Computational Intelligence, Volume 19, Number 2, 2003

A STUDY OF AdaBoost WITH NAIVE BAYESIAN CLASSIFIERS: WEAKNESS AND IMPROVEMENT

KAI MING TING

Gippsland School of Computing and Information Technology, Monash University, Australia

ZIJIAN ZHENG

Microsoft Corporation, Redmond, WA

This article investigates boosting naive Bayesian classification. It first shows that boosting does not improve the accuracy of the naive Bayesian classifier as much as we expected in a set of natural domains. By analyzing the reason for boosting's weakness, we propose to introduce tree structures into naive Bayesian classification to improve the performance of boosting when working with naive Bayesian classification. The experimental results show that although introducing tree structures into naive Bayesian classification increases the average error of the naive Bayesian classification for individual models, boosting naive Bayesian classifiers with tree structures can achieve significantly lower average error than both the naive Bayesian classifier and boosting the naive Bayesian classifier, providing a method of successfully applying the boosting technique to naive Bayesian classification. A bias and variance analysis confirms our expectation that the naive Bayesian classifier is a stable classifier with low variance and high bias. We show that the boosted naive Bayesian classifier has a strong bias on a linear form, exactly the same as its base learner. Introducing tree structures reduces the bias and increases the variance, and this allows boosting to gain advantage.

Key words: bias and variance analysis, boosting, naive Bayesian classification.

1. INTRODUCTION

Recent studies on the boosting technique have brought great success to the increasing prediction accuracy of decision tree learning algorithms (Breiman 1996; Freund and Schapire 1996; Quinlan 1996; Schapire et al. 1997; Bauer and Kohavi 1999; Friedman, Hastie, and Tibshirani 2000).

Bauer and Kohavi (1999) report that boosting for decision tree learning achieves a relative error reduction of 27% in a set of 14 domains. They also report that boosting the naive Bayesian classifier reduces the average relative error by 24% in the same set of domains.

In sharp contrast to Bauer and Kohavi's (1999) result on boosting the naive Bayesian classifier, our experiments show that boosting gets only 4% of average relative error reduction over the naive Bayesian classifier in a set of 25 domains from the University of California, Irvine (UCI) repository of machine learning databases (Blake, Keogh, and Merz 1998). In quite a number of domains, boosting cannot increase, or even decrease, the accuracy of the naive Bayesian classifier. We are interested in investigating the reason why boosting performs poorly with naive Bayesian classifiers and the ways to improve its performance.

One key difference between naive Bayesian classifiers and decision trees, with respect to boosting, is the stability of the classifiers—naive Bayesian classifier learning is relatively stable with respect to small changes in training data, but decision tree learning is not. Combining these two techniques will result in a learning method that produces more unstable classifiers than naive Bayesian classifiers. We expect it will improve the performance of boosting for naive Bayesian classification.

Using the results of previous study on the boosted naive Bayesian classifier (Elkan 1997; Ridgeway, Madigan, Richardson, and O'Kane 1998), we further show that the boosted naive Bayesian classifier is exactly the linear form in the log-odd scale of a naive Bayesian classifier. This strong and uncorrectable bias prevents performance improvement through boosting, which is basically a bias reduction method (Friedman et al. 2000). Tree structure transforms the final model into an interaction of attributes, not in the original linear form, reducing the bias and increasing the variance, which allows boosting to further reduce both the bias and the variance.

The hybrid approach generates naive Bayesian trees. Kohavi (1996) proposes such a hybrid approach, called NBTree. The purpose is to improve the accuracy of the naive Bayesian classifier by alleviating the attribute interdependence problem of naive Bayesian classification.

Here, we want to explore whether and how the introduction of tree structures into naive Bayesian classification affects the performance of boosting for naive Bayesian classification. For this purpose, we use leveled naive Bayesian trees, which generate tree structures to a predefined maximum depth. This allows us to investigate the effect in a systematic way.

The following section briefly presents the boosting technique including the details in our implementation. Section 3 describes the naive Bayesian classifier learning algorithm and the leveled naive Bayesian tree learning algorithm that are used as the base learners for boosting in this study. Section 4 empirically studies the effects of introducing tree structures into naive Bayesian classification on the performance of boosting for naive Bayesian classification. A bias and variance analysis is also conducted to investigate the variation of these error components for the naive Bayesian classifiers, leveled naive Bayesian trees, and their boosted counterparts. We provide a possible explanation for the discrepancy between Bauer and Kohavi's (1999) result and ours in Section 5. We provide an analysis of the boosted naive Bayesian classifier in the log-odd scale and explain the effect of introducing tree structure in terms of reducing the bias of the naive Bayesian classifier in Section 6, and we conclude in the final section.

2. BOOSTING

Boosting induces multiple classifiers in sequential trials by adaptively changing the distribution of the training set based on the performance of previously created classifiers. At the end of each trial, instance weights are adjusted to reflect the importance of each training example for the next induction trial. The objective of the adjustment is to increase the weights of misclassified training examples. Changing the instance weights causes the learner to concentrate on different training examples in different trials, thus resulting in different classifiers. Finally, the individual classifiers are combined through weighted voting to form a composite classifier.

In our implementation, boosting uses reweighting rather than resampling (Freund and Schapire 1996; Quinlan 1996). We concentrate on AdaBoost.M1 (Freund and Schapire 1996), which has been shown to work well for decision trees in both the two-class and multiclass problems (Bauer and Kohavi 1999). The weight adjustment formulas in step (iii) below are mathematically equivalent to those of AdaBoost.M1 (see Bauer and Kohavi 1999). They are used because they are simpler to implement. The boosting procedure is as follows:

Boosting procedure: Given a base learner and a training set $\mathcal{S}$ containing N examples, $w_t(n)$ denotes the weight of the nth example at the tth trial, where $w_1(n) = 1$ for every n. At each trial $t = 1, \ldots, T$, the following three steps are carried out:

(i) A model M_t is constructed using the base learner from $\mathcal{S}$ under the weight distribution w_t.

(ii) $\mathcal{S}$ is classified using the model M_t. Let $\delta(n) = 1$ if the nth example in $\mathcal{S}$ is classified incorrectly; $\delta(n) = 0$ otherwise. The error rate of this model, ϵ_t, is defined as

$$\epsilon_t = \frac{1}{N} \sum_n w_t(n)\delta(n). \tag{1}$$

If $\epsilon_t \geq 0.5$ or $\epsilon_t = 0$, then $w_t(n)$ is reinitialized using bootstrap sampling from $\mathcal{S}$ with weight 1 for each sampled example, and the boosting process continues from step (i).

(iii) The weight vector $w_{(t+1)}$ for the next trial is created from w_t as follows:
If the nth example is classified incorrectly by M_t,

$$w_{(t+1)}(n) = \frac{w_t(n)}{2\epsilon_t}; \tag{2}$$

otherwise

$$w_{(t+1)}(n) = \frac{w_t(n)}{2(1-\epsilon_t)}. \tag{3}$$

If $w_{(t+1)}(n) < 10^{-8}$, set it to 10^{-8} to solve the numerical underflow problem (Bauer and Kohavi 1999). These weights are then renormalized so that they sum to N.

After T trials, the models $M_1, \ldots, M_T$ are combined to form a single composite classifier. Given an example, the final classification of the composite classifier relies on the votes of all the individual models. The vote of the model M_t is worth $\log(\frac{1-\epsilon_t}{\epsilon_t})$ units. The voting scheme is simply summing up the vote for the predicted class of every individual model.

This implementation of the boosting procedure is very similar to that of Bauer and Kohavi (1999). The only difference is that whereas they halt induction and treat M_t as having infinite weight if $\epsilon_t = 0$, our implementation sets the vote for M_t to $\log(10^{10})$ when $\epsilon_t = 0$, and reinitializes the weights using bootstrap sampling to continue the boosting process. Some evidence exists that reinitialization works well in practice (Bauer and Kohavi 1999). We implement reinitialization to investigate the benefit, in addition to the following consideration in comparing the two methods.

We are of the view that the comparison is fairer with reinitialization because it ensures that the same number of models is used in the boosting procedure. If no reinitialization is used, in the extreme case, we could compare an ensemble of 2 models with another ensemble of 100 models, because they stop at hugely different trials. As we will show in this article, reinitialization either improves or maintains the level of accuracy. Reinitialization helps to ensure that the difference in performance is solely due to the base learning algorithm. Otherwise, one would have an added concern whether the difference is due to comparing a simple ensemble model with a complex ensemble model.

3. NAIVE BAYESIAN CLASSIFIER LEARNING AND LEVELED NAIVE BAYESIAN TREE LEARNING

In this section, we describe two base learning algorithms. One is the naive Bayesian classifier (Duda and Hart 1973; Kononenko 1990; Langley, Iba, and Thompson 1992; Domingos and Pazzani 1996). We will show in Section 4.1 that boosting does not significantly improve the performance of the naive Bayesian classifier. The other base learning algorithm is for generating leveled naive Bayesian trees. The objective of this algorithm is to investigate whether and how we can make boosting perform better for naive Bayesian classification.

3.1. Naive Bayesian Classifiers

The Bayesian approach to classification estimates the (posterior) probability that an instance belongs to a class, given the observed attribute values for the instance. When making

a categorical rather than probabilistic classification, the class with the highest estimated posterior probability is selected.

The posterior probability, $P(c \mid v)$, of an instance being class c, given the observed attribute values $v = \langle v_1, v_2, \ldots, v_a \rangle$, can be computed using the a priori probability of an instance being class c, $P(c)$; the probability of an instance of class c having the observed attribute values, $P(v \mid c)$; and the prior probability of observing the attribute values, $P(v)$. In the naive Bayesian approach, the likelihoods of the observed attribute values are assumed to be mutually independent for each class. With this attribute independence assumption, the probability $P(c \mid v)$ can be reexpressed as

$$P(c \mid v) = \frac{P(c)}{P(v)} \prod_{i=1}^{a} P(v_i \mid c) \propto P(c) \prod_{i=1}^{a} P(v_i \mid c). \tag{4}$$

Note that since $P(v)$ is the same for all classes, its value does not affect the final classification, and thus can be ignored. In our implementation of the naive Bayesian classifier, probabilities are estimated using frequency counts with an m-estimate correction (Cestnik 1990), where $m = 2$. It is a form of Dirichlet prior correction. Continuous attributes are preprocessed using an entropy-based discretization method (Fayyad and Irani 1993).

3.2. Leveled Naive Bayesian Trees

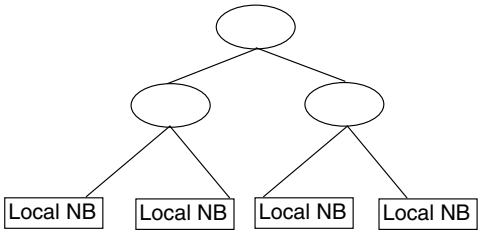
The decision nodes of a naive Bayesian tree contain the univariate tests of conventional decision trees. Each leaf contains a local naive Bayesian classifier that does not consider attributes involved in tests on the path leading to the leaf. Whereas a conventional decision tree labels each leaf with 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. Two examples of leveled naive Bayesian trees are given in Figure 1.

Our initial motivation for designing the leveled naive Bayesian tree learning algorithm is to introduce instability into naive Bayesian classification. We show in Section 4.3 that it effectively reduces the bias and increases the variance of the naive Bayesian classifier. We use the maximum depth of a tree as a parameter of the algorithm to control the level of tree structures incorporated into naive Bayesian classification.

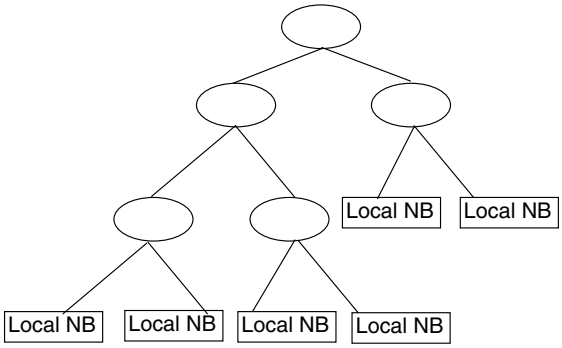
The leveled naive Bayesian tree learning algorithm, called LNBT, has the same tree-growing procedure as in a Top-Down Decision Tree induction algorithm. Like C4.5 (Quinlan 1993), LNBT also uses the information gain ratio criterion to select the best test at each decision node during the growth of a tree. The differences between LNBT and C4.5 are: (i) LNBT(d) grows a tree up to a predefined maximum depth d; (ii) LNBT generates one local naive Bayesian classifier at each leaf using all training examples at that leaf and all attributes that do not appear on the path from the root to the leaf; (iii) LNBT stops splitting a node, if it contains less than 30 training examples, to ensure reasonably accurate estimates for local naive Bayesian classifiers; and (iv) LNBT does not perform pruning. When the maximum tree depth d is set at 0, LNBT is identical to a normal naive Bayesian classifier generator. Therefore, the naive Bayesian classifier is referred to as LNBT(0) later in this article.

3.3. Bias and Variance

The *bias* and *variance* analysis can provide useful insights into the generalization performance of a learning algorithm. Bias is a measure of error due to the central tendency and variance is a measure of error due to disagreements between the classifiers formed by an algorithm across a group of training sets. Among several definitions of bias and variance, we, in this study, adopt Kohavi and Wolpert's (1996) definitions that decompose error into



(a) An example of a level 2 leveled naive Bayesian Tree: LNBT(2)



(b) An example of a level 3 leveled naive Bayesian Tree: LNBT(3)

FIGURE 1. Two examples of leveled naive Bayesian trees. Each internal node is a univariate test in the form of "Color = blue" or "Temperature $>70^{\circ}$C," selected from the given attributes in the training set.

intrinsic noise (optimal Bayes error), squared bias, and variance. Let C be the set of class labels, C_H be the random variable representing the class label of an example in the hypothesis space of a learner, and C_T be the random variable representing the class label of an example in the target space of the target concept. The error of a model can be decomposed into a sum of three terms (Kohavi and Wolpert 1996) as follows:

$$Error = \sum_{v} P(v)\big(noise_v + bias_v^2 + variance_v\big) \tag{5}$$

where

$$noise_v \equiv \frac{1}{2}\left(1 - \sum_{c \in C} P(C_T = c \mid v)^2\right)$$

$$bias_v^2 \equiv \frac{1}{2}\sum_{c \in C}[P(C_T = c \mid v) - P(C_H = c \mid v)]^2$$

$$variance_v \equiv \frac{1}{2}\left(1 - \sum_{c \in C} P(C_H = c \mid v)^2\right).$$

Due to the difficulty of estimating the intrinsic noise in practical experiments with real domains, we follow Kohavi and Wolpert's (1996) strategy of generating a bias term that includes both intrinsic noise and squared bias. The values are calculated directly from the performance of each classifier on each test case in the experiments described below.

4. EXPERIMENTS

In this section, we first empirically show that boosting the naive Bayesian classifier does not improve the accuracy as much as we expected. Then, we incorporate tree structures into naive Bayesian classification and show that this will improve the performance of boosting for naive Bayesian classification. We use BLNBT($d \geq 1$) to refer to the boosted leveled naive Bayesian tree classifier, and BLNBT($d = 0$) for the boosted naive Bayesian classifier. The parameter T controlling the number of models generated in boosting is set at 100 for all experiments, except when stated otherwise. It is interesting to see the performance improvement that can be gained by an increase in computation of two orders of magnitude.

Twenty-five domains from the UCI machine learning repository (Blake et al. 1998) are used in the experiments. Table 1 gives a description of these domains.

Ten threefold cross-validations are used in each domain in our experiments. These allow us to compute a good estimate of the bias and variance analysis in each domain.

The main concern of this study is the accuracy performance of learning algorithms. Therefore, we report the *error rate* of each algorithm under consideration on the test set in each domain. To compare two algorithms, A versus B, for example, we provide the error

TABLE 1. Description of the Domains Used in the Experiments.

Domain	Size	No. of classes	No. of attributes	
			Numeric	Nominal
Abalone	4,177	3	7	1
Annealing processes	898	6	6	32
Audiology	226	24	0	69
Breast cancer (Wisconsin)	699	2	9	0
Chess (king-rook vs. king-pawn)	3,169	2	0	36
Coding	20,000	2	0	15
Credit screening (Australia)	690	2	6	9
DNA	3,186	3	0	60
Glass identification	214	6	9	0
Heart disease (Cleveland)	303	2	13	0
Horse colic	368	2	7	15
House votes 84	435	2	0	16
Hypothyroid diagnosis	3,163	2	7	18
LED 24 (noise level = 10%)	200	10	0	24
Liver disorders	345	2	6	0
Nettalk (stress)	5,438	5	0	7
Nursery	12,960	5	8	0
Pima Indians diabetes	768	2	8	0
Primary tumor	339	22	0	17
Satellite	6,435	6	36	0
Solar flare	1,389	2	0	10
Sonar classification	208	2	60	0
Soybean large	683	19	0	35
Splice junction gene sequences	3,177	3	0	60
Tic-Tac-Toe endgame	958	2	0	9

rate ratio of A over B: ϵ_A/ϵ_B in each domain, where ϵ_A is the test error rate of A and ϵ_B is the test error rate of B. In each domain, the reported value is an average value over the ten threefold cross-validations. To compare two algorithms, A versus B, across the 25 domains, the w/l record is reported, which is the number of wins and losses between the error rates of A versus B in these domains. A one-tailed pairwise sign-test is applied on this w/l record to examine whether A performs better than B significantly more often than the reverse, at a level better than 0.05. The significance level p will be provided for the sign-test on each w/l record.

4.1. Boosting Naive Bayesian Classifiers

Table 2 shows the error rates of the naive Bayesian classifier and the boosted naive Bayesian classifier as well as their error ratios. From this table, we can see that although

TABLE 2. Error Rates (%) of the Naive Bayesian Classifier and the Boosted Naive Bayesian Classifier as well as Their Error Ratios.

	Error rate (%)		Error ratio
Domain	LNBT(0)	BLNBT(0)	BLNBT(0)/LNBT(0)
Abalone	42.1	41.6	0.99
Annealing	3.0	1.1	0.36
Audiology	26.4	23.4	0.89
Breast (W)	2.6	4.2	1.59
Chess (kr-kp)	12.5	3.8	0.30
Coding	28.7	28.8	1.00
Credit(A)	14.5	16.6	1.14
DNA	4.6	7.1	1.53
Glass	34.9	35.6	1.02
Heart (C)	17.6	19.1	1.09
Horse colic	20.7	23.4	1.13
House votes 84	10.0	5.8	0.58
Hypothyroid	1.8	1.5	0.83
LED24	38.3	52.8	1.38
Liver disorder	37.4	37.5	1.00
Nettalk(S)	16.3	15.6	0.96
Nursery	9.7	7.7	0.79
Pima	25.9	25.5	0.98
Primary tumor	52.6	52.3	0.99
Satellite	18.0	17.7	0.98
Solar flare	19.2	16.6	0.86
Sonar	24.7	27.1	1.10
Soybean	9.7	7.1	0.73
Splice junction	4.5	6.9	1.54
Tic-tac-toe	29.6	3.2	0.11
Mean	20.2	19.3	0.96
w/l			14/11
p. of w/l			0.3450

boosting can improve the accuracy of the naive Bayesian classifier in 14 out of the 25 domains, it does decrease the accuracy of the naive Bayesian classifier in the other 11 domains. A one-tailed pairwise sign-test fails to show that boosting reduces the error of the naive Bayesian classifier significantly more often than the reverse across the 25 domains at a level of 0.05. When boosting decreases the error of the naive Bayesian classifier, the error reduction is often small: only in five domains does boosting reduce the error of the naive Bayesian classifier by more than 25%. The mean relative error reduction of boosting over the naive Bayesian classifier in the 25 domains is only 4%, indicating very marginal improvement due to boosting. In terms of the mean error rate, the boosted naive Bayesian classifier is slightly better than the naive Bayesian classifier by 0.9 percentage points across the 25 domains.

By contrast, boosted C4.5 reduces the relative error rate of C4.5 (Quinlan 1993) by 23% in the same 25 domains. The mean error rate is reduced from 19.2% to 15.8%.

To check the effects of using a small number of trials and of the number of restarts on the performance of boosting naive Bayesian classifiers, we conduct experiments using $T = 1, 5, 10, 20, 50, 75,$ and 100 to examine these issues. The results are shown in Figure 2, which shows the average error rate and the average number of restarts due to $\epsilon \geq 0.5$ and $\epsilon = 0$, as a function of T. At $T = 100$, boosting naive Bayesian classifiers performs better than that at $T = 10$ in the Tic-Tac-Toe and Nettalk domains. In the other six domains, the difference is insignificant. In cases where improvement is possible, restarts allow more models to be generated and improve the boosting performance. In other cases, restarts do not cause significant degradation of performance. It is interesting to note that a small number of restarts due to $\epsilon = 0$ improves accuracy, as in the annealing domain. This type of restart only occurs when the base model has accuracy approaching 100%. That is the reason why it only occurs in the annealing domain, and the average number of restarts is considerably smaller (i.e., less than 0.5 at $T = 100$) than those in other domains in our experiment. Boosted naive Bayesian classifiers in domains not shown in Figure 2 demonstrate behavior similar to the cases in which restarts either improve performance or do not cause significant performance degradation. In short, the result suggests that using a large number of trials and restarts is preferred in a domain with little background knowledge.

In summary, boosting does not work as well with naive Bayesian classification as with decision trees. Our explanation for this observation is that the naive Bayesian classifier is very stable (with a strong bias) with respect to small changes to training data, since these changes will not result in big changes to the estimated probabilities. Therefore, boosting naive Bayesian classifiers may not generate multiple models with sufficient diversity. These models cannot effectively correct each other's errors during classification by voting. In such a situation, boosting cannot greatly reduce the error of naive Bayesian classification. Further analysis to be reported in Section 6 reveals that this strong uncorrectable bias cannot be reduced by boosting because the final boosted model has exactly the same form as that for the naive Bayesian classifier. An effect of bias reduction and variance increase is achieved through the introduction of tree structures. The experiments in the following subsection are designed to provide support to this argument.

4.2. Boosting Leveled Naive Bayesian Trees

Decision tree learning is not stable, in the sense that small changes in training data may result in different decision trees. We expect that introducing tree structures into naive Bayesian classifiers can increase the instability of naive Bayesian classification, thus improving the effectiveness of boosting for naive Bayesian classification. To investigate this issue, we compare the leveled naive Bayesian tree with its boosted version by gradually increasing

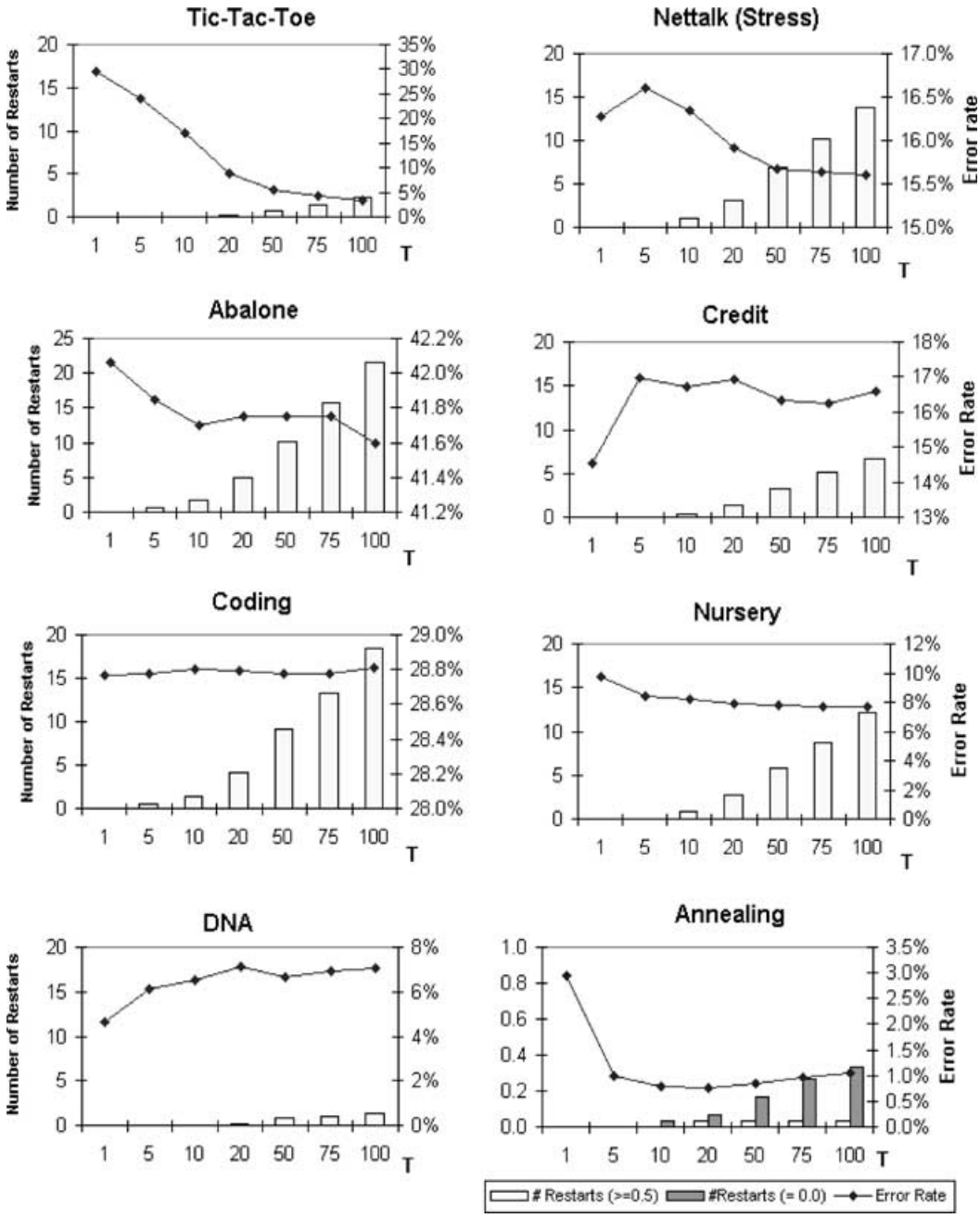


FIGURE 2. Average error rates and average number of restarts due to $\epsilon \geq 0.5$ and $\epsilon = 0$ for boosting naive Bayes in eight domains as a function of T. Note that this is the result of naive Bayes when $T = 1$.

the maximum depth of trees. The results are shown in Table 3, which shows the error rates of LNBT(d) and error ratios of BLNBT(d) over LNBT(d) for $d = 1, 3, 5$, and 10. As a baseline for comparison, the results of LNBT(0) and BLNBT(0) are also included in the table. From this table, we have the following observations:

TABLE 3. Error Rates (%) of LNBT(d) and Error Ratios of BLNBT(d) versus LNBT(d) for d = 0, 1, 3, 5, and 10.

	BLNBT(d) vs LNBT(d)									
	$d = 0$		$d = 1$		$d = 3$		$d = 5$		$d = 10$	
Domain	Err (%)	Ratio	Err (%)	Ratio	Err (%)	Ratio	Err (%)	Ratio	Err (%)	Ratio
Abalone	42.1	0.99	38.7	0.98	38.5	0.99	38.6	0.99	38.6	0.99
Annealing	3.0	0.36	2.7	0.24	4.9	0.19	4.9	0.23	4.9	0.25
Audiology	26.4	0.89	25.0	0.86	30.0	0.70	30.0	0.70	30.0	0.70
Breast (W)	2.6	1.59	5.5	0.69	5.5	0.64	5.5	0.66	5.5	0.66
Chess (kr-kp)	12.5	0.30	6.6	0.21	4.3	0.12	3.4	0.15	2.5	0.20
Coding	28.7	1.00	28.0	0.96	27.0	0.72	27.8	0.67	27.8	0.67
Credit (A)	14.5	1.14	13.7	1.29	15.8	1.08	15.8	1.09	15.8	1.10
DNA	4.6	1.53	4.6	1.46	8.6	0.59	10.8	0.48	11.1	0.52
Glass	34.9	1.02	38.7	0.90	38.0	0.91	38.0	0.91	38.0	0.91
Heart (C)	17.6	1.09	20.0	1.05	20.5	1.00	20.5	1.04	20.5	1.04
Horse colic	20.7	1.13	18.9	1.13	19.1	1.10	19.1	1.11	19.1	1.11
House Votes 84	10.0	0.58	4.9	1.06	4.9	1.08	4.9	1.04	4.9	1.04
Hypothyroid	1.8	0.83	1.5	0.95	1.5	0.89	1.5	0.88	1.5	0.91
LED24	38.3	1.38	41.6	1.10	50.1	0.90	50.1	0.90	50.1	0.90
Liver disorder	37.4	1.00	37.6	0.99	37.8	1.00	37.8	0.99	37.8	0.99
Nettalk(s)	16.3	0.96	15.9	1.11	18.8	0.74	18.8	0.74	18.8	0.74
Nursery	9.7	0.79	8.2	0.81	5.3	0.07	3.5	0.13	3.5	0.15
Pima	25.9	0.98	26.3	1.01	27.1	1.08	27.1	1.09	27.1	1.09
Primary tumor	52.6	0.99	53.3	1.00	57.0	1.00	57.3	1.01	57.3	1.01
Satellite	18.0	0.98	18.3	0.82	18.8	0.67	18.2	0.64	18.0	0.64
Solar flare	19.2	0.86	19.2	0.87	18.7	0.97	18.6	0.98	18.6	0.98
Sonar	24.7	1.10	25.9	0.96	26.2	0.87	26.2	0.87	26.2	0.87
Soybean	9.7	0.73	10.0	0.73	11.3	0.58	11.9	0.58	11.9	0.56
Splice junction	4.5	1.54	4.7	1.36	8.8	0.57	10.9	0.48	11.2	0.49
Tic-tac-toe	29.6	0.11	27.6	0.31	21.4	0.09	21.4	0.09	21.4	0.09
Mean	20.2	0.96	19.9	0.91	20.8	0.74	20.9	0.74	20.9	0.74
w/l		14/11		15/10		20/5		19/6		19/6
p. of w/l		0.3450		0.2122		0.0020		0.0073		0.0073

- Comparing BLNBT(d) with LNBT(d), boosting improves the average accuracy of the leveled naive Bayesian tree as we increase the maximum depth of the tree. The error ratio of BLNBT(d) over LNBT(d) continues to decrease from 0.96 to 0.74 as d increases from 0 to 10.
- For $d \geq 3$, boosting reduces the error of the leveled naive Bayesian tree significantly more often than the reverse at a level better than 0.05.
- Introducing tree structures into the naive Bayesian classification increases the error on average. After a small decrease in error resulted from introducing 1-level tree structures into the naive Bayesian classification, introducing deeper tree structures consistently increases the error on average: from 19.9 to 20.9 when d increases from 1 to 10.

The last observation indicates that the relative accuracy improvement of boosting for naive Bayesian classification when tree structures are added (which is shown in Table 3)

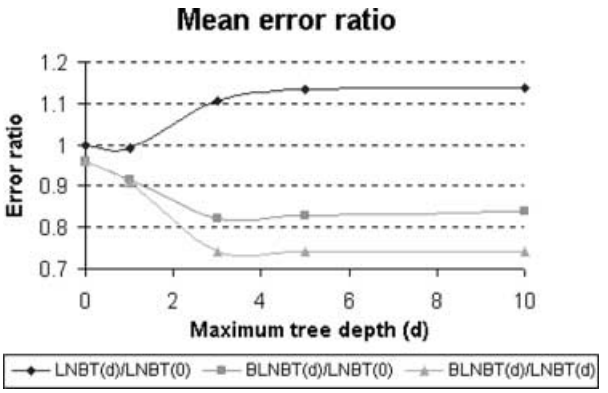


FIGURE 3. Mean error ratio for the 25 domains as functions of d.

might not be interesting, since the algorithms being compared have a higher error than the naive Bayesian classifier. To clarify this issue, we use the naive Bayesian classifier LNBT(0) as the base classifier for comparison. We analyze the error ratio of LNBT(d) over LNBT(0), and the error ratio of BLNBT(d) over LNBT(0) for $d = 1, 3, 5$, and 10, and present them in Figure 3. From this analysis, we obtain the following observations:

- The mean error ratio of BLNBT(d) over the naive Bayesian classifier in the 25 domains keeps decreasing from 0.96 to 0.82 when d increases from 0 to 3, and it increases slightly to 0.84 at $d = 10$. In addition, we find that the mean error rate of BLNBT(d) in the 25 domains is consistently reduced from 19.3% to 17.6%, when d increases from 0 to 3, and it stays at 17.6% for $d \geq 3$.
- When $d = 10$, BLNBT(d) achieves 16% mean relative error reduction over the naive Bayesian classifier in the 25 domains—a great improvement in comparison to the 4% reduction when boosting the naive Bayesian classifier. Also, the mean error rate of BLNBT(d) is 2.6 percentage points lower than that of the naive Bayesian classifier. In contrast, boosting the naive Bayesian classifier only reduces the mean error rate by 0.9 percentage points.
- Compared to BLNBT(0), BLNBT(10) achieves 15% relative error reduction over the 25 domains, although the base learning algorithm of BLNBT(10) has an mean error rate 7% higher than the base learning algorithm of BLNBT(0). A sign-test shows that BLNBT(10) has lower error than BLNBT(0) significantly more often than the reverse at a level 0.0216 across the 25 domains. The w/l record is 18/7.

The reason why BLNBT(d) does not further improve its performance for $d > 3$ in our experiments is because LNBT produces trees with $d < 3$ in more than half of the 25 domains. The mean tree depths for LNBT(10) and BLNBT(10) are 3.1 and 3.8, respectively, over the 25 domains. When it could produce trees with a large depth, boosting shows a continuing performance improvement as the depth of the trees increases. The two domains in which LNBT produces the deepest trees are Satellite and Chess with an average depth of 6.1 and 7.2, respectively. Figure 4 shows the mean error rate for LNBT(d) and BLNBT(d) in these two domains as a function of d. In the Satellite domain, BLNBT(d) continues to decrease its error as d increases, though from $d = 5$ to $d = 10$, it only decreases the error marginally from 11.6% to 11.4%. In the Chess domain, BLNBT(d) reaches its best performance of mean error rate 0.5% at $d = 3$, and it maintains that performance for $d > 3$.

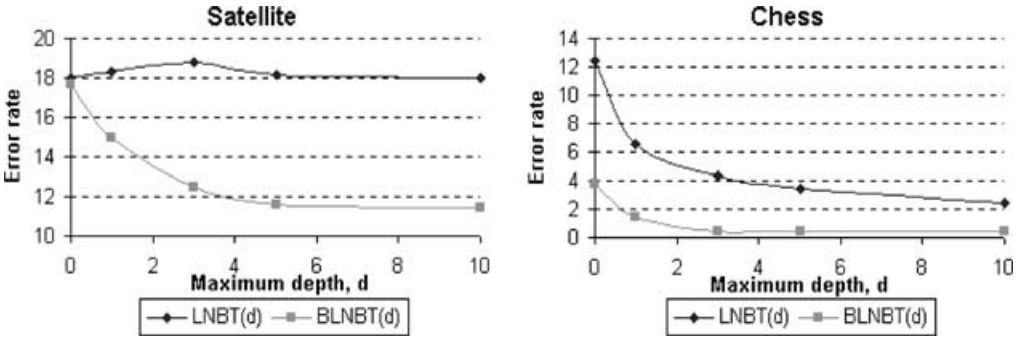


FIGURE 4. Mean error ratio for LNBT(d) and BLNBT(d) in the Satellite and Chess domains as a function of d.

4.3. Bias and Variance Analyses

Table 4 shows the bias and variance decomposition for leveled naive Bayesian trees and their boosted versions for $d = 0, 1, 3, 5$, and 10. The result clearly shows the following:

- Introducing tree structure into naive Bayesian classifier reduces its bias but increases its variance. This confirms our expectation that it will increase the instability of the naive Bayesian classifier. For $d \geq 3$, the increase in variance outweighs bias reduction; thus it increases the error rate.
- For all values of d, boosting reduces the bias of LNBT(d).
- For $d \geq 3$, boosting is able to reduce the variance substantially.
- For $d \geq 3$, the combined reduction of bias and variance by BLNBT(d) outweighs the net increase in error of its base LNBT(d) over the naive Bayesian classifier (stated in the first bullet point above). Thus, BLNBT(d) effectively reduces the error of the naive Bayesian classifier.

In summary, all the experimental results show that introducing tree structures into the naive Bayesian classification can lead to a better performance for boosting the naive Bayesian classifier. Introducing tree structures results in bias reduction and variance increase for the naive Bayesian classifier. Boosting continues to reduce the reduced bias and starts to reduce the increased variance. As a result, although the leveled naive Bayesian tree learner is less accurate than the naive Bayesian classifier on average, boosting leveled naive Bayesian trees can produce significantly higher accuracy than the naive Bayesian classifier, performing much better than directly boosting the naive Bayesian classifier.

TABLE 4. Bias and Variance of the Leveled Naive Bayesian Trees and the Boosted Leveled Naive Bayesian Trees.

Max. tree	Bias		Variance	
depth d	LNBT(d)	BLNBT(d)	LNBT(d)	BLNBT(d)
0	16.7	14.0	3.5	5.0
1	14.9	12.7	5.0	6.0
3	13.8	11.6	7.0	5.9
5	13.6	11.7	7.3	5.9
10	13.5	11.7	7.3	5.9

5. DISCUSSION

This study is motivated by recent research on boosting (Quinlan 1996; Schapire et al. 1997; Bauer and Kohavi 1999; Friedman et al. 2000). Although most researchers focus on boosting decision trees, Bauer and Kohavi (1999) explore the direct application of boosting to the naive Bayesian classifier.

Bauer and Kohavi (1999) state that "for Naive-Bayes, the average absolute error decreased from 13.5% to 12.3%, a 24% decrease in average relative error. . . . The bias reduced from 10.8% to 8.7%, an average relative reduction of 27%, while the variance *increased* from 2.8% to 3.6%. The increased variance is likely to be caused by different discretization thresholds, as real-valued attributes are discretized using an entropy-based method that is unstable."

While we get similar results for boosting the naive Bayesian classifier in terms of bias reduction and variance increase, the amount of reduction is almost equal to the amount of increase in our result. As a result, there is only a minor average relative error reduction of 4%. The discrepancy between Bauer and Kohavi's result and ours is because different sets of domains are used. Only six similar domains (i.e., Chess, DNA, Hypothyroid, LED24, Nursery, and Satellite) are used in both experiments. It seems that Bauer and Kohavi (1999) used the set of domains in which boosting the naive Bayesian classifier is more likely to succeed. In our result, of the six domains used by Bauer and Kohavi, boosting reduces the error in four domains and increases the error in two domains.

The fact that the naive Bayesian classifier is a stable classifier is reflected in the small proportion of variance in the total error—20% of total error in Bauer and Kohavi's result and 17% in our result. It does not seem to have room for variance reduction. Our result shows that introducing the tree structure into the naive Bayesian classifier reduces its bias and increases its variance, and this allows boosting to reduce error via both bias and variance reduction. While we agree that the increase in variance in boosting the naive Bayesian classifier is likely to be due to the different discretization thresholds, the more important reason why boosting does not result in error reduction is that there is simply no room for reduction—because of strong uncorrectable bias and small variance.

Leveled naive Bayesian trees are used in this article to illustrate the issues under investigation, because they provide an easy way to control the levels of bias and variance introduced to the naive Bayesian classifiers. In real-world applications we expect boosted NBTrees (Kohavi 1996) to perform better.

6. BOOSTED NAIVE BAYES IS JUST ANOTHER NAIVE BAYES

Ridgeway et al. (1998) show that the boosted naive Bayesian classifier is just another naive Bayesian classifier in the log-odd scale, using the Taylor approximation. In fact, by using the Real AdaBoost procedure (Schapire and Singer 1999), it is shown below that it is exactly the same.

Real AdaBoost produces the final classifier in the following form:

$$H(v) = \sum_{t=1}^{T} \alpha_t h_t(v), \tag{6}$$

where α_t is a real parameter. Writing the output of the naive Bayesian classifier in the log-odd scale for two-class problems gives:

$$h_t(v) = \log \frac{P(c = 1 \mid v)}{P(c = 0 \mid v)}.$$

Substituting equation (4) into the above expressions yields a naive Bayesian classifier that is exactly linear on the log-odd scale,

$$H(v) \propto \sum_{t=1}^{T} \alpha_t \log \frac{P_t(c=1)}{P_t(c=0)} + \sum_{i=1}^{a} \sum_{t=1}^{T} \alpha_t \log \frac{P_t(v_i \mid c=1)}{P_t(v_i \mid c=0)}. \tag{7}$$

This means that the boosted naive Bayes has a strong bias on a linear form, exactly the same form as its base classifier. This bias is so strong that it is not reducible by the boosting procedure.

Introducing tree structure reduces the bias of the final model BLNBT(d), which is no longer in linear form of the log-odd scale of attributes. This enables boosting to reduce its bias further.

7. CONCLUSIONS

Although boosting achieves great success with decision tree learning, we experimentally show that boosting does not work well for naive Bayesian classification. We suspect its relatively poor performance is mainly due to the fact that the naive Bayesian classifier is a stable classifier with a strong bias. We propose to introduce tree structures into the naive Bayesian classification to reduce its bias and increase its variance (thus its instability), and expect that this can improve the success of boosting for naive Bayesian classification.

To verify the expectation, we have conducted a set of experiments on boosting naive Bayesian classification with tree structures in 25 natural domains. The experimental results show that although introducing tree structures into naive Bayesian classification increases the average error of naive Bayesian classification for individual models, boosting naive Bayesian classifiers with tree structures can achieve significantly lower average error than the naive Bayesian classifier, providing a method of successfully applying the boosting technique to naive Bayesian classification.

The bias and variance analysis confirms our expectation that the naive Bayesian classifier is a stable classifier with low variance and high bias. As expected, introducing tree structures reduces the bias and increases the variance. Being a bias reduction method, boosting is able to further reduce the already reduced bias and starts to reduce the increased variance. An extension of the previous analysis on the boosted naive Bayesian classifier shows that the final boosted model has exactly the same form as its base learner, thus imposing a strong uncorrectable bias that prevents the bias reduction mechanism in boosting from taking effect. Introducing tree structure into the naive Bayesian classifier changes the structure of the final model and reduces the bias.

ACKNOWLEDGMENT

Comments from the anonymous reviewers have helped to improve this article. The explanation in Section 6 is attributed to one of the anonymous reviewers. Ross Quinlan provides C4.5.

REFERENCES

BAUER, E., and R. KOHAVI. 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning, **36:**105–139.

BLAKE, C., E. KEOGH, and C. J. MERZ. 1998. UCI repository of machine learning databases *http://www.ics.uci.edu/~mlearn/MLRepository.html*.

BREIMAN, L. 1996. Bias, variance, and arcing classifiers. Technical Report 460, Department of Statistics, University of California, Berkeley.

CESTNIK, B. 1990. Estimating probabilities: A crucial task in machine learning. *In* Proceedings of European Conference on Artificial Intelligence, pp. 147–149.

DOMINGOS, P., and M. PAZZANI. 1996. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. *In* Proceedings of the Thirteenth International Conference on Machine Learning. Morgan Kaufmann, San Francisco, pp. 105–112.

DUDA, R. O., and P. E. HART. 1973. Pattern Classification and Scene Analysis. John Wiley, New York.

ELKAN, C. 1997. Boosting and Naive Bayesian Learning. Technical Report CS97-557, University of California, Davis.

FAYYAD, U. M., and K. B. IRANI. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann, San Francisco, pp. 1022–1027.

FREUND, Y., and R. E. SCHAPIRE. 1996. Experiments with a new boosting algorithm. *In* Proceedings of the Thirteenth International Conference on Machine Learning. Morgan Kaufmann, San Francisco, pp. 148–156.

FRIEDMAN, J. H., T. HASTIE, and R. TIBSHIRANI. 2000. Additive logistic regression: A statistical view of boosting. Annals of Statistics, **28**(2):337–374.

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

KOHAVI, R., and D. WOLPERT. 1996. Bias plus variance decomposition for zero-one loss functions. *In* Proceedings of the Thirteenth International Conference on Machine Learning. Morgan Kaufmann, San Francisco, pp. 275–283.

KONONENKO, I. 1990. Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition. *In* Current Trends in Knowledge Acquisition. *Edited by* B. Wielinga et al. IOS Press, Amsterdam, pp. 190–197.

LANGLEY, P., W. F. IBA, and K. THOMPSON. 1992. An analysis of Bayesian classifiers. *In* Proceedings of the Tenth National Conference on Artificial Intelligence. AAAI Press, Menlo Park, CA, pp. 223–228.

QUINLAN, J. R. 1993. C4.5: Program for Machine Learning. Morgan Kaufmann, San Francisco.

QUINLAN, J. R. 1996. Bagging, boosting, and C4.5. *In* Proceedings of the Thirteenth National Conference on Artificial Intelligence. AAAI Press, Menlo Park, CA, pp. 725–730.

RIDGEWAY, G., D. MADIGAN, T. RICHARDSON, and J. O'KANE. 1998. Interpretable boosted naive Bayes classification. *In* Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, Menlo Park, CA, pp. 101–104.

SCHAPIRE, R. E., Y. FREUND, P. BARTLETT, and W. S. LEE. 1997. Boosting the margin: A new explanation for the effectiveness of voting methods. *In* Proceedings of the Fourteenth International Conference on Machine Learning. Morgan Kaufmann, San Francisco, pp. 322–330.

SCHAPIRE, R. E., and Y. SINGER. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, **37**(3):297–336.