a_5\} \Rightarrow P(a_1a_2a_3c_i)P(a_5|a_2c_i)\,P(a_4|a_1a_5c_i)$

iii. $\{a_1a_2a_3\},\{a_2a_5\},\{a_3a_4\} \Rightarrow P(a_1a_2a_3c_i)\,P(a_5|a_2c_i)\,P(a_4|a_3c_i)$

iv. $\{a_2a_5\},\{a_3a_5\},\{a_1a_2a_3\},\{a_1a_4a_5\} \Rightarrow$
$P(a_2a_5c_i)P(a_3|a_5c_i)\,P(a_1|a_2a_3\,c_i)\,P(a_4|a_1a_5c_i)$

Note that $\{a_1a_2a_3\},\{a_1a_4a_5\},\{a_2a_5\}$ (which contains the same itemsets as ii. but in a different order) is not a product approximation since all items of $\{a_2a_5\}$ 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 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). 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.

#	Covered items	Itemset selected	Product approximation	Available itemsets
0	$\emptyset$	$\emptyset$	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 2: Incremental construction of a product approximation for $P(a_1...a_5c_i)$.

Figure 2 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 $\{a_2a_5\},\{a_3a_4\}$ and $\{a_3a_5\}$ and all three have the same size. Therefore, the most interesting one is selected. Assume this is $\{a_2a_5\}$. It is inserted in the solution and items a_2 and a_5 are covered now. Among the remaining itemsets, $\{a_3a_5\}$ 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 $\{a_3a_4\}$ and $\{a_1a_2a_3\}$ both introduce one non-covered item (a_4 and a_1 respectively). Since $\{a_1a_2a_3\}$ is the longest it is preferred, as is $\{a_1a_4a_5\}$ preferred over $\{a_3a_4\}$ in the fourth step. Itemset $\{a_3a_4\}$ 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 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 $P(a_3|a_2a_5c_i)=P(a_3|a_5c_i)$. This is much weaker than assuming that $P(v_3|v_2v_5c_i)=P(v_3|v_5c_i)$ for all possible instantiations of v_2, v_3 and v_5; a global model would make this stronger assumption.

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

4. Algorithm Large-Bayes (LB)

Section 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 4.2 shows how these itemsets and the class supports are used to classify new cases, while Section 4.3 studies statistical fine-tunings.

4.1 Learning or Discovering Interesting Itemsets

Algorithm *genItemsets* (Figure 3) generates the interesting and frequent itemsets using a bottom-up approach based on Apriori [AS94]. 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.  F1 = {{ai} | ai is non class attribute}
2.  determine l.counti for all l ∈ F1 and all classes i
3.  for (k=2; Fk-1 ≠ ∅; k++) {
4.      Ck = genCandidates(Fk-1)
5.      for all tuples t ∈ D {
6.          Ct = subsets(Ck, t);
7.          i = class of t;
8.          for all candidates l ∈ Ct {
9.              l.counti++;
10.         }
11.     }
12.     Fk = selectF(Ck)
13. }
14. return F = ∪k Fk and l.counti for all l ∈ F and all I
```

Figure 3: 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, itemsets which are interesting (see below) and frequent are selected by procedure *selectF* (line 12).

Procedure *genCandidates* is an extension of the candidate generation procedure suggested in Apriori [AS94]. 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 $P(l,c_i)$:

Eq. 2. $$P_{j,k}(l,c_i) = \frac{P(l_j,c_i)\times P(l_k,c_i)}{P(l_j \cap l_k, c_i)}$$

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

Eq. 3. $$I(l\,|\,l_j,l_k) = \sum_{c_i}\left|P(l,c_i)\log\frac{P(l,c_i)}{P_{j,k}(l,c_i)}\right|.$$

$I(l\,|\,l_j,l_k)$ becomes zero if $P_{j,k}(l,c_i) = P(l,c_i)$; its value increases with the difference between the estimated an the actual probability. The absolute value in Eq. 3 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:

Eq. 4. $$I(l) = \frac{2}{|l|\cdot(|l|-1)}\cdot\sum_{\substack{j,k\in\{1,\ldots,|l|\}\\ j<k}} I(l\,|\,l_j,l_k)$$

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* $I_{P\text{-}P'}$ [L59], a measure used in probability theory to estimate the distance between two probability distributions P and P':

Eq. 5. $$I_{P\sim P'} = \sum_{\mathbf{x}} P(\mathbf{x})\log\frac{P(\mathbf{x})}{P'(\mathbf{x})}.$$

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

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 4. 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 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 A; B is the set of all subsets of A of maximal cardinality that belong to F. Procedure *pickNext* (Figure 5) then repeatedly inserts itemsets from the border in the solution marking their items as covered. The algorithm stops once all items in A have been covered.

classify(F,A)
Input: The set F of discovered itemsets a new instance A
Output: the classification c_i of A

1. $cov = \varnothing$ \\ the subset of A already covered
2. $nom = \varnothing$ \\ set of itemsets in nominator
3. $den = \varnothing$ \\ set of itemsets in denominator
4. $B = \{l\in F \mid l \subseteq A$ and $\neg\exists l'\in F: l' \subseteq A$ 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. $\quad den = den \cup \{l_k \cap \mathrm{cov}\}$
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 4: 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:

Eq. 6 $$P(l_k - \mathrm{cov}\,|\,l_k \cap \mathrm{cov}, c_i) = \frac{P(l_k,c_i)}{P(l_k \cap \mathrm{cov}, c_i)}$$

In line 7 and 8 the nominator and denominator itemsets of Eq. 6, l_k and $l_k \cap \mathrm{cov}$ 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 $P(A,c_i)$ for each class c_i. The most likely

pickNext(cov, B)
$T = \{\ l\in B: |l - covered| \geq 1\}$;
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 5: Procedure *pickNext*.

class is then returned.

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. In the rare case where more than one itemsets qualify one is randomly chosen.

4.3 Zero Counts and Smoothing

The set of conditional probabilities used in line 11 of *classify* can be unreliable for small data sets. A standard statistical technique is to incorporate a small-sample correction into the observed probabilities [N87]. This is called *smoothing* in [FGG97] and helps eliminating 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 ($\mathbf{x}$ and $\mathbf{y}$ 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:

Eq. 7. $$P^s(\mathbf{x} \mid \mathbf{y}, c_i) = \frac{|D| \cdot P(\mathbf{x},\mathbf{y},c_i) + n_0 \cdot P(\mathbf{x})}{|D| \cdot P(\mathbf{y},c_i) + n_0}$$

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 examples support the observed conditional probability $P(\mathbf{x}|\mathbf{y},c_i)$, the closer the actual conditional probability is to $P(\mathbf{x})$.

In the special case where both $P(\mathbf{x},\mathbf{y},c_i)$ and $P(\mathbf{y},c_i)$ are zero, Eq.7 yields $P^s(\mathbf{x} \mid \mathbf{y}, c_i) = P(\mathbf{x})$. In all our experiments n_0 is set to 5.

5. Experimental Evaluation and Discussion

We use 21 data sets from UCI ML Repository [MM96] to evaluate the performance and the behavior of Large Bayes (LB) algorithm. We compare LB with Naïve Bayes [DH73] and three other state-of-the-art classifiers: the widely known and used decision tree classifier C4.5 [Q93]; TAN, a state of the art extension of NB shown to outperform many Bayesian Network approaches and also NB [FGG97]; and CBA, a recently proposed classifier similar to ours in the sense that it employs a frequent pattern miner to discover association rules [LHM98].

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 [FI93] as implemented in the MLC++ system [K+94]. No discretization was applied for C4.5.

Table 1 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 1. Firstly, it is clear that TAN and LB are in general ahead of 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 are still producing 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

	Data set Properties					Accuracy				
Data Set	**# Attributes**	**# Classes**	**Missing Values**	**# Train**	**# Test**	**NB**	**C4.5**	**TAN**	**CBA**	**LB τ_t =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 1. Description of data sets and summary of results.

		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 2. Accuracy of LB for various values of τ_I; values which outperform all other methods in Table 1 are in bold.

values, its accuracy for data sets containing missing attribute values can not 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.

Varying the minimum support threshold helps somewhat pruning the number of itemsets generated. But 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 far 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

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

generated and the classification accuracy.

Tables 2 and 3 provide more insights into the behavior of LB. Table 2 shows the accuracy and Table 3 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 3 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.

Figure 6 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 most 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.

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 7 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 7 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 [DP97] 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. In contrast to NB, LB can take advantage of the largest samples by

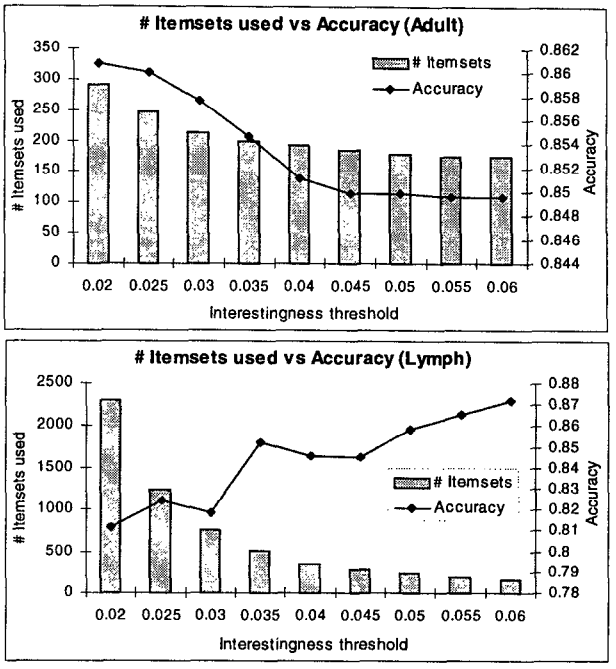

Figure 6. Accuracy and the number of itemsets used by LB for the Adult and Lymph data sets as a function of τ_I.

adjusting τ_I and allowing more complex models to be built. In order to directly compare LB with the other methods (see Table 1) we simply fixed τ_I to 0.04, the middle of the range of values we used in our experiments so as to cope 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 2 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.

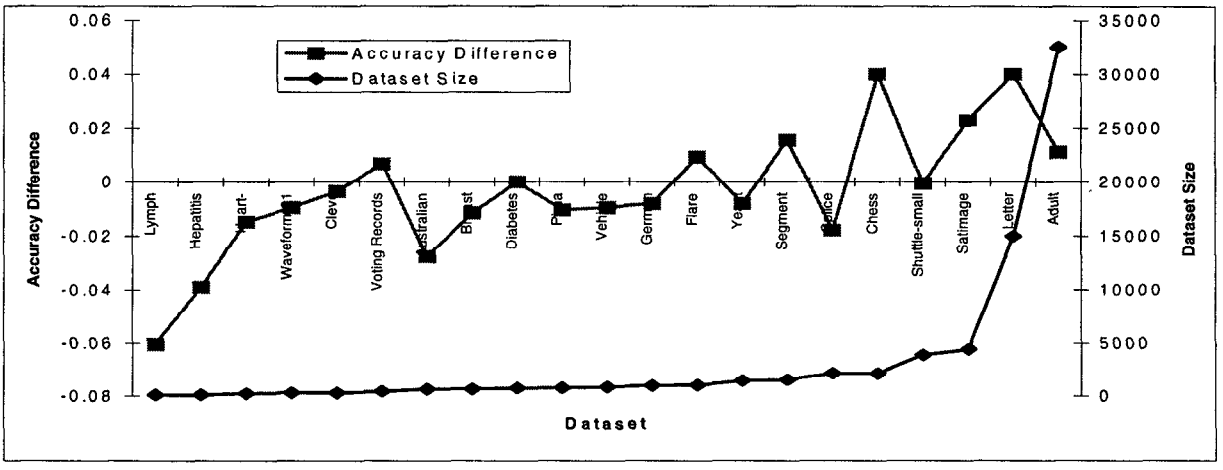

Figure 7. Accuracy change when decreasing τ_I from 0.06 to 0.02 with *minsup* set to 1%. Domains are sorted in increasing data set size. Positive change indicates accuracy improvement when τ_I decreases.

The classification time of LB mainly depends on the number of discovered itemsets. As shown in Table 3, for most data sets a fairly small number of itemsets is generated and is sufficient to produce accurate classifications. Table 4 shows the CPU time in seconds required for training and testing of LB with τ_l=0.02 and τ_l=0.04 respectively.

		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 4. CPU time in seconds for training and testing LB.

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 4 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 [B97] and classification speed can be further improved by indexing the set F of itemsets.

6. Conclusions

A new classification algorithm, called Large 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. In the extreme case where all the discovered itemsets are of size one only, LB reduces to Naïve Bayes. Large 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.

7. ACKNOWLEDGMENTS

The authors were partially supported by Sino Software Research Institute Project No SSRI97/98.EG03 and HKUST DAG 98/99.EG06. We would like to thank Bing Liu and Yiming Ma for providing the CBA algorithm and helping us during the experiments.

8. REFERENCES

[A97] D.W.Aha, *Lazy Learning*, (Reprinted from Artificial Intelligence Review 11), Kluwer Academic Publishers, 1997.

[AS94] R. Agrawal, R. Srikant, "Fast algorithms for mining association rules", *Proc. VLDB-94*, 1994.

[AMS97] K.Ali, S. Manganaris, R. Srikant, "Partial Classification using Association Rules", *Proc. 3rd Int'l Conf. on Knowledge Discovery & Data Mining*, 1997.

[B97] R.J.Bayardo, "Brute-Force Mining of High Confidence Classification Rules", *Proc. 3rd Int'l Conf. on Knowledge Discovery & Data Mining*, 123-126, 1997.

[B+96] C. Boutilier, N.Friedman,M.Goldszmidt,D.Koller, "Context-Specific Independence in Bayesian Networks, *In Proc. Twelfth Conf. on Uncertainty in Artificial Intelligence (UAI)*. 1996.

[CL68] C.K.Chow, C.N.Liu, "Approximating discrete probability distributions with dependence trees", *IEEE trans. on Information Theory,14*, 462-467, 1968.

[DP97] P.Domingos, M.Pazzani, "On the optimality of the Simple Bayesian Classifier under Zero-One Loss", *Machine Learning*, 29, 103-130, 1997.

[DH73] R.Duda, P.Hart, *Pattern Classification and Scene Analysis*, New York: John Wiley & Sons, 1973.

[FGG97] N.Friedman, D. Geiger, M. Goldszmidt, "Bayesian Network Classifiers", *Machine Learning*, 29, 131-163, 1997.

[FI93] U.M.Fayyad, K.B.Irani, "Multi-Interval discretization of continuous-valued attributes for classification learning", *Proc. of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, 1022-1027, 1993.

[GFC98] G.Graefe, U.Fayyad, S.Chaudhuri, "On the Efficient Gathering of Sufficient Statistics for Classification from Large SQL Databases", *Proc. 4th Int'l Conf. on Knowledge Discovery & Data Mining*, 1998.

[H91] D.Heckerman, *Probabilistic Similarity Networks*, Cambridge, MA:MIT Press, 1991.

[K96] R.Kohavi, "Scaling up the accuracy of naïve-Bayes classifiers: A decision-tree hybrid", *Proc. 2nd Int'l Conf. on Knowledge Discovery & Data Mining*, 202-207, 1996.

[K+94] R.Kohavi, G.John, R.Long, D.Manley, K.Pfleger, "MLC++: a machine learning library in C++", *Tools with Artificial Intelligence*, 740-743, 1994.

[K91] I.Kononenko, "Semi-Naïve Bayesian Classifier", *Proc. 6th European Working Session on Learning*, 206-219, 1991.

[L59] P.M. Lewis, "Approximating Probability Distributions to Reduce Storage Requirements", *Information and Control*, 2:214-225, 1959.

[LHM98] B.Liu, W.Hsu, Y.Ma, "Integrating Classification and Association Rule Mining", *Proc. 4th Int'l Conf. on Knowledge Discovery & Data Mining*, 1998.

[LS94] P.Langley, S.Sage, "Induction of selective Bayesian classifiers", *Proc. 10th Conf. On Uncertainty in Artificial Intelligence*, 399-406, 1994.

[MM96] C.J.Merz, P.Murphy, UCI repository of machine learning databases, 1996 (http://www.cs.uci.edu/~mlearn/MLRepository.html)

[MT96] H.Mannila, H.Toivonen, "Multiple uses of frequent sets and condensed representations", *Proc. 2nd Int'l Conf. on Knowledge Discovery & Data Mining*, 1996.

[N87] T.Niblett, "Constructing Decision trees in noisy domains", *Proc. of the Second European Working Session on Learning*, (pp 67-78), 1987.

[P91] J.Pearl, *Probabilistic Reasoning in Intelligent Systems, Networks of Plausible Inference*, San Mateo, CA: Morgan Kaufmann, 1991.

[Q93] J.R.Quinlan, *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann, 1993.

[S96] M.Sahami, "Learning limited dependence Bayesian Classifiers *Proc. 2nd Int'l Conf. on Knowledge Discovery & Data Mining*, 1996.

[W94] 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.

[WP98] G. Webb & M. Pazzani (1998) "Adjusted probability naïve Bayesian induction", *Proc. Tenth Australian Joint Conference on Artificial Intelligence*,1998.