Pre-publication draft of paper published in Machine Learning, 41, 53–87 (2000)


# Lazy Learning of Bayesian Rules

ZIJIAN ZHENG AND GEOFFREY I. WEBB {zijian,webb}@deakin.edu.au
*School of Computing and Mathematics*
*Deakin University, Geelong, Victoria 3217, Australia*

*Revised April 2000*

**Editor:** Douglas Fisher

**Abstract.** The naive Bayesian classifier provides a simple and effective approach to classifier learning, but its attribute independence assumption is often violated in the real world. A number of approaches have sought to alleviate this problem. A Bayesian tree learning algorithm builds a decision tree, and generates a local naive Bayesian classifier at each leaf. The tests leading to a leaf can alleviate attribute inter-dependencies for the local naive Bayesian classifier. However, Bayesian tree learning still suffers from the small disjunct problem of tree learning. While inferred Bayesian trees demonstrate low average prediction error rates, there is reason to believe that error rates will be higher for those leaves with few training examples. This paper proposes the application of lazy learning techniques to Bayesian tree induction and presents the resulting lazy Bayesian rule learning algorithm, called LBR. This algorithm can be justified by a variant of Bayes theorem which supports a weaker conditional attribute independence assumption than is required by naive Bayes. For each test example, it builds a most appropriate rule with a local naive Bayesian classifier as its consequent. It is demonstrated that the computational requirements of LBR are reasonable in a wide cross-section of natural domains. Experiments with these domains show that, on average, this new algorithm obtains lower error rates significantly more often than the reverse in comparison to a naive Bayesian classifier, C4.5, a Bayesian tree learning algorithm, a constructive Bayesian classifier that eliminates attributes and constructs new attributes using Cartesian products of existing nominal attributes, and a lazy decision tree learning algorithm. It also outperforms, although the result is not statistically significant, a selective naive Bayesian classifier.

**Keywords:** Bayesian classification, semi-naive Bayesian classification, decision trees, decision rules, lazy learning

## 1. Introduction

Bayes' theorem provides an optimal way to predict the class of an unseen instance described by a conjunction of attribute values $V = v_1 \wedge v_2 \wedge ... \wedge v_n$ (Duda & Hart, 1973). The predicted class is the one with the highest probability given the instance $V$:

$$P(C_i \mid V) = P(C_i) \times P(V \mid C_i)/P(V). \quad (1)$$

The application of this formula in machine learning is restricted by the inability to determine accurate values for $P(V \mid C_i)$[1]. In standard machine learning applications, these probabilities must be estimated from the training data. If there were sufficient randomly sampled examples of every possible combination of attribute values, such estimation would be straightforward and acceptably reliable and accurate. In practice, however, most combinations are not represented in the training

data at all, let alone in sufficient numbers to support accurate estimation of the required conditional probabilities. Naive Bayesian classification (Kononenko, 1990; Langley, Iba, & Thompson, 1992; Langley & Sage, 1994) circumvents this problem by assuming that all attributes are mutually independent within each class. This allows the following equality to be used:[2]

$$P(V \mid C_i) = \prod_{v_j \in V} P(v_j \mid C_i). \tag{2}$$

Naive Bayesian classifier learning is simple and computationally efficient. It has been shown that in many domains the prediction accuracy of the naive Bayesian classifier compares surprisingly well with that of other more complex learning algorithms including decision tree learning, rule learning, and instance-based learning algorithms (Cestnik, Kononenko, & Bratko, 1987; Langley *et al.*, 1992; Domingos & Pazzani, 1996). In addition, the naive Bayesian classifier is robust to noise and irrelevant attributes. Some experts report that the learned theories are easy to understand (Kononenko, 1993). The naive Bayesian classifier considers evidence from many attributes to classify examples. This is important in a situation where many attributes affect the classification. However, when the attribute independence assumption is violated, which appears to be very common,[3] the performance of the naive Bayesian classifier might be poor. In other words, the performance of the naive Bayesian classifier in this kind of domain can be further improved.

This paper utilizes a variant of Bayes theorem,

$$P(C_i \mid V_1 \wedge V_2) = P(C_i \mid V_2) \times P(V_1 \mid C_i \wedge V_2) / P(V_1 \mid V_2). \tag{3}$$

This can be derived as follows, where Eq. 4 is by definition and Eq. 5 is by factorization:

$$\begin{aligned} P(C_i \mid V_1 \wedge V_2) &= P(C_i \wedge V_1 \wedge V_2) \,/\, P(V_1 \wedge V_2) && (4)\\ &= P(V_1 \mid C_i \wedge V_2) P(C_i \mid V_2) P(V_2) \,/\, P(V_1 \mid V_2) P(V_2) && (5)\\ &= P(V_1 \mid C_i \wedge V_2) P(C_i \mid V_2) \,/\, P(V_1 \mid V_2) && (6) \end{aligned}$$

Eq. 3 can be substituted for Eq. 1 with $V_1$ and $V_2$ being any two conjunctions of values such that each $v_i$ from $V$ belongs to exactly one of $V_1$ or $V_2$. A naive-Bayes-like simplifying independence assumption can be used:

$$P(V_1 \mid C_i \wedge V_2) = \prod_{v_j \in V_1} P(v_j \mid C_i \wedge V_2). \tag{7}$$

Eq. 7 is a weaker assumption than the naive Bayes assumption, Eq. 2, assuming independence between fewer variables and under stronger conditions. The more attribute values in $V_2$ the weaker the assumption required. However, a counterbalancing disadvantage of adding attribute values to $V_2$ is that the numbers of training examples from which the required conditional probabilities are estimated decrease and hence the accuracy of estimation can be expected to also decrease.

This paper describes a novel lazy Bayesian rule (LBR) approach to alleviating the attribute inter-dependence problem of the naive Bayesian classifier by utilizing

Eq. 3 in place of Eq. 1 and hence the weaker independence assumptions required by Eq. 7 in place of those of Eq. 2. In addition, it provides an extensive experimental comparison of key existing techniques for improving the performance of the naive Bayesian classifier. The following two sections discuss existing techniques for improving the naive Bayesian classifier and the motivation for LBR. Section 3 discusses lazy learning. Section 4 describes the LBR algorithm. Experimental evaluation of LBR is presented in Section 5, including a comparison with six related algorithms. The final section draws conclusions and outlines some directions for further research.

## 2. Existing techniques for improving the naive Bayesian classifier and motivation

A number of techniques have been developed to improve upon the performance of the naive Bayesian classifier. They include the semi-naive Bayesian classifier (Kononenko, 1991), attribute deletion (Langley & Sage, 1994), the constructive Bayesian classifier (BSEJ) (Pazzani, 1996), probability adjustment (Webb & Pazzani, 1998), Bayesian networks (Friedman & Goldszmidt, 1996; Sahami, 1996; Singh & Provan, 1995; 1996), the recursive Bayesian classifier (RBC) (Langley, 1993), and the naive Bayesian tree learner (NBTREE) (Kohavi, 1996). These studies have shown that it is possible to improve upon the general error performance of the naive Bayesian classifier, although Domingos and Pazzani (1996) argue that the naive Bayesian classifier is still in fact optimal when the attribute independence assumption is violated, so long as the ranks of the conditional probabilities of classes given an example are correct. The extent to which the above approaches improve upon the performance of the naive Bayesian classifier suggests that these ranks are in practice incorrect in a substantial number of cases.

Kononenko's (1991) semi-naive Bayesian classifier performs exhaustive search to iteratively join pairs of attribute values. The aim is to optimize the trade-off between the "non-naivety" and the reliability of estimates of probabilities. For the same purpose, BSEJ (Backward Sequential Elimination and Joining) adopts a wrapper model (John, Kohavi, & Pfleger, 1994), using $N$-fold cross-validation ($N$-CV, also called leave-1-out) estimation (Breiman, Friedman, Olshen, & Stone, 1984) to find the best Cartesian product attributes from existing nominal attributes for the naive Bayesian classifier (Pazzani, 1996). It also considers deleting existing attributes. Here, $N$ is the number of training examples. The Cartesian product attribute formed from two nominal attributes is a nominal attribute whose value set is the Cartesian product of the value sets of the nominal attributes. For example, two nominal attributes $A$ and $B$ have value sets $\{a_1, a_2, a_3\}$ and $\{b_1, b_2\}$ respectively. The Cartesian product attribute formed from $A$ and $B$ has the value set $\{a_1b_1, a_1b_2, a_2b_1, a_2b_2, a_3b_1, a_3b_2\}$.

Langley and Sage (1994) have shown that attribute deletion can improve upon the performance of the naive Bayesian classifier when attributes are inter-dependent, especially when some attributes are redundant. This technique is referred to as selective naive Bayesian classification. In Langley and Sage's (1994) study, the

Forward Sequential Selection (FSS) method is used for selecting a subset of the available attributes with which to build a naive Bayesian classifier. Pazzani (1996) also investigates attribute deletion for naive Bayesian classifiers using the Backward Sequential Elimination (BSE) and FSS approaches (Kittler, 1986). In addition, Kubat, Flotzinger, and Pfurtscheller (1993) show that using decision tree learning as a pre-process to select attributes for naive Bayesian classification performs better than either decision tree learning or naive Bayesian classification alone in a domain for discovering patterns in EEG-signals.

Instead of manipulating the set of attributes for generating the naive Bayesian classifier, Webb and Pazzani (1998) propose a technique for adjusting the probabilities inferred by the naive Bayesian classifier. The adjustments are numeric weights learned for each class. These weights can correct incorrect ranking of the conditional probabilities produced by the naive Bayesian classifier due to certain violations of the attribute independence assumption.

In order to relax the attribute independence assumption of naive Bayesian classification, Friedman and Goldszmidt (1996) explore the Tree Augmented Naive Bayes (TAN) model for classifier learning. The tree augmented naive Bayes model belongs to a restricted sub-class of Bayesian network (Pearl, 1988), which is previously proposed by Geiger (1992) using a method introduced by Chow & Liu (1968). Experimental comparisons suggest that TAN can frequently reduce the error of the naive Bayesian classifier. Sahami (1996) proposes a framework using Bayesian networks and a learning algorithm, KDB, for studying approaches to relaxing the restriction on attribute independencies of the naive Bayesian classifier. KDB learns Bayesian classifiers that allow each attribute to depend on at most $k$ other attributes within a class for a given number $k$. When $k$ is equal to 0, KDB generates naive Bayesian classifiers, while when $k$ is equal to the number of all attributes - 1, KDB creates full Bayesian classifiers without attribute independencies. Singh and Provan (1995; 1996) investigate the forward sequential attribute subset selection method for learning Bayesian networks and compare it with conventional Bayesian networks, the naive Bayesian classifier, and the selective naive Bayesian classifier.

RBC (Langley, 1993) alleviates the attribute inter-dependence problem of naive Bayesian classification by identifying regions of the instance space in which the independence assumption holds. It recursively splits the instance space into subspaces using a tree structure. Each internal node of the tree is a naive Bayesian classifier that divides the local training examples at the node into clusters of which each corresponds to an instance sub-space of the (sub) space at the node. Each cluster usually contains training examples from more than one class. When a cluster consists of training examples from only one class, the tree growing procedure halts. Langley (1993) shows that RBC can perform well learning conjunctive concepts which involve attribute inter-dependencies using artificial domains. However, RBC did not prove superior to the naive Bayesian classifier on a set of natural domains.

Kohavi (1996) proposes NBTREE as a hybrid approach combining the naive Bayesian classifier and decision tree learning (Quinlan, 1993; Breiman *et al.*, 1984). It has been shown that NBTREE frequently achieves higher accuracy than either

a naive Bayesian classifier or a decision tree learner (Kohavi, 1996). Like RBC, NBTREE also uses a tree structure to split the instance space into sub-spaces and generates one naive Bayesian classifier in each sub-space. However, it uses a different splitting method and generates Bayesian trees. The decision nodes of these trees contain the univariate tests of conventional decision trees. Each leaf of a Bayesian tree contains a local naive Bayesian classifier that does not consider attributes involved in tests on the path leading to the leaf. Whereas a conventional decision tree labels each leaf with a single class and predicts this class for examples that reach the leaf, the naive Bayesian tree uses a local naive Bayesian classifier to predict the classes of these examples. Appropriate selection of tests may enable the tree to factor out damaging attribute inter-dependencies for local naive Bayesian classifiers at leaves. This can be illustrated with reference to Eq. 3. Suppose that we wish to classify an instance $\langle A = v_{A1}, B = v_{B1}, D = v_{D1}, E = v_{E1}\rangle$. Suppose further that $A$ is not conditionally independent of any of $B$, $D$, or $E$, given the class, but that the latter are conditionally independent of each other given the class and $A$. If we substitute $v_{A1}$ for $V_2$ and $v_{B1}\wedge v_{D1}\wedge v_{E1}$ for $V_1$ in Eq. 3, we obtain

$$P(C_i \,|\, v_{A1}\wedge v_{B1}\wedge v_{D1}\wedge v_{E1}) = P(C_i \,|\, v_{A1}) \times \frac{P(v_{B1}\wedge v_{D1}\wedge v_{E1} \,|\, C_i\wedge v_{A1})}{P(v_{B1}\wedge v_{D1}\wedge v_{E1} \,|\, v_{A1})}. \quad (8)$$

As

$$P(C_i \,|\, v_{A1}) \times \frac{P(v_{B1}\wedge v_{D1}\wedge v_{E1} \,|\, C_i\wedge v_{A1})}{P(v_{B1}\wedge v_{D1}\wedge v_{E1} \,|\, v_{A1})} \propto P(C_i \,|\, v_{A1}) \times P(v_{B1}\wedge v_{D1}\wedge v_{E1} \,|\, C_i\wedge v_{A1}) \quad (9)$$

we need not estimate $P(v_{B1}\wedge v_{D1}\wedge v_{E1} \,|\, v_{A1})$. As $B$, $D$, and $E$ are conditionally independent given the class and $v_{A1}$,

$$P(v_{B1}\wedge v_{D1}\wedge v_{E1} \,|\, C_i\wedge v_{A1}) = \prod_{v\in\{v_{B1},v_{D1},v_{E1}\}} P(v \,|\, C_i\wedge v_{A1}). \quad (10)$$

The relative probability of each class can thus be calculated by

$$P(C_i \,|\, v_{A1}) \times \prod_{v\in\{v_{B1},v_{D1},v_{E1}\}} P(v \,|\, C_i\wedge v_{A1}) \quad (11)$$

which is the value estimated by an NBTree below a single split on $A$ for cases with values $v_{A1}\wedge v_{B1}\wedge v_{D1}\wedge v_{E1}$.

However, tree learning algorithms, including NBTREE and other conventional decision tree learning algorithms such as C4.5 (Quinlan, 1993), suffer from the *problem of small disjuncts* (Holte, Acker, & Porter, 1989; Ting, 1994a). This type of algorithm usually tries to build a single tree that is most appropriate, on average, to all examples. However, it is likely that this tree is not good for those examples that match paths with few training examples at their leaves. The prediction performance of such leaves is usually very poor (Holte *et al.*, 1989). Rule learning algorithms may also suffer from this problem, since some rules in a rule set may only cover few training examples (Holte *et al.*, 1989).

To avoid splits with little value, NBTree requires that the relative error reduction of a split should be greater than 5% and there should be at least 30 training instances at a decision node. Even so, when a decision node does not evenly divide the instances between its branches, leaves with few training examples may result. Local naive Bayesian classifiers at such leaves are likely to be unreliable. For example, if a split divides a training set of 60 examples into two subsets with 55 and 5 examples respectively, the overall relative error reduction might be greater than the threshold 5%. In this case, the local naive Bayesian classifier at the leaf with 55 training examples might perform reasonably well. It dominates the overall error reduction of the split. This results in acceptance of the split. Nevertheless, the other local naive Bayesian classifier at the small leaf is not trustworthy, because the number of training examples used to generate it is too small. Later on, when the Bayesian tree is used to classify unseen examples, examples that are traced down to this small leaf are very likely to be classified incorrectly. This problem may degrade the performance of NBTree in some domains.

## 3. Lazy learning

The new algorithm we propose utilizes lazy learning (Aha, 1997), of which the archetypal example is nearest neighbor or instance-based learning (Cover & Hart, 1967; Hart, 1968; Gates, 1972; Dasarathy, 1980; Aha, Kibler, & Albert, 1991). Lazy learning is distinguished by delaying computation until classification time. No explicit theories, such as decision trees or rules, are created at training time. When facing a test example, lazy learning algorithms access training examples to make a prediction. One such algorithm is LazyDT. Although it is called a lazy decision tree learning algorithm (Friedman, Kohavi, & Yun, 1996), it can be considered to generate decision rules[4] at classification time. For each example to be classified, LazyDT builds one rule that is most appropriate to the example by using an entropy measure. The antecedent of the rule is a conjunction of conditions in the form of attribute-value pairs. The consequent of the rule is the class to be predicted, being the majority class of the training examples that satisfy the antecedent of the rule. With caching of relevant information from classification to classification, LazyDT is reasonably fast (Friedman *et al.*, 1996). This provides another approach to alleviating the small disjunct problem of decision tree learning. Osr (Briand and Thomas, 1992) is an approach with some similarities to LazyDT. It differs by developing a number of rules for each case and by allowing the use of a Bayesian loss matrix to select the best classification, rather than simply using the majority class of the cases selected by a rule. Also related is the Learning All Rules technique (Webb, 1996; Viswanathan & Webb, 1998) that uses complete search to select the decision rule with the highest Laplace accuracy estimate from those that cover the case to be classified.

Fulton, Kasif, Salzberg, and Waltz (1996) describe a number of further variations on this lazy learning scheme, exploring a number of alternative techniques that can be used to select training cases for a given test case. These selected training cases

are used to construct one or more decision trees that are finally used to classify the test case.

## 4. The lazy Bayesian rule learning algorithm

Our proposal is to use lazy learning to learn Bayesian rules at classification time. LBR is similar to LAZYDT with respect to performing lazy learning of decision rules. For each test example, LBR also builds an individual rule that is most appropriate to the test example, albeit, using a different technique. However, whereas the consequent of a rule in LAZYDT is a single class that is used for classification, LBR uses a local naive Bayesian classifier. LBR can be considered as a combination of the two techniques NBTREE and LAZYDT. Alternatively, it can be considered as a lazy approach to classification using Eq. 3. At classification time, for each case to be classified, the attribute values in $V$ are allocated to $V_1$ and $V_2$ in a manner that is expected to minimize estimation error. $V_2$ is the antecedent of the rule and $V_1$ is the set of attribute values used in the local naive Bayesian classifier.

Lazy learning techniques can be viewed in terms of a two stage process. First, a subset of training cases is selected for a case to be classified (the k closest for nearest neighbor or instance-based learning, those matching the decision rule for LAZYDT, and so on). Next, these selected cases are used as a training set for a learning technique that ultimately forms a classifier with which the test case is classified. For a nearest neighbor learning algorithm, LAZYDT, OSR, and Learning All Rules, this classifier is a majority class classifier. Fulton *et al.* (1996) use one or more decision trees. All these existing techniques seek to include in the subset those cases that seem pertinent to classifying the case. For selecting such subsets, they use criteria that are not directly relevant to the classifiers used in the second stage. In contrast, we are proposing an approach that seeks to exclude from the subset only those cases for which there is evidence that inclusion will be harmful to the classifier used in the second stage, a naive Bayesian classifier.

In order to avoid the small disjunct problem from which NBTREE may suffer, LBR uses lazy learning. It retains all training examples until classification time. Before classifying a test example, LBR generates a rule (called a *Bayesian rule*) that is most appropriate to the test example. This contrasts with creating a theory at training time, such as a single tree, that is, on average, most appropriate to all examples, as NBTREE does. The antecedent of a Bayesian rule is a conjunction of attribute-value pairs (conditions) each in the form of (attribute = value). The current version of LBR can only directly deal with nominal attributes. Numeric attributes are discretized as a pre-process. The consequent of a Bayesian rule is a local naive Bayesian classifier created from those training examples (called *local training examples*) that satisfy the antecedent of the rule. This local naive Bayesian classifier only uses those attributes that do not appear in the antecedent of the rule.

During the generation of a Bayesian rule, the test example to be classified is used to guide the selection of attributes for creating attribute-value pairs. The values in the attribute-value pairs are always the same as the corresponding attribute values of the test example. The objective is to grow the antecedent of a Bayesian rule that

ultimately decreases the errors of the local naive Bayesian classifier in the consequent of the rule. The antecedent of the Bayesian rule defines a sub-space of the instance space to which the test example belongs. This sub-space selects a subset of the available training instances. For all instances in the instance sub-space, each of the attributes occurring in the antecedent has the value specified in the antecedent. In consequence, these attributes can be excluded from the local naive Bayesian classifier as irrelevant to classification within the instance sub-space. These attributes are removed from the local naive Bayesian classifier for computational efficiency. Finally, the local naive Bayesian classifier of the Bayesian rule classifies the test example, since this example satisfies the antecedent of the rule. This is the same as dividing the available attribute values into two sets, assigning one to $V_1$ and the other to $V_2$ and then employing Eq. 3 assuming conditional independence between the attribute values in $V_1$ when conditioned by $V_2$ and $C_i$. The attribute values assigned to $V_2$ are the values in the antecedent of the Bayesian rule.

Note that including an attribute $A$ in the antecedent of a Bayesian rule factors out any distortions of inferred probabilities due to conditional inter-dependencies between $A$ and any other attribute.

### *4.1. An operational description of LBR*

Table 1 outlines the LBR algorithm. For a given training set and each test example, LBR starts from a special Bayesian rule whose antecedent is *true*. Its local naive Bayesian classifier in the consequent part is trained on the entire training set using all attributes. This Bayesian rule is identical to a conventional naive Bayesian classifier. LBR then uses a greedy search to grow an antecedent that matches the test example, with the aim of reducing the errors of its local naive Bayesian classifier. The inference is made that attributes that most seriously affect error have the most harmful conditional interdependencies and hence are most important to factor out.

Any attribute $A$ that is added to the antecedent is removed from the local naive Bayesian classifier. This effectively removes any redundant attributes of $A$, since these redundant attributes have the same value $v_A$ on all examples in the instance sub-space defined by the antecedent of the rule. Therefore, they do not affect the classification behavior of the local naive Bayesian classifier.

During the growth of a Bayesian rule, each candidate Bayesian rule is evaluated by performing $N$-fold cross-validation estimation of its local naive Bayesian classifier on the local training set. We choose $N$-CV as the evaluation method because $N$-CV errors are more reliable estimates of true errors than re-substitution errors (Breiman *et al.*, 1984) and for a naive Bayesian classifier, the operations of removing and adding an example are very easy and efficient. The $N$-CV errors of a Bayesian rule form a baseline reference against which the performance of rules formed by adding another condition is evaluated.

At each step of the greedy search, LBR tries to add, to the current Bayesian rule, each attribute that has not already been in the antecedent of the rule, so long as its value on the test example is not missing. The objective is to determine whether including a test on this attribute can significantly improve upon the classification

*Table 1.* The Lazy Bayesian Rule learning algorithm

**LBR**($Att, D_{training}, E_{test}$)
  *INPUT*: *Att*: a set of attributes,
    $D_{training}$: a set of training examples described using *Att* and classes,
    $E_{test}$: a test example described using *Att*.
  *OUTPUT*: a predicted class for $E_{test}$.
*LocalNB* = a naive Bayesian classifier trained using *Att* on $D_{training}$
*Errors* = errors of *LocalNB* estimated using $N$-CV on $D_{training}$
*Cond* = *true*
*REPEAT*
  $TempErrors_{best}$ = the number of examples in $D_{training} + 1$
  *FOR* each attribute $A$ in *Att* whose value $v_A$ on $E_{test}$ is not missing *DO*
    $D_{subset}$ = examples in $D_{training}$ with $A = v_A$
    *TempNB* = a naive Bayesian classifier trained using $Att - \{A\}$ on $D_{subset}$
    *TempErrors* = errors of *TempNB* estimated using $N$-CV on $D_{subset}$ +
      errors from *Errors* for examples in $D_{training} - D_{subset}$
    *IF* ((*TempErrors* < $TempErrors_{best}$) *AND*
      (*TempErrors* is significantly lower than *Errors*))
    *THEN*
      $TempNB_{best}$ = *TempNB*
      $TempErrors_{best}$ = *TempErrors*
      $A_{best} = A$
  *IF* (an $A_{best}$ is found)
  *THEN*
    *Cond* = *Cond* $\wedge$ ($A_{best} = v_{A_{best}}$)
    *LocalNB* = $TempNB_{best}$
    $D_{training} = D_{subset}$ corresponding to $A_{best}$
    $Att = Att - \{A_{best}\}$
    *Errors* = errors of *LocalNB* estimated using $N$-CV on $D_{training}$
  *ELSE*
    *EXIT* from the *REPEAT* loop
classify $E_{test}$ using *LocalNB*
*RETURN* the class

accuracy. Adding one condition to a Bayesian rule can be considered as reducing the instance space (or sub-space) defined by the rule to a further sub-space and moving its local naive Bayesian classifier to this new sub-space. If the attribute being added causes damaging attribute inter-dependencies in the current local naive Bayesian classifier, the new local naive Bayesian classifier in the reduced instance sub-space should have lower errors.

Attributes with missing values on the test example are ignored both when the local naive Bayesian classifier computes the posterior probabilities $P(C_i \mid V)$ and when LBR selects attribute-value pairs to grow a Bayesian rule. This gives LBR a natural approach to dealing with missing values for test examples. When computing the frequencies of attribute values given each class from the local training set, as the estimates of the conditional probabilities $P(v_j \mid C_i)$ for a local naive Bayesian classifier, the algorithm does not count missing attribute values.

The utility of each possible attribute-value pair to be added to the antecedent of a Bayesian rule is evaluated in the following manner. The local training examples that satisfy the attribute-value pair are used to train a temporary naive Bayesian classifier using all attributes that do not occur in the antecedent of the current Bayesian rule and are not the attribute being examined. These examples are then classified by conducting a $N$-CV using this temporary naive Bayesian classifier. The metric used to evaluate the benefit obtained by using an attribute-value pair for growing the current Bayesian rule is calculated as follows. The errors of the temporary naive Bayesian classifier on the local training examples that satisfy the attribute-value pair are calculated. To these are added the errors of the existing local naive Bayesian classifier on the remaining local training examples.[5] If this measure is lower than the errors of the local naive Bayesian classifier of the current Bayesian rule at a significance level better than 0.05 using a one-tailed pairwise sign-test (Chatfield, 1978), this attribute-value pair becomes a candidate condition to be added to the current Bayesian rule. The sign-test is used to control the likelihood of adding conditions that reduce error by chance.

At the end of this step, the candidate attribute-value pair with the lowest measure (errors) is added to the antecedent of the current Bayesian rule. All the current local training examples that satisfy this attribute-value pair form the local training set of the new Bayesian rule, while all other examples are discarded. Those training examples whose values for this attribute are missing are treated as not satisfying the attribute-value pair, and are removed. The temporary naive Bayesian classifier corresponding to this attribute-value pair becomes the local naive Bayesian classifier of the new Bayesian rule. Its $N$-CV estimation errors on the new local training set will be used as a reference for further growing the Bayesian rule.

This process is repeated until no candidate attribute-value pair can be found. Then, the growth of the Bayesian rule stops. This happens when no damaging attribute inter-dependencies exist for the local naive Bayesian classifier, or the local training set is too small to further reduce the instance sub-space by specializing the antecedent of the Bayesian rule. In such cases, further growing the Bayesian rule would not significantly reduce its errors. Then, the local naive Bayesian classifier of this Bayesian rule is used to classify the test example.

### *4.2. Characteristics of LBR*

In the process of generating a Bayesian rule, there is a trade-off between decreasing error by removing harmful attributes and increasing error as a result of reducing the accuracy of the probability estimations of the local naive Bayesian classifier due to decreases in the size of the available training set. The use of a sign-test to filter candidate conditions manages this trade-off by rejecting the relatively small decreases in estimated error that result from rules with small local training sets.

For LBR, each rule can be considered in isolation from others. Between two attribute-value pairs that result in Bayesian rules with similar error rates, the attribute-value pair selection criteria for growing a Bayesian rule in LBR implicitly favor an attribute-value pair that makes the antecedent of the Bayesian rule cover more training examples. LBR tries to create a single Bayesian rule that is most appropriate to a given test case rather than a theory that is most appropriate, on average, to all cases. Thus, it is free to select and consider all examples that may relate to the test case, giving greater scope for utilizing a large number of training examples for any test case. In addition, LBR's stopping criteria cease the growth of a Bayesian rule before the local training set becomes too small. All these factors guard LBR against the small disjunct problem.

It is important to notice that building a Bayesian rule with attributes causing damaging attribute inter-dependencies included in its antecedent is better than just deleting these attributes for naive Bayesian classification. The reason is that some information which is important for classification is lost when deleting these attributes, while this information is retained in the local naive Bayesian classifier of the Bayesian rule as a result of the composition of the instance sub-space. For example, consider a learning problem with a target concept of *exclusive-or* of two binary attributes $A$ and $B$. The naive Bayesian classifier has poor performance with this problem, since these two attributes are inter-dependent. After deleting either $A$ or $B$, the naive Bayesian classifier will still not perform well, since a half of the information for determining the classes of examples is lost. However, for those test examples with $A = 0$, we can build a Bayesian rule with $(A = 0)$ as its antecedent. This Bayesian rule can easily solve the problem, since there are no attribute inter-dependencies in the sub-space defined by $(A = 0)$ and the classes of examples in this sub-space are fully determined by the values of $B$. In short, building Bayesian rules can alleviate damaging attribute inter-dependencies for naive Bayesian classification.

Like other lazy learning algorithms, LBR needs to keep all the training examples for use at classification time. In contrast, non-lazy learning algorithms, such as C4.5 (Quinlan, 1993) and the naive Bayesian classifier, learn a theory at training time. After this theory is learned, the training examples can be discarded. Therefore, LBR may have higher memory requirements than non-lazy learning algorithms at the classification stage, especially when training sets are very large.

## 5. Experiments

The theoretical advantages of LBR over naive Bayesian classification, decision tree learning, and Bayesian tree learning have been discussed. In this section, we provide empirical evaluation of the relative performance of these approaches. Here, we focus on the prediction accuracy of the algorithms.

The following subsection briefly describes six learning algorithms used for comparison. Then, the experimental domains and methods are presented in the second subsection. Subsection 5.3 reports and discusses error rates of these algorithms. Subsection 5.4 further explores in what situations LBR outperforms the naive Bayesian classifier in terms of lower error rate. Subsection 5.5 provides experimental evidence that LBR can avoid the small disjunct problem. The final subsection briefly addresses the computational requirements of LBR.

### *5.1. Comparison algorithms*

The six algorithms used for comparison with LBR are NB, C4.5, BT, BSEJ, BSE, and LAZYTREE. We describe them, and our motivations for including them in the study, as follows.

The primary design aim for LBR is to improve upon the performance of the naive Bayesian classifier, so it is compared with NB, our implementation of the naive Bayesian classifier. Its working principle is as described at the beginning of this paper. When the probability of an attribute value conditional on a class is estimated from a training set, the m-estimate (Cestnik, 1990) with $m = 2$ is used. When the probability of a class is estimated, the Laplace estimate (Cestnik, 1990) is used. At the training stage, NB does not count missing attribute values of training examples. During classification, NB ignores attributes with missing values on the test example being classified. To enhance comparability of results, NB is used to create the local naive Bayesian classifiers for LBR, BT, BSEJ, and BSE.

Since LBR was motivated by a naive Bayesian tree learning algorithm, and we argue that LBR can avoid the small disjunct problem from which tree learning algorithms may suffer, it is interesting to compare its performance with that of a conventional decision tree learning algorithm. Here, C4.5 (release 8) (Quinlan, 1996), a state-of-the-art decision tree learning algorithm, is used. We report results for pruned trees (Quinlan, 1993). C4.5 uses a probabilistic approach for dealing with missing values in a training set. When partitioning training examples at a decision node, a training example with a missing value of the test attribute at the decision node is partially assigned to each branch of the node. The fraction of the example assigned to a branch is in proportion to the fraction of examples with known values of that branch in the set of examples with known values at the node. When classifying an unseen example with an unknown value of a test attribute, C4.5 explores all the branches and combines the resulting classifications arithmetically. The result is then a class distribution rather than a single class. The final predicted class is the one with the highest frequency. For other details about C4.5, please refer to Quinlan (1993).

We have mentioned before that LBR is similar to NBTREE in the sense that they use a conjunction of attribute-value pairs to define an instance sub-space to build a local naive Bayesian classifier, the most significant difference being that LBR builds a separate Bayesian rule for each instance to be classified while NBTREE builds a single model that is used to classify all unseen instances. Therefore, LBR is compared with BT, a Bayesian tree learning algorithm. BT is our implementation of Kohavi's (1996) NBTREE. Like NBTREE, BT also uses at least 5% relative error reduction and 30 training instances at a node as the stopping criterion for splitting the node. The utility of a split is the weighted sum of the utility of its nodes as in NBTREE, but the utility of a node is the accuracy of the local naive Bayesian classifier at the node estimated using a $N$-CV instead of a 5-fold cross-validation. BT adopts the method of dealing with missing values used by C4.5 when building and using tree structures (Quinlan, 1993) as described above. When training local naive Bayesian classifiers and using them to classify examples, BT uses the same approaches to handling missing values as LBR. Actually, both systems use NB for local naive Bayesian classification as mentioned before.

BSEJ (Pazzani, 1996) uses constructive induction and attribute deletion to alleviate the attribute inter-dependence problem of the naive Bayesian classifier. It is also included in the comparison. BSEJ uses a wrapper model (John *et al.*, 1994) with $N$-CV estimation to join and delete attributes. A greedy search, at each step, either deletes one attribute or creates one new attribute through joining (generating the Cartesian product of) two attributes. BSEJ starts from the set of all the original attributes, and stops when neither join nor deletion can improve upon the accuracy of the naive Bayesian classifier estimated using $N$-CV on the training set. Then, the naive Bayesian classifier built on the current set of attributes including new attributes is returned as the final classifier. The BSEJ algorithm used here is our implementation.

The selective naive Bayesian classifier BSE is our implementation of the backward sequential attribute elimination algorithm for naive Bayesian classification (Pazzani, 1996; Kittler, 1986). It is exactly the same as BSEJ except that at each step of the greedy search, BSE only considers deleting one existing attribute.

Another algorithm with which LBR should be compared is LAZYDT (Friedman *et al.*, 1996). LAZYDT also generates rules (Friedman *et al.*, 1996, refer to them as decision paths) using lazy learning. The main difference between these two algorithms is that the consequent of a rule, for LAZYDT, is the majority class of the local training examples, while it is a local naive Bayesian classifier for LBR. In this paper, we use our implementation of LAZYDT, called LAZYTREE. LAZYTREE uses the same techniques as LAZYDT except that the caching scheme (Friedman *et al.*, 1996) is not implemented in LAZYTREE since it only reduces the computational requirements of the algorithm and does not affect its accuracy.

### *5.2. Experimental domains and methods*

Twenty-nine natural domains are used in the experiments reported here. They include all the domains (twenty-eight) used by Domingos and Pazzani (1996) for

*Table 2.* Description of learning tasks

| Domain | Size | No. of Classes | No. of Attributes | |
|---|---|---|---|---|
| | | | Numeric | Nominal |
| Annealing processes | 898 | 6 | 6 | 32 |
| Audiology | 226 | 24 | 0 | 69 |
| Breast cancer (Wisconsin) | 699 | 2 | 9 | 0 |
| Chess (King-rook-vs-king-pawn) | 3169 | 2 | 0 | 36 |
| Credit screening (Australia) | 690 | 2 | 6 | 9 |
| Echocardiogram | 131 | 2 | 6 | 1 |
| Glass identification | 214 | 6 | 9 | 0 |
| Heart disease (Cleveland) | 303 | 2 | 13 | 0 |
| Hepatitis prognosis | 155 | 2 | 6 | 13 |
| Horse colic | 368 | 2 | 7 | 15 |
| House votes 84 | 435 | 2 | 0 | 16 |
| Hypothyroid diagnosis | 3163 | 2 | 7 | 18 |
| Iris classification | 150 | 3 | 4 | 0 |
| Labor negotiations | 57 | 2 | 8 | 8 |
| LED 24 (noise level = 10%) | 200 | 10 | 0 | 24 |
| Liver disorders | 345 | 2 | 6 | 0 |
| Lung cancer | 32 | 3 | 0 | 56 |
| Lymphography | 148 | 4 | 0 | 18 |
| Pima Indians diabetes | 768 | 2 | 8 | 0 |
| Postoperative patient | 90 | 3 | 1 | 7 |
| Primary tumor | 339 | 22 | 0 | 17 |
| Promoter gene sequences | 106 | 2 | 0 | 57 |
| Solar flare | 1389 | 2 | 0 | 10 |
| Sonar classification | 208 | 2 | 60 | 0 |
| Soybean large | 683 | 19 | 0 | 35 |
| Splice junction gene sequences | 3177 | 3 | 0 | 60 |
| Tic-Tac-Toe end game | 958 | 2 | 0 | 9 |
| Wine recognition | 178 | 3 | 13 | 0 |
| Zoology | 101 | 7 | 0 | 16 |

studying naive Bayesian classification. In addition, the Tic-Tac-Toe domain is chosen because it contains inter-dependent attributes and its target concept is known. We want to evaluate whether LBR can really increase the prediction accuracy of a naive Bayesian classifier in this type of domain. This test suite covers a wide variety of domains from the UCI machine learning repository (Blake, Keogh, & Merz, 1998). Table 2 gives a summary of the characteristics of these domains, including dataset size, the number of classes, the number of numeric attributes, and the number of nominal attributes.

In each domain, two stratified 10-fold cross-validations (Breiman, Friedman, Olshen, & Stone, 1984; Kohavi, 1995) are carried out for each algorithm. All the algorithms are run with their default option settings on the same training and test set partitions in every domain. An error rate reported in the following subsections is an average over the 20 trials for an algorithm.

Since some of these algorithms can only deal with nominal attributes, numeric attributes are discretized as a pre-process in the experiments. An entropy-based discretization method (Fayyad & Irani, 1993; Ting, 1994b) is used. For each pair of training set and test set, both training set and test set are discretized by using cut points found from the training set alone.

### *5.3. Error rates*

Figures 1, 2, and 3 compare the error rates of LBR to each of NB, C4.5, BT, BSEJ, BSE, and LAZYTREE in the 29 domains. The domains (the X axis) are sorted in ascending order of the error differences of the two corresponding algorithms such that the two curves cross only once in each graph. A bar in the figures indicates one standard error on each side of a curve. The detailed error rates of these seven algorithms are available in Table 3. Their standard errors are given in Table 4. For ease of comparison, error rate ratios of LBR over other algorithms are derived from Table 3 and presented in Table 5. An error rate ratio, for example for LBR vs NB, presents a result for LBR divided by the corresponding result for NB - a value less than 1 indicates an improvement due to LBR. To compare the error rates of two algorithms in a domain, a one-tailed pairwise t-test (Chatfield, 1978) on the error rates of the 20 trials is carried out. The difference is considered as significant, if the significance level of the t-test is better than 0.05. In Table 5, **boldface** (*italic*) font, for example for LBR vs NB, indicates that LBR is significantly more (less) accurate than NB. The second last row in Table 5 shows the numbers of wins, ties, and losses between the error rates of the corresponding two algorithms in the 29 domains, and the significance level of a one-tailed pairwise sign-test (Chatfield, 1978) on these win/tie/loss records.[6] The last row in this table presents similar comparison results but treating insignificant wins and losses in individual domains as ties.

From Figures 1, 2, and 3 as well as Tables 3 and 5, we have the following observations:

(1) Of the seven learning algorithms, LBR achieves the lowest average error rate across the 29 domains.

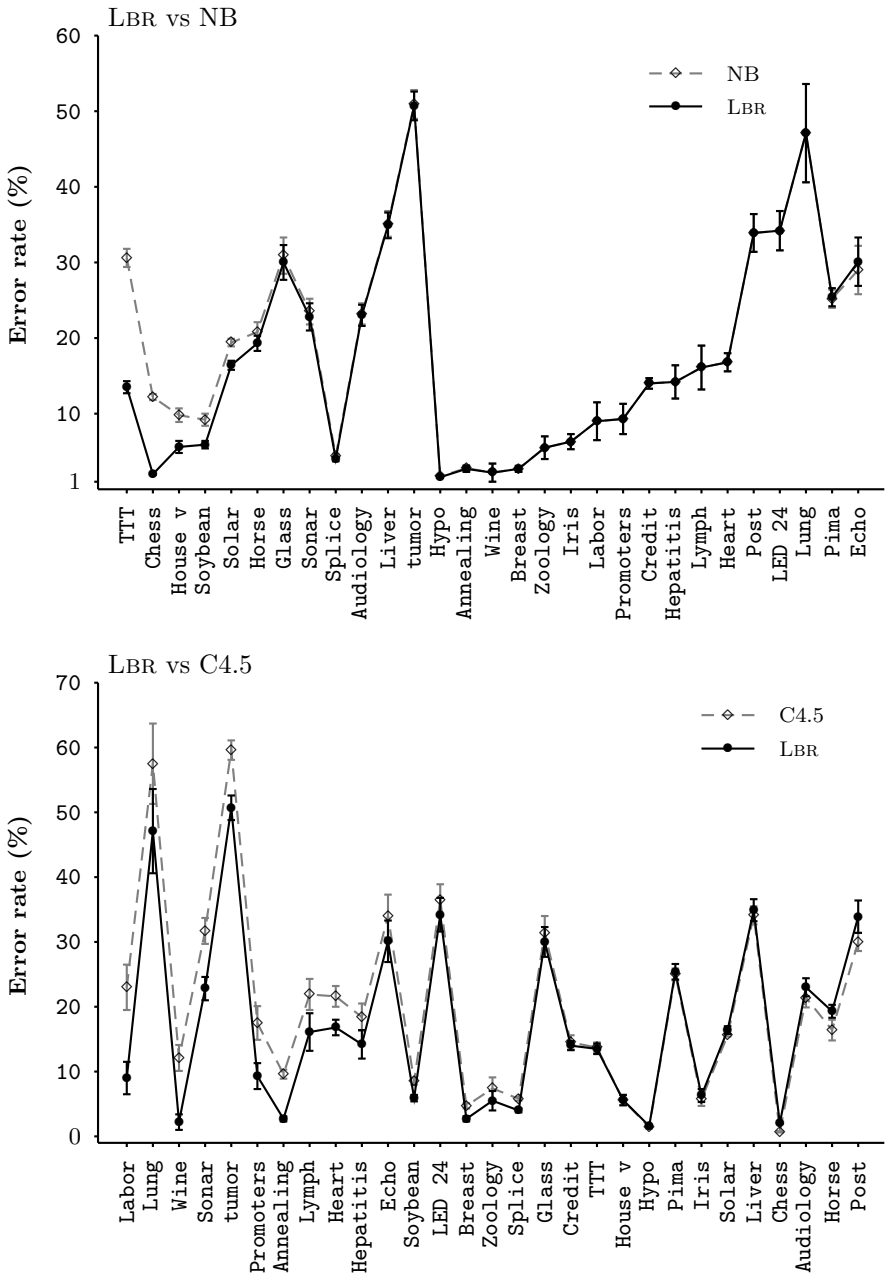


*Figure 1.* Comparing the error rates of LBR to those of NB and C4.5. A one-tailed pairwise sign-test shows that against each alternative LBR has lower error significantly more often than it has higher error.

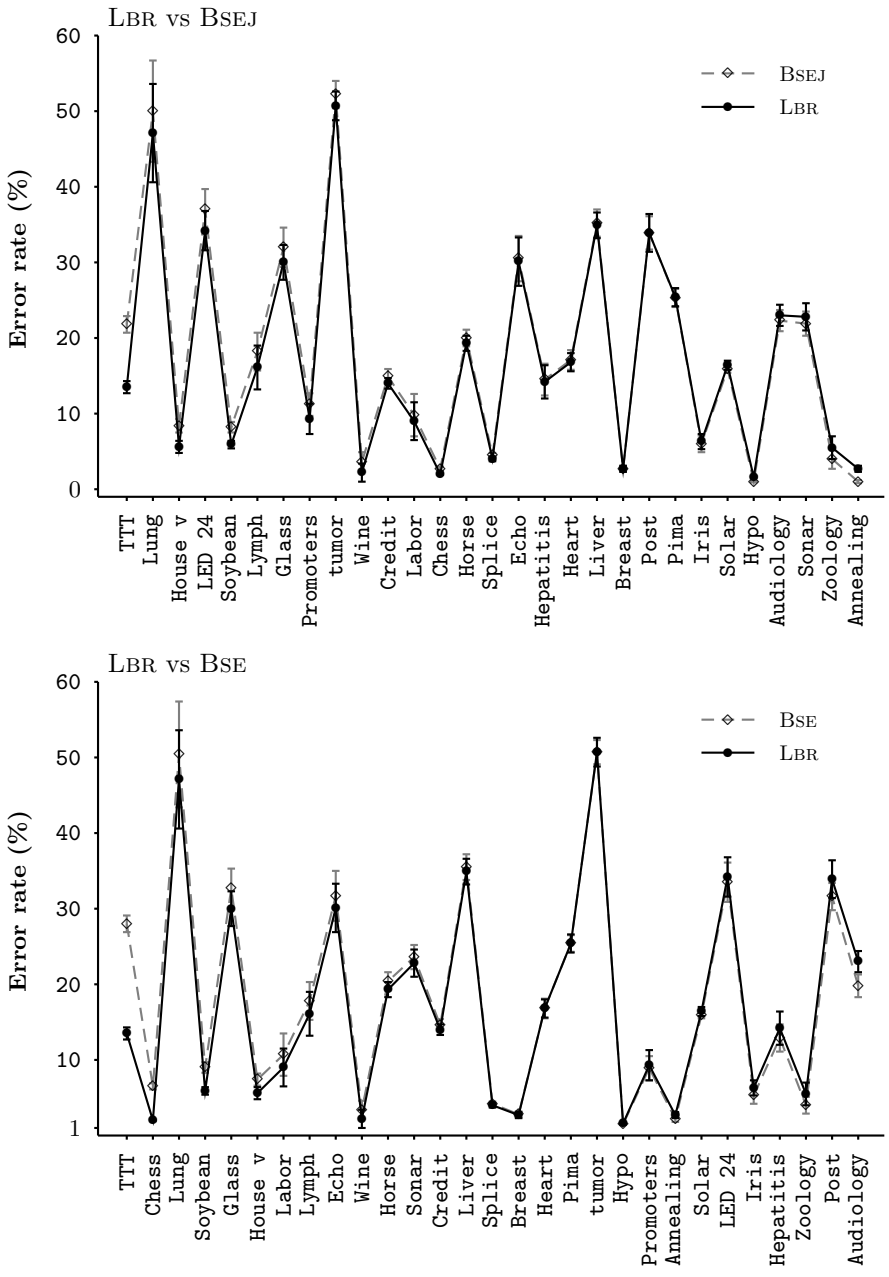


*Figure 2.* Comparing the error rates of LBR to those of BSEJ and BSE. Against each alternative, LBR has higher accuracy more often than lower accuracy. A one-tailed pairwise sign-test shows that this accuracy advantage is significant with respect to BSEJ.

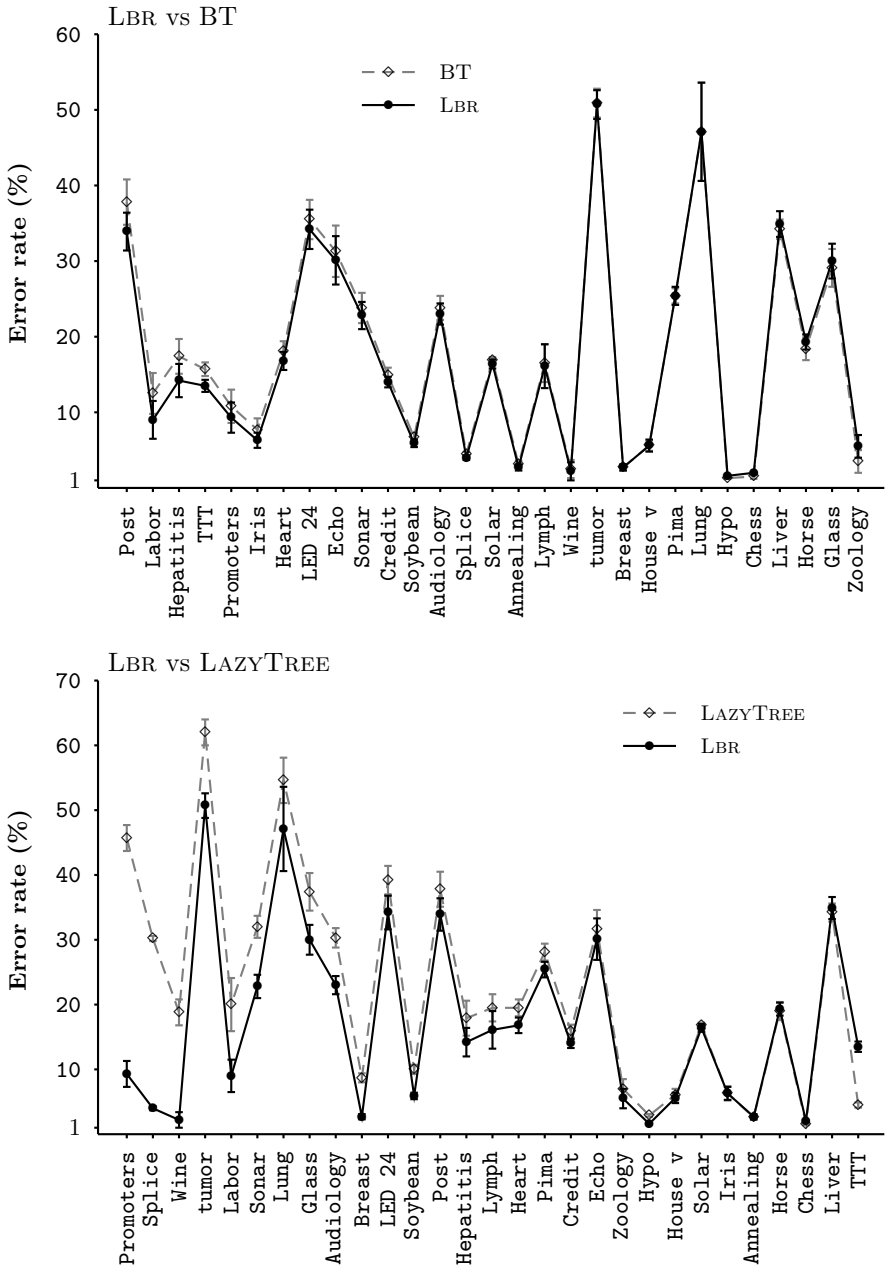


*Figure 3.* Comparing the error rates of LBR to those of BT and LAZYTREE. A one-tailed pairwise sign-test shows that against each alternative LBR has lower error significantly more often than it has higher error.

*Table 3.* Error rates (%) of NB, C4,5, BT, BSEJ, BSE, LAZYTREE, and LBR. LBR obtains a lower average error rate than any other algorithm.

| Domain | NB | C4.5 | BT | BSEJ | BSE | LAZYTREE | LBR |
|---|---|---|---|---|---|---|---|
| Annealing | 2.8 | 9.6 | 3.1 | 1.0 | 2.2 | 2.6 | 2.7 |
| Audiology | 23.2 | 21.4 | 23.8 | 22.3 | 19.8 | 30.3 | 23.0 |
| Breast (W) | 2.7 | 4.7 | 2.7 | 2.7 | 2.8 | 8.7 | 2.7 |
| Chess (KR-KP) | 12.2 | 0.7 | 1.5 | 2.7 | 6.5 | 1.6 | 2.0 |
| Credit (A) | 14.0 | 14.5 | 14.9 | 14.9 | 14.6 | 15.9 | 14.0 |
| Echocardiogram | 29.0 | 34.0 | 31.3 | 30.5 | 31.7 | 31.7 | 30.1 |
| Glass | 30.9 | 31.4 | 29.1 | 32.0 | 32.7 | 37.4 | 30.0 |
| Heart (C) | 16.8 | 21.6 | 18.1 | 17.1 | 16.8 | 19.5 | 16.8 |
| Hepatitis | 14.2 | 18.4 | 17.4 | 14.5 | 12.9 | 17.9 | 14.2 |
| Horse colic | 20.8 | 16.4 | 18.4 | 20.0 | 20.4 | 19.0 | 19.3 |
| House votes 84 | 9.8 | 5.6 | 5.6 | 8.4 | 7.4 | 6.1 | 5.6 |
| Hypothyroid | 1.7 | 1.4 | 1.3 | 1.0 | 1.5 | 2.9 | 1.6 |
| Iris | 6.3 | 5.7 | 7.7 | 6.0 | 5.3 | 6.3 | 6.3 |
| Labor | 9.0 | 23.0 | 12.5 | 9.8 | 10.7 | 20.0 | 9.0 |
| LED 24 | 34.2 | 36.5 | 35.5 | 37.0 | 33.5 | 39.2 | 34.2 |
| Liver disorders | 35.1 | 34.1 | 34.2 | 35.2 | 35.5 | 34.2 | 34.9 |
| Lung cancer | 47.1 | 57.5 | 47.1 | 50.0 | 50.4 | 54.6 | 47.1 |
| Lymphography | 16.1 | 21.9 | 16.5 | 18.2 | 17.8 | 19.5 | 16.1 |
| Pima | 25.2 | 25.0 | 25.4 | 25.3 | 25.4 | 28.1 | 25.4 |
| Postoperative | 33.9 | 30.0 | 37.8 | 33.9 | 31.6 | 37.8 | 33.9 |
| Primary tumor | 50.9 | 59.6 | 50.9 | 52.2 | 50.7 | 62.0 | 50.7 |
| Promoters | 9.3 | 17.5 | 10.8 | 11.2 | 8.9 | 45.7 | 9.3 |
| Solar flare | 19.4 | 15.6 | 16.9 | 15.9 | 15.9 | 16.8 | 16.4 |
| Sonar | 23.5 | 31.7 | 23.8 | 21.9 | 23.6 | 32.0 | 22.8 |
| Soybean | 9.2 | 8.5 | 6.7 | 8.2 | 9.0 | 10.0 | 5.9 |
| Splice junction | 4.4 | 5.8 | 4.5 | 4.5 | 4.2 | 30.2 | 4.0 |
| Tic-Tac-Toe | 30.6 | 13.7 | 15.7 | 21.8 | 28.0 | 4.5 | 13.5 |
| Wine | 2.2 | 12.1 | 2.5 | 3.6 | 3.4 | 18.8 | 2.2 |
| Zoology | 5.5 | 7.4 | 3.5 | 4.0 | 4.0 | 7.0 | 5.5 |
| average | 18.6 | 20.2 | 17.9 | 18.1 | 18.2 | 22.8 | 17.2 |

*Table 4.* Standard errors (%) of the error rates of NB, C4,5, BT, BSEJ, BSE, LAZYTREE, and LBR.

| Domain | NB | C4.5 | BT | BSEJ | BSE | LAZYTREE | LBR |
|---|---|---|---|---|---|---|---|
| Annealing | 0.4 | 0.7 | 0.5 | 0.2 | 0.4 | 0.4 | 0.4 |
| Audiology | 1.4 | 1.5 | 1.6 | 1.4 | 1.5 | 1.5 | 1.4 |
| Breast (W) | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.7 | 0.4 |
| Chess (KR-KP) | 0.4 | 0.1 | 0.3 | 0.3 | 0.4 | 0.2 | 0.1 |
| Credit (A) | 0.7 | 1.1 | 1.0 | 1.0 | 0.8 | 1.0 | 0.7 |
| Echocardiogram | 3.2 | 3.3 | 3.4 | 3.1 | 3.3 | 2.9 | 3.2 |
| Glass | 2.4 | 2.6 | 2.5 | 2.6 | 2.6 | 2.9 | 2.3 |
| Heart (C) | 1.2 | 1.6 | 1.3 | 1.3 | 1.3 | 1.4 | 1.2 |
| Hepatitis | 2.2 | 2.1 | 2.3 | 2.1 | 1.8 | 2.7 | 2.2 |
| Horse colic | 1.3 | 1.6 | 1.5 | 1.2 | 1.2 | 1.4 | 1.0 |
| House votes 84 | 0.9 | 0.8 | 0.7 | 0.9 | 0.8 | 0.9 | 0.8 |
| Hypothyroid | 0.1 | 0.2 | 0.1 | 0.1 | 0.1 | 0.2 | 0.1 |
| Iris | 1.0 | 1.0 | 1.5 | 1.1 | 1.1 | 1.1 | 1.0 |
| Labor | 2.5 | 3.5 | 2.7 | 2.8 | 2.7 | 4.1 | 2.5 |
| LED 24 | 2.6 | 2.4 | 2.6 | 2.7 | 2.6 | 2.2 | 2.6 |
| Liver disorders | 1.7 | 1.4 | 1.3 | 1.8 | 1.7 | 1.4 | 1.7 |
| Lung cancer | 6.5 | 6.2 | 6.5 | 6.7 | 7.0 | 3.5 | 6.5 |
| Lymphography | 2.9 | 2.4 | 2.5 | 2.5 | 2.5 | 2.1 | 2.9 |
| Pima | 1.2 | 0.9 | 1.0 | 1.2 | 1.1 | 1.4 | 1.2 |
| Postoperative | 2.5 | 1.4 | 3.0 | 2.2 | 1.9 | 2.7 | 2.5 |
| Primary tumor | 1.9 | 1.5 | 1.9 | 1.8 | 1.6 | 2.0 | 1.9 |
| Promoters | 2.0 | 2.6 | 2.2 | 1.6 | 1.6 | 2.0 | 2.0 |
| Solar flare | 0.5 | 0.2 | 0.4 | 0.5 | 0.4 | 0.4 | 0.6 |
| Sonar | 1.7 | 2.0 | 2.0 | 1.6 | 1.6 | 1.7 | 1.8 |
| Soybean | 0.8 | 0.5 | 0.6 | 0.7 | 0.7 | 0.7 | 0.5 |
| Splice junction | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 | 0.4 | 0.3 |
| Tic-Tac-Toe | 1.2 | 0.8 | 0.9 | 1.1 | 1.1 | 0.4 | 0.8 |
| Wine | 1.2 | 2.0 | 1.2 | 1.3 | 1.2 | 2.0 | 1.2 |
| Zoology | 1.5 | 1.7 | 1.5 | 1.3 | 1.1 | 1.5 | 1.5 |

*Table 5.* Error rate ratios: LBR vs NB, C4,5, BT, BSEJ, BSE, and LAZYTREE

| Domain | LBR | | | | | |
|---|---|---|---|---|---|---|
| (vs) | NB | C4.5 | BT | BSEJ | BSE | LAZYTREE |
| Annealing | .96 | **.28** | .87 | *2.70* | *1.23* | 1.04 |
| Audiology | .99 | 1.07 | .97 | 1.03 | 1.16 | **.76** |
| Breast (W) | 1.00 | **.57** | 1.00 | 1.00 | .96 | **.31** |
| Chess (KR-KP) | **.16** | *2.86* | 1.33 | **.74** | **.31** | 1.25 |
| Credit (A) | 1.00 | .97 | .94 | .94 | .96 | .88 |
| Echocardiogram | 1.04 | .89 | .96 | .99 | **.95** | .95 |
| Glass | .97 | .96 | 1.03 | .94 | .92 | **.80** |
| Heart (C) | 1.00 | **.78** | .93 | .98 | 1.00 | .86 |
| Hepatitis | 1.00 | .77 | **.82** | .98 | 1.10 | **.79** |
| Horse colic | .93 | 1.18 | 1.05 | .97 | .95 | 1.02 |
| House votes 84 | **.57** | 1.00 | 1.00 | **.67** | .76 | .92 |
| Hypothyroid | **.94** | 1.14 | 1.23 | *1.60* | 1.07 | **.55** |
| Iris | 1.00 | 1.11 | .82 | 1.05 | 1.19 | 1.00 |
| Labor | 1.00 | **.39** | **.72** | .92 | .84 | **.45** |
| LED 24 | 1.00 | .94 | .96 | **.92** | 1.02 | .87 |
| Liver disorders | .99 | 1.02 | 1.02 | .99 | .98 | 1.02 |
| Lung cancer | 1.00 | .82 | 1.00 | .94 | .93 | .86 |
| Lymphography | 1.00 | .74 | .98 | .88 | .90 | .83 |
| Pima | 1.01 | 1.02 | 1.00 | 1.00 | 1.00 | .90 |
| Postoperative | 1.00 | *1.13* | .90 | 1.00 | 1.07 | .90 |
| Primary tumor | 1.00 | **.85** | 1.00 | .97 | 1.00 | **.82** |
| Promoters | 1.00 | **.53** | .86 | .83 | 1.04 | **.20** |
| Solar flare | **.85** | 1.05 | .97 | 1.03 | 1.03 | .98 |
| Sonar | .97 | **.72** | .96 | 1.04 | .97 | **.71** |
| Soybean | **.64** | **.69** | .88 | **.72** | **.66** | **.59** |
| Splice junction | **.91** | **.69** | **.89** | **.89** | .95 | **.13** |
| Tic-Tac-Toe | **.44** | .99 | .86 | **.62** | **.48** | *3.00* |
| Wine | 1.00 | **.18** | .88 | **.61** | **.65** | **.12** |
| Zoology | 1.00 | .74 | *1.57* | 1.38 | 1.38 | .79 |
| average | .91 | .90 | .98 | .97 | .95 | .84 |
| w/t/l | 14/13/2 | 19/1/9 | 19/4/6 | 19/2/8 | 16/3/10 | 23/1/5 |
| $p$ of wtl | .0021 | .0436 | .0073 | .0261 | .1635 | .0005 |
| significant w/t/l | 7/22/0 | 10/17/2 | 3/25/1 | 7/20/2 | 5/23/1 | 12/16/1 |
| $p$ of sign. wtl | .0078 | .0193 | .3125 | .0898 | .1094 | .0017 |

(2) A one-tailed pairwise sign-test on the win/tie/loss records shows that LBR significantly outperforms NB, in terms of lower average error rate, at a significance level better than 0.05 across the 29 domains. This holds both using exact comparisons and when differences that are not statistically significant are treated as ties. LBR achieves an average relative error rate reduction of 9% over NB in the 29 domains.

Considering the performance in individual domains, LBR has lower error rates than NB in 14 out of the 29 domains, and higher error rates in only 2 domains. LBR has the same error rate as NB in 13 domains. For nine of these domains, LBR uses no conditions, and hence the original naive Bayesian classifier, to classify every test example. In the other four domains, the average number of conditions in a Bayesian rule is less 0.12. This means that LBR only builds very short rules for small numbers of test examples. As a result, LBR uses conventional naive Bayesian classification for almost all test cases in these 13 domains.

When only counting significant error differences, LBR is significantly more accurate than NB in 7 out of the 29 domains. In no domain, does LBR get significantly higher error rates than NB.

(3) LBR obtains lower error rates than C4.5 in 19 out of the 29 domains with 10 being significant and higher error rates in 9 domains with 2 being significant. These win/loss records (both exact, and counting non-significant differences as draws) are statistically significant at the 0.05 level using a one-tailed pairwise sign-test. The average relative error reduction of LBR over C4.5 in the 29 domains is 10%.

(5) The average relative error rate reductions of LBR over BT, BSEJ, BSE, and LAZYTREE in the 29 domains are 2%, 3%, 5%, and 16% respectively. LBR is more accurate than BT in 19 out of the 29 domains and less accurate in 6 domains. It is more accurate than BSEJ and BSE in 19 and 16 domains, as well as less accurate in 8 and 10 domains respectively. LBR has lower error rates than LAZYTREE in 23 domains and higher error rates in 5 domains. A one-tailed pairwise sign-test on these win/tie/loss records shows that LBR is significantly more accurate than BT, BSEJ, and LAZYTREE at a significance level better than 0.05 across these 29 domains, but fails to show that LBR is significantly more accurate than BSE at a level of 0.05. If insignificant error differences in individual domains are treated as ties, a one-tailed pairwise sign-test fails to show that LBR is significantly more accurate than BT, BSEJ, or BSE at a level of 0.05, but still shows that LBR is significantly more accurate than LAZYTREE at a level better than 0.05 across the 29 domains.

From the experiments, we also found that none of BT, BSEJ, and BSE has lower error rates than NB significantly more often than the reverse across the 29 domains, while LBR does. The average relative error rate reductions of BT, BSEJ, and BSE over NB in the 29 domains are 4%, 5%, and 4% respectively, lower than that for LBR. These results suggest that LBR outperforms BT, BSEJ, or BSE in terms

of improving upon the performance of the naive Bayesian classifier. LBR obtains higher error rates than NB in only 2 domains. This occurs in 16, 15, and 11 domains for BT, BSEJ, and BSE respectively. It is clear that compared with BT, BSEJ, and BSE, LBR less frequently obtains higher error than NB. In addition, LBR has never been found to significantly decrease the accuracy of the naive Bayesian classifier in the domains under study, while BT, BSEJ, and BSE do in some domains (2, 2, and 1 respectively).

It is noticed that BT does not have higher accuracies than C4.5 significantly more often than the reverse in the 29 domains. LAZYTREE even has lower accuracies than C4.5 significantly more often than the reverse.[7] These indicate that LBR performs better on average over the 29 domains than either BT or LAZYTREE for improving upon the accuracy of C4.5.

Compared to BSE, BSEJ is significantly more accurate in 4 out of the 29 domains and significantly less accurate in one domain. This suggests that joining attributes together with deleting attributes may have some advantage over just deleting attributes for naive Bayesian classification. However, the average error rates of BSEJ (18.1%) and BSE (18.2%) over the 29 domains are very similar.

Since C4.5 is good at dealing with numeric attributes, it might impede C4.5 to carry out discretization as a pre-process when running C4.5. In an additional experiment, C4.5 (release 8) was run without pre-discretization. The average error rate of C4.5 without pre-discretization in these 29 domains using the same training and test set partitions is 20.0%. This is slightly lower than that for C4.5 with pre-discretization. Comparing the error rates of LBR with those of C4.5 without pre-discretization in the 29 domains, the average error rate of LBR is 2.8 percentage points lower than that of C4.5; and the average relative error rate reduction of LBR over C4.5 is 5%. A one-tailed pairwise sign-test shows that LBR is superior to C4.5 without pre-discretization at a significance level of 0.0178 across the 29 domains in terms of lower error rate.

In summary, although it is known that no algorithm can have superior generalization performance to another across all possible domains (Wolpert, 1994; Schaffer, 1994; Rao, Gordon, & Spears, 1995), the experimental results show that LBR obtains lower error significantly more often than the reverse in comparison to NB, C4.5, BT, BSEJ, and LAZYTREE for the data sets studied. We believe these data sets to be broadly representative of those in the UCI repository. While the win/tie/loss outcomes against BSE were not statistically significant, LBR outperformed BSE with respect to all metrics considered—mean error, error rate ratio, and win/tie/loss record.

### *5.4. When does LBR outperform the naive Bayesian classifier?*

In the previous subsection, we compared the general performance of LBR against that of the other algorithms across the 29 domains. LBR obtained lower error than NB significantly more often than the reverse. However, for almost 50% of the domains, LBR did not improve upon NB's error rate. In this subsection, we analyze

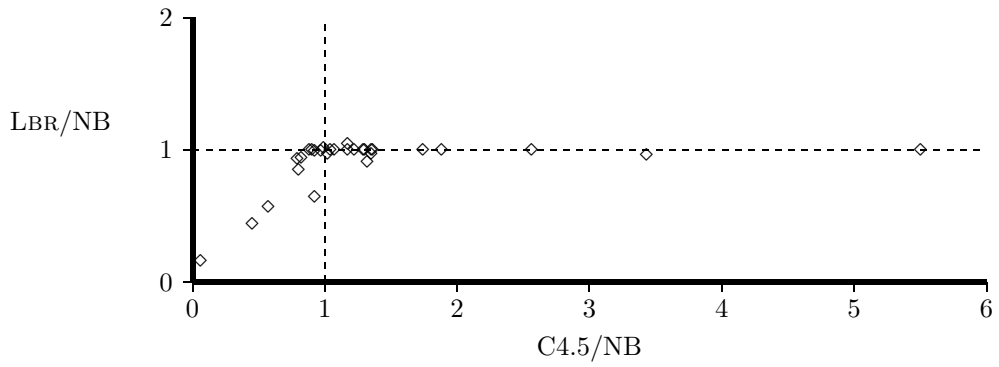


*Figure 4.* The error ratio of LBR over NB as a function of the error ratio of C4.5 over NB

in what situations LBR can be expected to reduce the error of the naive Bayesian classifier.

As specified before, LBR is intended to alleviate the attribute inter-dependence problem of naive Bayesian classification. To determine whether it is successful at this goal, we need to analyze whether LBR can outperform NB in domains where NB suffers from this problem. It is also interesting to consider how LBR performs relative to NB in other domains. Domingos and Pazzani (1996) have observed that the naive Bayesian classifier may be optimal when the independence assumption is violated but the ranks of the conditional probabilities of classes given an example are correct. In consequence, it is important to know not just whether the independence assumption is violated, but also to what extent the accuracy of the naive Bayesian classifier is affected by the attribute inter-dependence problem. Therefore, measures of attribute inter-dependencies are not appropriate indicators for this analysis. On the other hand, it is known that the naive Bayesian classifier is optimal when the independence assumption is not violated (assuming that the conditional probability estimates employed are accurate). As an approximation, if the naive Bayesian classifier performs worse than another learning algorithm in a domain, we can assume that the independence assumption is violated and the naive Bayesian classifier suffers from the attribute inter-dependence problem in this domain. Here, we choose C4.5, a well known decision tree learning algorithm, as the reference for our analysis. We use the relative performance of C4.5 vs NB as an approximate indicator of whether NB suffers from the attribute inter-dependence problem in a domain.

Figure 4 shows the error ratio of LBR over NB as a function of the error ratio of C4.5 over NB. Each point on the scatter graph represents one of the 29 domains. Points on the left of the vertical line at 1 indicate that C4.5 performs better than NB, that is, the independence assumption is violated and NB suffers from the attribute inter-dependence problem in these domains. Points below the horizontal line at 1 show that LBR outperforms NB in those domains.

From Figure 4, we can see a clear positive correlation between when NB suffers from the attribute inter-dependence problem and when LBR outperforms NB. C4.5 has lower error rates than NB in 12 out of the 29 domains. Among these 12 domains, LBR has lower error rates than NB in 9 domains, the same error rate in 2 domains, and a slightly higher error rate in one domain. The (product moment) correlation coefficient based on these 12 data points is 0.92, indicating a highly significant correlation (Chatfield, 1978). When considering only significant error differences, C4.5 is significantly more accurate than NB in 6 domains. Among these 6 domains, LBR is significantly more accurate than NB in 4 domains. The (product moment) correlation coefficient based on these 6 data points is 0.98, again indicating a highly significant correlation. The two exception domains are Horse colic and Postoperative. In the Horse colic domain, LBR is more accurate than NB at a significance level of 0.061 which is very close to the normally used significance level 0.05. In the Postoperative domain, the data set size (90) is too small for LBR to create Bayesian rules. It is notable that for six out of the ten domains with data set sizes smaller than 200, LBR did not create Bayesian rules, thus performing identically to NB. For the remaining four domains, Bayesian rules were created rarely. In no case were rules created for more than 11% of test examples. Hence, error was not greatly affected.

In addition, Figure 4 clearly illustrates that when C4.5 is as accurate, or less accurate than, NB (see the points on and on the right of the vertical line at 1), that is, when there is no evidence showing that NB suffers from the attribute inter-dependence problem, LBR has the same error rates as, or lower error rates than, NB. Only in one domain (Echocardiogram) among the domains of this type does LBR have a slightly higher error rate than NB. For example, it is known that in the LED24 domain, all attributes are independent from each other within each class (some attributes are irrelevant). In such a domain, NB should be optimal. The average length of Bayesian rules of LBR is 0. That is, LBR does not generate Bayesian rules in this domain. Instead, it just employs a naive Bayesian classifier using all attributes on the entire training set, thus having the same accuracy as NB.

In short, these results suggest that when NB suffers from the attribute inter-dependence problem, given that the training sets are not too small, LBR can effectively alleviate this problem, thus reducing error. When NB does not suffer from the attribute inter-dependence problem, LBR usually maintains NB's accuracy performance.

Next, we use the Tic-Tac-Toe domain as an example to analyze, in greater detail, the behavior of LBR compared with NB as well as other algorithms. The learning problem for Tic-Tac-Toe is to predict the class (either win for x or not) of an example that is described using 9 attributes corresponding to 9 squares in the Tic-Tac-Toe game board. The target concept of Tic-Tac-Toe is that the game is a win for x (the class $p$) if three squares in one line or one column or one diagonal are occupied by x. We know that attributes, especially the three attributes corresponding to squares that make the game a win for x or not a win for x, are not independent for a given class.

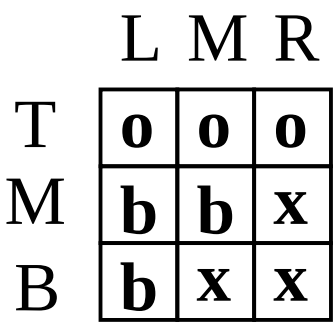


*Figure 5.* The Tic-Tac-Toe game board corresponding to the test example $\langle o, o, o, b, b, x, b, x, x\rangle$. The class of this example is $n$.

In this domain, NB has a significantly higher error rate than C4.5, indicating that NB suffers from the attribute inter-dependence problem. All of LBR, BT, BSEJ, and BSE significantly decrease the error rate of NB, with LBR achieving the lowest error rate among these four algorithms. LBR also achieves a lower error rate than C4.5. In this domain, BSEJ is significantly more accurate than BSE, because some attributes are highly inter-dependent but not completely redundant. It is worth mentioning that in this domain, LAZYTREE achieves a very low error rate. The reason is that when creating rules, LBR only considers attribute-value pairs each in the form of $(A = v)$ where $v$ is the test example's value, while LAZYTREE also allows attribute-value pairs each in the form of $(A \neq v')$ where $v'$ is a value of the attribute $A$ that is not equal to the test example's value. This is important for this domain. If LBR also considers "not equal", its error rate can be further reduced in this domain.

One test example, $\langle o, o, o, b, b, x, b, x, x\rangle$, at one trial is chosen at random for examination in the Tic-Tac-Toe domain. The game board of this test example is shown in Figure 5. This game is not a win for x, that is, the class of the example is $n$.

Given $V = \langle o, o, o, b, b, x, b, x, x\rangle$, NB produces $P(p \mid V) \propto P(p) \times \prod_j P(v_j \mid p) = 0.00001413$ and $P(n \mid V) \propto P(n) \times \prod_j P(v_j \mid n) = 0.00000996$. Therefore, it incorrectly classifies the test example as class $p$. To classify this test example, LBR builds a Bayesian rule with (T-L = o) AND (T-M = o) as its antecedent. The local naive Bayesian classifier of this Bayesian rule is trained with all attributes except T-L and T-M on those training examples that satisfy (T-L = o) AND (T-M = o). Using this local naive Bayesian classifier, LBR gives $P(p \mid V) \propto 0.00000263$ and $P(n \mid V) \propto 0.00012125$, thus predicting that the test example belongs to class $n$. If, instead of using a Bayesian rule, the attributes T-L and T-M are deleted, the naive Bayesian classifier trained on all training examples using all other attributes produces $P(p \mid V) \propto 0.00013110$ and $P(n \mid V) \propto 0.00007527$. Consequently, unlike LBR, it incorrectly predicts $p$ as the class of the test example.

In this example, NB cannot correctly classify this test example due to the high inter-dependencies among the attributes within each class. LBR makes a correct classification by successfully creating a Bayesian rule, removing two damaging inter-dependent attributes from the local naive Bayesian classifier and using them to define an instance sub-space for the local naive Bayesian classifier. Simply deleting these two damaging inter-dependent attributes cannot make the naive Bayesian classifier correctly classify this test example as such deletion removes critical information.

### *5.5. LBR's performance relative to BT for avoiding the small disjunct problem*

In Subsection 4.2, we argue that LBR can avoid the small disjunct problem. This subsection provides experimental evidence in support of this claim. We use two measures: the average local training set size and the minimum local training set size. When BT and LBR classify a test example during a trial on a domain, we count the local training examples used to train the local naive Bayesian classifier at the leaf used by BT and those in the consequent of the Bayesian rule that LBR constructs. The average number of local training examples for all test examples is the average local training set size at this trial. The minimum local training set size at this trial is the minimum number of local training examples over all the test examples. We report the average value over the 20 trials in a domain for each of these two measures.

Across the 29 domains, the average value of the average local training set size for LBR is 414.7, while it is 290.5 for BT. The average ratio of this measure of LBR over BT in the 29 domains is 1.97. LBR has significantly larger average local training set sizes than BT in 22 out of the 29 domains. Only in one domain does LBR have a significantly smaller average local training set size than BT. A one-tailed pairwise sign-test on the significant win/tie/loss records shows that LBR is significantly superior to BT in terms of larger average local training set size across the 29 domains at a significance level better than 0.0001.

It is also interesting to consider the smallest training set size employed by each algorithm during each test. The average value of the minimum local training set size for LBR is 238.6, while it is 193.4 for BT in the 29 domains. The average ratio of this measure of LBR over BT in the 29 domains is 7.97. LBR has significantly larger minimum local training set sizes than BT in 21 out of the 29 domains, and a significantly smaller minimum local training set size than BT in one domain. A one-tailed pairwise sign-test on the significant win/tie/loss records shows that LBR has significant advantage over BT in terms of larger minimum local training set size across the 29 domains at a significance level better than 0.0001.

The only domain where LBR has a significantly smaller average local training set size and a significantly smaller minimum local training set size than BT is Splice junction. These two measures are 1991.1 and 1251.2 respectively for LBR. They are 2511.2 and 2335.8 respectively for BT. Given the magnitude of these values, it does not appear that LBR is suffering from the small disjunct problem in this domain. Rather, these values suggest that LBR can make better use of the large training

set in this domain than BT to identify and eliminate damaging inter-dependent attributes for naive Bayesian classification. In the Splice junction domain, the average size of the antecedents of Bayesian rules for LBR is 0.7, while the average path length used for classifying the test examples for BT is 0.2. The latter is much shorter than the former. This contributes to the significantly lower error rate of LBR than BT in this domain.

The smallest mean minimum local training set size over the 29 domains for BT is only 0.5 in the Soybean domain, where the minimum local training set size of LBR is 56.6. The average local training set size in the Soybean domain for BT is 230.2, much smaller than that for LBR, 431.0. The smallest minimum local training set size over the 29 domains for LBR is 22.2 in the Tic-Tac-Toe domain, where the minimum local training set size for BT is 3.6. The latter is much smaller than the former. The average local training set size in the Tic-Tac-Toe domain for LBR is 136.2, much larger than that for BT, 37.9. In both of these two domains, LBR has lower error rates than BT.

In summary, these experimental results suggest that BT suffers from the small disjunct problem in some domains, while LBR performs much better than BT in this respect. It seems likely that the ability of LBR to avoid the small disjunct problem contributes to the higher average accuracy of LBR over BT.

### *5.6. Computational requirements*

Lazy learning defers computation until classification time. The manner in which lazy learning delays computation contrasts with non-lazy learners that perform most computation during training and have low computational requirements for classification. Typically, constructing a single theory for non-lazy learning will require more computation than lazy classification of a single case. The relative desirability of these two types of computational profile will depend upon the operational context. If only a few classifications are performed for each training set (for example, if the training set is regularly updated), then lazy learning is likely to have an advantage. For any given domain and training set there will be a test set size at which the total computation cost of a lazy learning system overtakes that of a non-lazy system. LBR's classification process involves non-trivial computation. In consequence, where a large number of cases are to be classified, LBR will incur large computational overheads. These overheads can be reduced somewhat by caching useful information from one classification to the rest.

To give an idea of the computational requirements of LBR, Table 6 shows the execution time of LBR in CPU seconds on a SUN UltraSPARC 2 server for a single trial from the type of experiment reported before. The execution times of other comparison algorithms are also included in the table as references. The execution time of an algorithm in a domain reported in this subsection is the time used for both training on the training set and classifying all test examples in the test set at the first trial of a 10-fold cross-validation. That is, 90% of the data is used for training and 10% for classification. It is worth mentioning that none of the implementations of NB, BT, BSEJ, BSE, LAZYTREE, and LBR are optimized with

*Table 6.* Computational requirements in CPU seconds of NB, C4,5, BT, BSEJ, BSE, LAZYTREE, and LBR. Times are rounded to the closest tenth of a second.

| Domain | NB | C4.5 | BT | BSEJ | BSE | LAZYTREE | LBR |
|---|---|---|---|---|---|---|---|
| Annealing | 0.0 | 0.1 | 49.2 | 838.2 | 8.5 | 0.6 | 1.0 |
| Audiology | 0.0 | 0.1 | 116.5 | 12188.4 | 94.0 | 0.3 | 11.4 |
| Breast (W) | 0.0 | 0.0 | 0.1 | 0.5 | 0.1 | 8.4 | 0.1 |
| Chess (KR-KP) | 0.1 | 0.3 | 40.6 | 1719.6 | 43.9 | 15.6 | 146.5 |
| Credit (A) | 0.0 | 0.0 | 0.3 | 4.5 | 0.5 | 0.6 | 0.3 |
| Echocardiogram | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Glass | 0.0 | 0.0 | 0.1 | 0.9 | 0.1 | 0.3 | 0.0 |
| Heart (C) | 0.0 | 0.0 | 0.1 | 1.6 | 0.0 | 0.0 | 0.1 |
| Hepatitis | 0.0 | 0.0 | 0.1 | 2.8 | 0.2 | 0.0 | 0.0 |
| Horse colic | 0.0 | 0.0 | 1.2 | 21.2 | 0.5 | 0.1 | 0.6 |
| House votes 84 | 0.0 | 0.0 | 0.8 | 5.7 | 0.7 | 0.2 | 0.5 |
| Hypothyroid | 0.1 | 0.2 | 18.6 | 230.3 | 6.9 | 17.8 | 31.2 |
| Iris | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Labor | 0.0 | 0.0 | 0.0 | 0.5 | 0.0 | 0.0 | 0.0 |
| LED 24 | 0.0 | 0.0 | 1.0 | 55.5 | 2.6 | 0.1 | 0.5 |
| Liver disorders | 0.0 | 0.0 | 0.0 | 0.1 | 0.0 | 0.0 | 0.0 |
| Lung cancer | 0.0 | 0.0 | 0.0 | 62.5 | 1.0 | 0.0 | 0.2 |
| Lymphography | 0.0 | 0.0 | 0.2 | 6.6 | 0.3 | 0.0 | 0.1 |
| Pima | 0.0 | 0.0 | 0.1 | 0.8 | 0.1 | 0.5 | 0.1 |
| Postoperative | 0.0 | 0.0 | 0.0 | 0.1 | 0.0 | 0.0 | 0.0 |
| Primary tumor | 0.0 | 0.1 | 2.1 | 51.8 | 2.2 | 0.2 | 1.0 |
| Promoters | 0.0 | 0.0 | 0.9 | 172.8 | 1.0 | 0.0 | 0.1 |
| Solar flare | 0.0 | 0.1 | 0.6 | 4.1 | 0.6 | 1.9 | 1.8 |
| Sonar | 0.0 | 0.0 | 7.6 | 581.5 | 1.8 | 0.1 | 0.8 |
| Soybean | 0.0 | 0.2 | 44.3 | 1211.2 | 37.5 | 9.3 | 24.9 |
| Splice junction | 0.4 | 0.8 | 39.6 | 11694.5 | 118.0 | 761.5 | 103.2 |
| Tic-Tac-Toe | 0.0 | 0.0 | 0.5 | 1.3 | 0.2 | 0.6 | 0.9 |
| Wine | 0.0 | 0.0 | 0.1 | 0.5 | 0.0 | 0.1 | 0.0 |
| Zoology | 0.0 | 0.0 | 0.4 | 3.6 | 0.2 | 0.0 | 0.0 |
| average | 0.0 | 0.1 | 11.2 | 995.2 | 11.1 | 27.1 | 11.2 |

respect to computational efficiency, since this issue is not the focus of our current research. As a result, these timing results should be taken as broadly indicative only.

When collecting these timing results, only one trial for each algorithm in each domain was conducted in order to minimize the overall computational requirements of the experiment. Note that timing results for the experiments presented in Subsection 5.3 are not useful, as the algorithms were run on four different types of workstation, making computation times incommensurable.

From Table 6, we can see that BT, BSEJ, BSE, LAZYTREE, and LBR are slower than NB and C4.5. However, the execution time for LBR is reasonably acceptable, given that it is very close to that of BT even though LBR uses lazy learning. The two worst cases for LBR are 146.6 seconds and 103.2 seconds in the Chess (KR-KP) and Splice junction domains. These two domains have 3169 and 3177 examples respectively. They have 73 and 240 different attribute values respectively.

In the current implementation of LBR, a simple caching technique was incorporated. The evaluation function values of attribute-value pairs that have been examined are retained from one test item to the next. This can avoid the re-calculation of evaluation function values of some attribute-value pairs when classifying unseen examples that appear later, thus reducing the entire execution time of LBR. Our experiment shows that caching this information reduces the execution time of LBR by 96% on average in these 29 domains. This happens, because the evaluations of attribute-value pairs for different test examples are often repeated, including repeated generation of identical Bayesian rules for different test examples. LBR could be made even more efficient by caching further information such as local naive Bayesian classifiers and indices for training examples at different stages of the growth of Bayesian rules. Of course, this would increase memory requirements.

It is noticeable that BSEJ requires long execution time in domains with a large number of attribute values. BSE is faster than BSEJ, since the search space of BSE is much smaller than that of BSEJ. The number of attribute values of a domain also significantly affects the execution time of LBR, since it determines the number of possible attribute-value pairs that need to be evaluated. However, the length of antecedents of Bayesian rules to be generated in a domain is a more important factor for LBR. For example, longer Bayesian rules are created by LBR in the Chess (KR-KP) domain than in the Splice junction domain. LBR uses more time in the former than in the latter, although the latter has more attribute values than the former.

## 6. Conclusions and future work

We have proposed a novel lazy Bayesian rule learning algorithm, LBR. It builds for each test example a most appropriate Bayesian rule with a local naive Bayesian classifier as its consequent. Compared with other existing approaches to improving the naive Bayesian classifier, decision tree learning, and rule learning, the novelty of LBR is that it generates Bayesian rules using lazy learning. The antecedent of a Bayesian rule defines an instance sub-space for its local naive Bayesian classifier.

We explained that, by doing so, damaging attribute inter-dependencies for naive Bayesian classification can be factored out. In addition, the small disjunct problem can be avoided. During the growth of a Bayesian rule, LBR uses leave-1-out cross-validation and a significance test to manage the trade-off between the degree to which the attribute independence assumption of the naive Bayesian classifier is violated and the number of training examples available for training the local naive Bayesian classifier.

LBR uses lazy learning. Although it does not create a single theory for a learning problem, unlike some other lazy learning algorithms such as instance based learning, LBR still generates a Bayesian rule with a conjunction of attribute-value pairs as the antecedent and a local naive Bayesian classifier as the consequent for each test example. Both of these are comprehensible. LBR is further distinguished from previous lazy learners by the generic strategy used to select the training cases with which a test case is to be classified. Whereas previous algorithms have sought to identify and include relevant training cases without specific reference to the classification technique which will utilize them, LBR seeks to exclude only those training cases for which there is clear evidence that the classification method (a naive Bayesian classifier) will suffer from their inclusion. In addition, LBR has a natural method for dealing with missing values of test examples, not considering attributes with missing values on test examples when generating antecedents of Bayesian rules.

When generating the antecedent of a Bayesian rule, the current implementation of LBR only considers attribute-value pairs each in the form of $(A = v)$, where $v$ is equal to the test example's value. An alternative is considering $(A \neq v)$, where $v$ is a value of $A$ that is not equal to the test example's value. A further step is exploring $(A \in v_{subset})$, where $v_{subset}$ is a subset of values of $A$ to which the test example's value belongs. Currently, for numeric attributes, discretization is carried out as a pre-process of LBR. An alternative is the direct use of numeric attributes when generating rules. The current implementation of LBR with a simple caching technique provides reasonable computational performance. Caching more information could make it more efficient.

LBR has been compared experimentally with a naive Bayesian classifier, a state-of-the-art decision tree learner, a Bayesian tree learning algorithm, a constructive Bayesian classifier, a selective naive Bayesian classifier (i.e., BSE), and a lazy decision tree learning algorithm in a wide variety of natural domains. This extensive experimental comparison of key existing techniques for improving the naive Bayesian classifier is another contribution of this paper. In our experiments, LBR obtained lower error than the alternative algorithm significantly more often than the reverse against all the other algorithms but one. The one algorithm against which LBR did not achieve a significant advantage on this metric is the selective naive Bayesian classifier (i.e., BSE). However, LBR still achieved lower error more often than the reverse in comparison to this algorithm. Further, LBR achieved lower mean error and mean error ratios against all alternatives across the 29 domains in the study. LBR has been demonstrated to alleviate the attribute inter-dependence problem of naive Bayesian classification, although it rarely alters the error performance of

the naive Bayesian classifier when applied to small training sets. Investigation of appropriate extensions of the technique for small datasets remains an interesting direction for future research. Our experimental results indicate that in typical domains where the error performance of the naive Bayesian classifier does not suffer from the attribute inter-dependence problem, LBR usually maintains the naive Bayesian classifier's level of prediction accuracy. All these results suggest that LBR provides a very competitive learning technique where error minimization is an important criterion.

## Acknowledgments

The authors are grateful to J. Ross Quinlan for providing C4.5, and Kai Ming Ting for supplying the discretization program. Thanks to Douglas Newlands for his helpful comments on earlier drafts of the paper. This paper has benefited greatly from the anonymous reviewers' suggestions and Douglas Fisher's suggestions and editorial comments.

## Notes

1. While it would also be difficult to estimate $P(V)$ accurately, $P(C_i) \times P(V \mid C_i) \propto P(C_i) \times P(V \mid C_i)/P(V)$, and hence it is not necessary to estimate $P(V)$.
2. We, here, use $v_j \in V$ to denote that $v_j$ appears in the conjunction $V$.
3. We found, in our experiments, that the error rate of the naive Bayesian classifier can be reduced in about a half of the domains by using techniques for alleviating the attribute inter-dependence problem.
4. These are referred to as paths by Friedman *et al.* (1996).
5. Since different attribute-value pairs cover different subsets of the local training examples, estimated errors for different attribute-value pairs on their corresponding subsets of training examples are not comparable. We assume that each temporary naive Bayesian classifier has the same errors as the local naive Bayesian classifier of the current Bayesian rule on those local training examples that are not covered by the corresponding attribute-value pair. In this manner, we can measure each attribute-value pair on the whole local training set.
6. The sign-test is a distribution free test for testing the null hypothesis that, for example, it occurs by chance that one method gives higher results than the other method. It is appropriate for the situation where no assumption can be made about the distribution of the observations (Chatfield, 1978). Note that the t-test assumes that the observations are normally distributed. Since we cannot assume that the error rates of an algorithm across domains are normally distributed, and indeed, there is cause to doubt whether error rates in two different domains are even commensurable, we use the sign-test to compare the performance of two algorithms across the 29 domains. In most cases, we apply the sign-test to the win/tie/loss records. Ties are disregarded. The resulting $p$ value is the probability that the observed number of wins or greater would occur if wins and losses were equiprobable random events. As this test considers the possibility of some wins or losses occurring by chance, there is no need to pre-filter the win/tie/loss records, for example by applying a t-test.
7. LAZYDT was reported to have an error rate 1.9 percentage points lower than C4.5 on average in a set of domains (Friedman *et al.*, 1996). Some of these domains are different from those used in this paper. However, 15 domains are used in both of these two studies. The average error rates of LAZYTREE and those reported for LAZYDT in these 15 domains are very similar. They are 15.8% and 15.6% respectively. This suggests that the performance of LAZYTREE, our implementation of LAZYDT, is very close to that of LAZYDT in terms of prediction accuracy.

## References


Aha, D.W., Kibler, D., & Albert, M.K. (1991). Instance-based learning algorithms. *Machine Learning, 6*, 37-66.

Aha, D.W. (ed.), (1997). *Lazy Learning*. Dordrecht: Kluwer Academic.

Blake, C., Keogh, E., & Merz, C.J. (1998). UCI Repository of Machine Learning Databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Breiman, L., Friedman, J.H., Olshen, R.A., & Stone, C.J. (1984). *Classification And Regression Trees*. Belmont, CA: Wadsworth.

Briand, L.C., & Thomas, W.M. (1992). A pattern recognition approach for software engineering data analysis. *IEEE Transactions on Software Engineering, 18*, 931-942.

Cestnik, B. (1990). Estimating probabilities: A crucial task in machine learning. *Proceedings of the European Conference on Artificial Intelligence* (pp. 147-149).

Cestnik, B., Kononenko, I., & Bratko, I. (1987). ASSISTANT 86: A knowledge-elicitation tool for sophisticated users. In I. Bratko & N. Lavrač (Eds.), *Progress in Machine Learning – Proceedings of the Second European Working Session on Learning (EWSL87)* (pp. 31-45). Wilmslow, UK: Sigma Press.

Chatfield, C. (1978). *Statistics for Technology: A Course in Applied Statistics*. London: Chapman and Hall.

Chow, C.K., & Liu, C.N. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory, 14*, 462-467.

Cover, T.M., & Hart, P.E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*, 21-27.

Dasarathy, B.V. (1980). Noising around the neighborhood: A new system structure and classification rule for recognition in partially exposed environments. *Pattern Analysis and Machine Intelligence, 2*, 67-71.

Domingos, P., & Pazzani, M. (1996). Beyond independence: Conditions for the optimality of the simple Bayesian classifier. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 105-112). San Francisco, CA: Morgan Kaufmann.

Duda, R.O., & Hart, P.E. (1973). *Pattern Classification and Scene Analysis*. New York: John Wiley.

Fayyad, U.M., & Irani, K.B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (pp. 1022-1027). San Mateo, CA: Morgan Kaufmann.

Friedman, J., Kohavi, R., & Yun, Y. (1996). Lazy decision trees. *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (pp. 717-724). Menlo Park, CA: The AAAI Press.

Friedman, N., & Goldszmidt, M. (1996). Building classifiers using Bayesian networks. *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (pp. 1277-1284). Menlo Park, CA: The AAAI Press.

Fulton, T., Kasif, S., Salzberg, S., & Waltz, D. (1996). Local induction of decision trees: Towards interactive data mining. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (pp. 14-19). Menlo Park, CA: AAAI Press.

Gates, G.W. (1972). The reduced nearest neighbor rule. *IEEE Transactions on Information Theory, 18*, 431-433.

Geiger, D. (1992). An entropy-based learning algorithm of Bayesian conditional trees. *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence* (pp. 92-97). Morgan Kaufmann.

Hart, P.E. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory, 14*, 515-516.

Holte, R.C., Acker, L.E., & Porter, B.W. (1989). Concept learning and the problem of small disjuncts. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 813-818). San Mateo, CA: Morgan Kaufmann.

John, G.H., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 121-129). San Francisco, CA: Morgan Kaufmann.

Kittler, J. (1986). Feature selection and extraction. In T.Y. Young & K. Fu (Eds.), *Handbook of Pattern Recognition and Image Processing* (pp. 59-81). San Diego, CA: Academic Press.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1137-1143). San Mateo, CA: Morgan Kaufmann.

Kohavi, R. (1996). Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (pp. 202-207). Menlo Park, CA: The AAAI Press.

Kononenko, I. (1990). Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition. In B. Wielinga *et al.* (eds.), *Current Trends in Knowledge Acquisition*. Amsterdam: IOS Press.

Kononenko, I. (1991). Semi-naive Bayesian classifier. *Proceedings of European Conference on Artificial Intelligence* (pp. 206-219).

Kononenko, I. (1993). Inductive and Bayesian learning in medical diagnosis. *Applied Artificial Intelligence, 7*, 317-337.

Kubat, M., Flotzinger, D., & Pfurtscheller, G. (1993). Discovering patterns in EEG-signals: Comparative study of a few methods. *Proceedings of European Conference on Machine Learning* (pp. 366-371). Berlin: Springer-Verlag.

Langley, P., Iba, W.F., & Thompson, K. (1992). An analysis of Bayesian classifiers. *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 223-228). Menlo Park, CA: The AAAI Press.

Langley, P. (1993). Induction of recursive Bayesian classifiers. *Proceedings of the European Conference on Machine Learning* (pp. 153-164). Berlin: Springer-Verlag.

Langley, P., & Sage, S. (1994). Induction of selective Bayesian classifiers. *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence* (pp. 339-406). Seattle, WA: Morgan Kaufmann.

Pazzani, M.J. (1996). Constructive induction of Cartesian product attributes. *Proceedings of the Conference, ISIS'96: Information, Statistics and Induction in Science* (pp. 66-77). Singapore: World Scientific.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Quinlan, J.R. (1996). Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research, 4*, 77-90.

Rao, R.B., Gordon, D., & Spears, W. (1995). For every generalization action, is there really an equal and opposite reaction? Analysis of the conservation law for generalization performance. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 471-479). San Francisco, CA: Morgan Kaufmann.

Sahami, M. (1996). Learning limited dependence Bayesian classifiers. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (pp. 334-338). Menlo Park, CA: The AAAI Press.

Schaffer, C. (1994). A conservation law for generalization performance. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 259-265). San Francisco, CA: Morgan Kaufmann.

Singh, M., & Provan, G.M. (1995). A comparison of induction algorithms for selective and non-selective Bayesian classifiers. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 497-505). San Francisco, CA: Morgan Kaufmann.

Singh, M., & Provan, G.M. (1996). Efficient learning of selective Bayesian network classifiers. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 453-461). San Francisco, CA: Morgan Kaufmann.

Ting, K.M. (1994a) The problem of small disjuncts: Its remedy in decision trees. *Proceedings of the Tenth Canadian Conference on Artificial Intelligence* (pp. 91-97). Canadian Society for Computational studies of Intelligence.

Ting, K.M. (1994b). Discretization of continuous-valued attributes and instance-based learning (Technical Report 491). Sydney, Australia: University of Sydney, Basser Department of Computer Science.

Viswanathan, M., & Webb, G.I. (1998). Classification learning using all rules. *Proceedings of the Tenth European Conference on Machine Learning* (pp. 149-159). Berlin: Springer-Verlag.

Webb, G.I., & Pazzani, M.J. (1998). Adjusted probability naive Bayesian induction. *Proceedings of the Eleventh Australian Joint Conference on Artificial Intelligence* (pp. 285-295). Berlin: Springer-Verlag.

Webb, G.I. (1996). A heuristic covering algorithm outperforms learning all rules. *Proceedings of the Conference, ISIS'96: Information, Statistics and Induction in Science* (pp. 20-30). Singapore: World Scientific.

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