Integrating Boosting and Stochastic Attribute Selection Committees for Further Improving the Performance of Decision Tree Learning

Zijian Zheng, Geoffrey I. Webb, and Kai Ming Ting
School of Computing and Mathematics
Deakin University, Geelong
Victoria 3217, Australia
{zijian,webb,kmting}@deakin.edu.au

Abstract

Techniques for constructing classifier committees including Boosting and Bagging have demonstrated great success, especially Boosting for decision tree learning. This type of technique generates several classifiers to form a committee by repeated application of a single base learning algorithm. The committee members vote to decide the final classification. Boosting and Bagging create different classifiers by modifying the distribution of the training set. SASC *(Stochastic Attribute Selection Committees) uses an alternative approach to generating classifier committees by stochastic manipulation of the set of attributes considered at each node during tree induction, but keeping the distribution of the training set unchanged. In this paper, we propose a method for improving the performance of Boosting. This technique combines Boosting and* SASC. *It builds classifier committees by manipulating both the distribution of the training set and the set of attributes available during induction. In the synergy,* SASC *effectively increases the model diversity of Boosting. Experiments with a representative collection of natural domains show that, on average, the combined technique outperforms either Boosting or* SASC *alone in terms of reducing the error rate of decision tree learning.*

1. Introduction

In order to increase the prediction accuracy of classifiers, classifier committee[1] learning techniques have been developed with great success [11, 12, 13, 18, 4, 3, 9, 1, 6, 20, 10, 2, 21], especially Boosting[2] [13, 18, 2]. This type of technique generates several classifiers to form a committee by using a single base learning algorithm. At the classification stage, the committee members vote to make the final decision.

Given a training set described using a set of attributes, conventional classifier learning algorithms such as decision tree learning algorithms [5, 17] build one classifier. Usually, the classifier is correct for most parts of the instance space, but incorrect for some small parts of the instance space. If classifiers in a committee partition the instance space differently, and most points in the instance space are correctly covered by the majority of the committee, then the committee has a lower error rate than the individual classifiers.

Bagging [4] and Boosting [19, 11, 12, 13, 20], as two representative methods of this type, can significantly decrease the error rate of decision tree learning [18, 13, 2] with Boosting being generally better than Bagging [18, 2]. They repeatedly build different classifiers using a base learning algorithm, such as a decision tree generator, by changing the distribution of the training set. Bagging generates different classifiers using different bootstrap samples. Boosting builds different classifiers sequentially. The weights of training examples used for creating each classifier are modified based on the performance of the previous classifiers. The objective is to make the generation of the next classifier concentrate on the training examples that are misclassified by the previous classifiers. The main difference between Bagging and Boosting is that the latter adaptively changes the distribution of the training set based on the performance of previously created classifiers and uses a function of the performance of a classifier as the weight for voting, while the former uses

[1]Committees are also referred to as ensembles [7].

[2]Breiman [3] refers to Boosting as the arcing (adaptively resample and combine) method.

equal weight voting.

In contrast to Bagging and Boosting, SASC (Stochastic Attribute Selection Committees) adopts an alternative approach to generating different classifiers to form a committee [21]. It builds different classifiers by modifying the set of attributes considered at each node during tree induction, while the distribution of the training set is kept unchanged. The selection of an attribute set is carried out stochastically. Experiments show that as BOOST, SASC can also significantly reduce the error rate of decision tree learning, although the two techniques use quite different mechanisms [21].

In the light of this finding, we propose, in this paper, a novel approach to further improving the accuracy of decision tree learning. The new approach is called **SASCB** (**S**tochastic **A**ttribute **S**election **C**ommittees with **B**oosting), a combination of the Boosting and SASC techniques. Since SASC and Boosting improve the accuracy of decision tree learning using different mechanisms, we expect that combining them can take advantage from both. Indeed, we show that the synergy performs, on average, better than either SASC or Boosting alone. Our analysis suggests that the improvement is achieved through increasing model diversity by SASC, in addition to that by Boosting.

There are some other classifier committee learning approaches such as generating multiple trees by manually changing learning parameters [15], error-correcting output codes [8], and generating different classifiers by randomizing the base learning process [9, 1] which is similar to SASC [21]. Reviews of related methods are provided by Dietterich [7] and Ali [1]. A collection of recent research in this area can be found from [6].

In the following section, we briefly describe the Boosting and SASC techniques for decision tree learning. Section 3 presents the SASCB method of combining Boosting and SASC. SASCB is, then, empirically evaluated using a representative collection of natural domains. Finally, we summarize our conclusions.

2. Boosting and SASC

Since the SASCB technique is a combination of Boosting and SASC, we briefly discuss Boosting and SASC in this section before describing the approach to combining them. The classification process of Boosting and SASC is presented in Section 2.3, since it is the same for both of them. For details about Boosting, see [19], [18], [20], and [2]. For details about SASC, see [21].

2.1. Boosting

Boosting is a general framework for improving base learning algorithms, such as decision tree learning, rule learning, and neural networks. The key idea of Boosting was presented in Section 1. Here, we describe our implementation of the Boosting algorithm with decision tree learning, called BOOST. It follows the Boosted C4.5 algorithm (AdaBoost.M1) [18] but uses a new Boosting equation as shown in Equation 1, derived from [20].

Given a training set D consisting of m instances and an integer T, the number of trials, BOOST builds T pruned trees over T trials by repeatedly invoking C4.5. Let $w_t(x)$ denote the weight of instance x in D at trial t. At the first trial, each instance has weight 1; that is, $w_1(x) = 1$ for each x. At trial t, decision tree H_t is built using D under the distribution w_t. The error ϵ_t of H_t is, then, calculated by summing up the weights of the instances that H_t misclassifies and divided by m. If ϵ_t is greater than 0.5 or equal to 0, $w_t(x)$ is re-initialized using bootstrap sampling, and then the Boosting process continues. Note that the tree with $\epsilon_t > 0.5$ is discarded,[3] while the tree with $\epsilon_t = 0$ is accepted by the committee. Otherwise, the weight $w_{t+1}(x)$ of each instance x for the next trial is computed using Equation 1. These weights are, then, renormalized so that they sum to m.

$$w_{(t+1)}(x) = w_t(x)exp((-1)^{d(x)}\alpha_t), \qquad (1)$$

where $\alpha_t = \frac{1}{2}ln((1-\epsilon_t)/\epsilon_t)$; $d(x) = 1$ if H_t correctly classifies x and $d(x) = 0$ otherwise.

2.2. SASC

During the growth of a decision tree, at each decision node, a decision tree learning algorithm searches for the best attribute to form a test based on some test selection functions [5, 17]. The key idea of SASC is to vary the members of a decision tree committee by stochastic manipulation of the set of attributes available for selection at decision nodes. This creates decision trees that each partition the instance space differently. In order to have a good quality tree in the sense that it can correctly cover most parts of the instance space, the tests used at decision nodes should be as good as possible with respect to the test selection function employed.

[3]To make the algorithm efficient, this step is limited to $10 \times T$ times.

```
C4.5SAS(Att, D, W, P)
    INPUT: Att: a set of attributes,
           D: a training set represented using Att and
              classes,
           W: instance weights for D,
           P: a probability value.
    OUTPUT: a pruned tree.

    C := the majority class in D
    RawTree := Grow-Tree-SAS(Att, D, W, C, P)
    PrunedTree := Prune-Tree(RawTree, Att, D, W)
    RETURN PrunedTree
```

Figure 1. The C4.5SAS decision tree learning algorithm

We use C4.5 [17] with the modifications described below as the base classifier learning algorithm in SASC. When building a decision node, by default C4.5 uses the information gain ratio to search for the best attribute to form a test [17]. To force C4.5 to generate different trees using the same training set, we modified C4.5 by stochastically restricting the set of attributes available for selection at a decision node. This is implemented by using a probability parameter P.[4] At each decision node, an attribute subset is randomly selected with each available attribute having the probability P of being selected. The available attributes refer to those attributes that have non-negative gain values. For nominal attributes, they must not have been used in the path from the root to the current node. Numeric attributes are always available for selection. This stochastic attribute subset selection process will be repeated, if no attribute was selected and there are some attributes available at this node. The objective is to make sure that at least one available attribute is included in the subset if possible. After attribute subset selection, the algorithm chooses the attribute with the highest gain ratio to form a test for the decision node from the subset. A different random subset is selected at each node of the tree. The modified version of C4.5 is called **C4.5SAS** (**C4.5 S**tochastic **A**ttribute **S**election). The probability of each attribute being included in a subset is specified by the parameter P with a default value of 33%. That is, by default, each available attribute has at least one third of chance of being selected into the subset. This allows C4.5SAS to have a chance to use different alternative attributes to form tests at decision nodes when building trees at different trials using the same training set. C4.5SAS still uses the best attribute from the subset to form a test each

[4]The value of P does not change during induction.

```
Grow-Tree-SAS(Att, D, W, C, P)
    INPUT: Att: a set of attributes,
           D: a training set,
           W: instance weights for D,
           C: the majority class at the parent node,
           P: a probability value.
    OUTPUT: a decision tree.
    IF (D is empty)
        RETURN a leaf node labeled with C
    ELSE
    {   C := the majority class in D
        IF (the stopping criterion is satisfied)
            RETURN a leaf node labeled with C
        ELSE
        {   $Att_{subset}$ := select a subset from Att in which
                          each available attribute has
                          the probability P of being selected.
            $Test_{best}$ := Find-Best-Test($Att_{subset}$, D, W)
            IF ($Test_{best}$ is reasonable (with a positive test
                evaluation function value))
            {   Use $Test_{best}$ to partition D and W into n
                    subsets $D_1, D_2, \cdots, D_n$ and $W_1, W_2, \cdots,$
                    $W_n$ respectively, one for each outcome
                    of the test
                RETURN the tree formed by a decision node
                    with the test $Test_{best}$ and subtrees:
                    Grow-Tree-SAS(Att, $D_1$, $W_1$, C, P),
                    Grow-Tree-SAS(Att, $D_2$, $W_2$, C, P),
                    ... ,
                    Grow-Tree-SAS(Att, $D_n$, $W_n$, C, P)
            }
            ELSE
                RETURN a leaf node labeled with C
        }
    }
```

Figure 2. The algorithm for growing a tree with stochastic attribute selection

time, although it may not use the best one among all the attributes available at a node every time.

Figures 1 and 2 provide a description of the C4.5SAS algorithm. The only difference between C4.5SAS and C4.5 is that when growing a tree, at a decision node, C4.5SAS creates an attribute subset Att_{subset} and uses the best attribute in it to form a test as described above. All other parts are identical for these two algorithms. With $P = 1$, C4.5SAS generates the same tree as C4.5.

Having C4.5SAS, the design of SASC is very simple. As described in Figure 3, C4.5SAS is invoked T times to generate T different decision trees to form a committee. Here, all the trees are pruned trees. As in Boosting, the first tree produced by SASC is the same

```
SASC($Att$, $D$, $P$, $T$ )
    INPUT: $Att$: a set of attributes,
           $D$: a training set represented using $Att$ and
              classes,
           $P$: a probability value,
           $T$: the number of trials.
    OUTPUT: a committee, $H$, containing $T$ pruned trees.
    Set instance weight $w(x) = 1$ for every $x$ in $D$
    $H_1$ := C4.5SAS($Att$, $D$, $w$, 1)
    FOR each $t$ from 2 to $T$
        $H_t$ := C4.5SAS($Att$, $D$, $w$, $P$ )
    RETURN $H$
```

Figure 3. The SASC learning algorithm

as the tree generated by C4.5.

2.3. Decision making in Boosting and SASC

At the classification stage, for a given example, both BOOST and SASC make the final prediction through committee voting. In this paper, a voting method that uses the probabilistic predictions produced by all committee members without voting weights is adopted. With this method, each decision tree returns a distribution over classes that the example belongs to. This is performed by tracing the example down to a leaf of the tree. The class distribution for the example is estimated using the proportion of the training examples of each class at the leaf, if the leaf is not empty. This is the same as C4.5 [17]. When the leaf contains no training examples, C4.5 produces a class distribution with the labeled class of the leaf having the probability 1, and all other classes having the probability 0. In this case, BOOST and SASC are different from C4.5. They estimate the class distribution using the training examples at the parent node of the empty leaf. The decision tree committee members vote by summing up the class distributions provided by all trees. The class with the highest score (sum of probabilities) wins the voting, and serves as the predicted class of BOOST and SASC for this example. There is no voting weight when summing up class distributions. Class distribution provides more detailed information than is obtained when each committee member votes for a single class, and this information is meaningful for committee voting.

There are three other approaches to voting. One is using the categorical predictions provided by all trees, without voting weights. In this case, each tree produces a single predicted class for an example. Then, the committee members vote by predicting the most frequent class returned by all trees.

The other two methods are the same as the two mentioned above but each tree is given a weight α_t for voting, which is a function of the performance of the decision tree on the training set, and is defined in Equation 1. The last of these alternatives, weighted voting of categorical predictions, corresponds to the original AdaBoost.M1. These three voting methods perform either worse than or similarly to the method that we use here [21].

3. Combining Boosting and SASC

Our objective in combining Boosting and SASC is to combine the advantages of both Boosting and SASC when generating different decision trees that partition the instance space differently. The combination strategy adopted here employs, when generating decision trees, both the stochastic selection of attribute subsets of SASC and the adaptive modification of the distribution of the training set of Boosting, that is, generating Stochastic Attribute Selection Committees with Boosting (SASCB).

Figure 4 presents the details of the SASCB algorithm, where C4.5SAS is defined in Figures 1 and 2. SASCB uses the same Boosting procedure as BOOST except that C4.5SAS instead of C4.5 is used as the base tree generator. Another difference is that when the weighted error rate of a tree is greater than 0.5 or equal to 0, SASCB does not change instance weights. Another tree will be built using the same distribution of the training set. Since the stochastic attribute subset selection process is involved, the tree generated this time should be different from the one created previously even though the same distribution of the training set is used. As in BOOST, the tree with an error rate greater than 0.5 is discarded,[5] while the tree with no errors on the training set is kept.

SASCB can be considered as introducing the stochastic attribute selection process into the generation of each tree in the Boosting process. It can also be thought of as adaptively modifying the distribution of the training set after the generation of each decision tree in the SASC process. SASCB uses the same voting method as BOOST and SASC, since it generally performs better than the other three voting approaches [21] as mentioned in Section 2.3.

[5]This step is also limited to 10 × T times, where T is the number of Boosting trials.

```
SASCB(Att, D, P, T)
  INPUT: Att: a set of attributes,
         D: a training set represented using Att and classes,
         P: a probability value,
         T: the number of trials.
  OUTPUT: a committee, H, containing T pruned trees.
  Set instance weight $w_1(x) = 1$ for each $x$ in $D$
  $H_1$ := C4.5SAS(Att, D, $w_1$, 1)
  $t$ := 2
  WHILE ($t <= T$)
  {  $D'$ := training cases in $D$ that are misclassified
        by $H_{(t-1)}$
     $\epsilon_{(t-1)} = \frac{1}{|D|} \sum_{x \in D'} w_{t-1}(x)$
     IF ($\epsilon_{(t-1)} > 1/2$)
        $t := t - 1$
     ELSE IF ($\epsilon_{(t-1)} \neq 0$)
        Calculate $w_t(x)$, the weight of each $x$ in
          $D$, from $w_{(t-1)}(x)$ using Equation 1
          and renormalize these weights so that
          they sum to $|D|$
     $H_t$ := C4.5SAS(Att, D, $w_t$, P)
     $t := t + 1$
  }
  RETURN H
```

Figure 4. The SASCB learning algorithm

4. Experiments

We conjectured that combining Boosting and SASC can further reduce the error rate of decision tree learning. In this section, we evaluate SASCB using experiments to examine whether it can benefit from the combination with respect to further reducing the error rate of C4.5. We also explore whether SASCB can outperform BOOST and SASC in terms of lower error rate. SASCB is compared with C4.5, BOOST, and SASC, using error rate as the primary performance metric.

4.1. Experimental domains and methods

Forty natural domains from the UCI machine learning repository [16] are used. They include all the domains used by Quinlan [18] for studying Boosting. This test suite covers a wide variety of different domains with respect to dataset size, the number of classes, the number of attributes, and types of attribute.

In every domain, two stratified 10-fold cross-validations [5, 14] were carried out for each algorithm. The result reported for each algorithm in each domain is an average value over 20 trials. All the algorithms are run on the same training and test set partitions with their default option settings. All BOOST, SASC, and SASCB use probabilistic predictions (without voting weights) for voting to decide the final classification. Schapire *et al.* [20] show that the test accuracy of Boosting increases as T increases even after the training error reaches zero. It is interesting to see the performance improvement that can be achieved with two orders of magnitude increase in computation. Therefore, the number of trials (the parameter T) is set at 100 in the experiments for all BOOST, SASC, and SASCB. The probability of each attribute being selected into the subset (the parameter P) is set at the default, 33%, for both SASC and SASCB.

4.2. Comparing SASCB with C4.5, BOOST, and SASC

Table 1 shows the error rates of the four algorithms. To facilitate pairwise comparisons among the four algorithms, error ratios are derived from Table 1 and presented in Table 2. An error ratio, for example for BOOST vs C4.5, presents a result for BOOST divided by the corresponding result for C4.5 – a value less than 1 indicates an improvement due to BOOST. To compare the error rates of two algorithms in a domain, a two-tailed pairwise t-test 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 2, **boldface** (*italic*) font, for example for BOOST vs C4.5, indicates that BOOST is significantly more (less) accurate than C4.5. The last row in Table 2 presents the significance levels of a one-tailed pairwise sign-test for comparing the number of wins, ties, and losses between the error rates of the corresponding two algorithms in the 40 domains.

From Tables 1 and 2, we have the following observations.

- By combining Boosting and SASC, on average SASCB further reduces the error rate of C4.5. A one-tailed pairwise sign-test shows that the number of domains for which SASCB reduces error is significant at a level better than 0.0001.

  While BOOST and SASC reduce the average error rate of C4.5 from 19.18% to 15.97% and 16.10% respectively, SASCB further reduces it to 15.76%. The average relative error reduction of SASCB over C4.5 is 22%. It is 20% and 16% for BOOST and SASC respectively.

  SASCB is significantly more accurate than C4.5 in 27 out of the 40 domains, and significantly less ac-

Table 1. Error rates (%)

Domain	Dataset size	C4.5	Boost	Sasc	SascB
Annealing	898	7.40	4.90	5.85	4.12
Audiology	226	21.39	15.41	18.73	15.19
Automobile	205	16.31	13.42	14.35	15.88
Breast (W)	699	5.08	3.22	3.44	3.08
Chess (KR-KP)	3169	0.72	0.36	0.67	0.36
Chess (KR-KN)	551	8.89	3.54	9.26	4.09
Credit (Aust)	690	14.49	13.91	14.71	14.20
Credit (Ger)	1000	29.40	25.45	25.10	25.15
Echocardiogram	131	37.80	36.24	37.01	39.20
Glass	214	33.62	21.09	25.27	21.99
Heart (C)	303	22.07	18.80	16.65	18.63
Heart (H)	294	21.09	21.25	18.88	21.09
Hepatitis	155	20.63	17.67	18.40	15.79
Horse colic	368	15.76	19.84	17.39	19.43
House votes 84	435	5.62	4.82	4.59	4.25
Hypo	3772	0.46	0.32	0.46	0.36
Hypothyroid	3163	0.71	1.14	0.76	0.98
Image	2310	2.97	1.58	2.06	1.58
Iris	150	4.33	5.67	5.00	5.67
Labor	57	23.67	10.83	18.83	9.83
LED 24	200	36.50	32.75	29.00	32.50
Letter	20000	12.16	2.95	3.74	2.76
Liver disorders	345	35.36	28.88	29.90	29.47
Lung cancer	32	57.50	53.75	45.83	53.75
Lymphography	148	21.88	16.86	18.48	16.50
NetTalk(Letter)	5438	25.88	22.14	21.98	19.91
NetTalk(Ph)	5438	18.97	16.01	18.03	14.60
NetTalk(Stress)	5438	17.25	11.91	12.44	11.30
Pima	768	23.97	26.57	23.76	26.43
Postoperative	90	29.44	38.89	28.89	38.89
Primary tumor	339	59.59	55.75	54.72	55.02
Promoters	106	17.50	4.68	7.09	4.73
Sick	3772	1.30	0.92	1.42	1.04
Solar flare	1389	15.62	17.57	15.70	17.57
Sonar	208	26.43	14.64	16.32	13.93
Soybean	683	8.49	6.22	5.42	5.64
Splice junction	3177	5.81	4.80	4.50	3.65
Vehicle	846	28.50	22.40	25.12	22.40
Waveform-21	300	23.83	18.33	19.83	17.50
Wine	178	8.96	3.35	4.48	1.96
average		19.18	15.97	16.10	15.76

Table 2. Error rate ratios

Domain	Boost	Sasc	SascB	SascB vs	
	vs C4.5			Boost	Sasc
Annealing	**.66**	**.79**	**.56**	.84	**.70**
Audiology	**.72**	**.88**	**.71**	.99	**.81**
Automobile	**.82**	.88	.97	1.18	1.11
Breast (W)	**.63**	**.68**	**.61**	.96	.90
Chess (KR-KP)	**.50**	.93	**.50**	1.00	**.54**
Chess (KR-KN)	**.40**	1.04	**.46**	1.16	**.44**
Credit (Aust)	.96	1.02	.98	1.02	.97
Credit (Ger)	**.87**	**.85**	**.86**	.99	1.00
Echocardiogram	.96	.98	1.04	1.08	1.06
Glass	**.63**	**.75**	**.65**	1.04	.87
Heart (C)	**.85**	**.75**	**.84**	.99	1.12
Heart (H)	1.01	.90	1.00	.99	1.12
Hepatitis	.86	**.89**	**.77**	.89	.86
Horse colic	*1.26*	1.10	*1.23*	.98	1.12
House votes 84	.86	.82	**.76**	.88	.93
Hypo	**.70**	1.00	.78	1.12	**.78**
Hypothyroid	*1.61*	1.07	*1.38*	.86	*1.29*
Image	**.53**	**.69**	**.53**	1.00	**.77**
Iris	1.31	1.15	1.31	1.00	1.13
Labor	**.46**	.80	**.42**	.91	**.52**
LED 24	.90	**.79**	.89	.99	*1.12*
Letter	**.24**	**.31**	**.23**	**.94**	**.74**
Liver disorders	**.82**	**.85**	**.83**	1.02	.99
Lung cancer	.93	**.80**	.93	1.00	1.17
Lymphography	**.77**	.84	**.75**	.98	.89
NetTalk(Letter)	**.86**	**.85**	**.77**	**.90**	**.91**
NetTalk(Ph)	**.84**	**.95**	**.77**	**.91**	**.81**
NetTalk(Stress)	**.69**	**.72**	**.66**	**.95**	**.91**
Pima	*1.11*	.99	*1.10*	.99	*1.11*
Postoperative	*1.32*	.98	*1.32*	1.00	*1.35*
Primary tumor	**.94**	**.92**	**.92**	.99	1.01
Promoters	**.27**	**.41**	**.27**	1.01	.67
Sick	**.71**	1.09	**.80**	1.13	**.73**
Solar flare	*1.12*	1.01	*1.12*	1.00	*1.12*
Sonar	**.55**	**.62**	**.53**	.95	.85
Soybean	**.73**	**.64**	**.66**	.91	1.04
Splice junction	**.83**	**.77**	**.63**	**.76**	**.81**
Vehicle	**.79**	**.88**	**.79**	1.00	**.89**
Waveform-21	**.77**	**.83**	**.73**	.95	**.88**
Wine	**.37**	**.50**	**.22**	.59	**.44**
average	.80	.84	.78	.97	.91
p. of wtl	< .0001	< .0001	< .0001	.0068	.0769

curate in 5 domains. BOOST is significantly more accurate than C4.5 in 27 domains, and significantly less accurate in 5 domains. SASC is significantly more accurate than C4.5 in 23 domains, and significantly less accurate in no domains. This indicates that the performance of SASCB and BOOST is more variable than that of SASC.

- On average, SASCB outperforms both BOOST and SASC.

  SASCB achieves the lowest error rate among the three committee learning algorithms in 18 out of the 40 domains. The error rate of SASCB is between those of BOOST and SASC in 20 other domains. Only in 2 domains, does SASCB obtain higher error rates than both BOOST and SASC.

  SASCB is significantly more accurate than BOOST in 5 domains, and significantly less accurate in no domains (one-tailed sign-test, $p = 0.0313$). It is significantly more accurate than SASC in 16 domains, and significantly less accurate in 5 domains (one-tailed sign-test, $p = 0.0133$).

  The average error rate of SASCB is 0.21 and 0.34 percentage points lower than those of BOOST and SASC respectively in the 40 domains. The average relative error reductions of SASCB over BOOST and SASC are 3% and 9% respectively. A one-tailed pairwise sign-test over the error rates in the 40 domains shows that SASCB has lower error rate more frequently than BOOST at a significance level of 0.0068. The significance level of the equivalent test between SASCB and SASC is 0.0769. This significance level is not better than 0.05, preventing us from reaching the firm conclusion that SASCB has a general advantage over SASC. However, the balance of evidence available at this time, specifically that the former achieves a big average relative error reduction over the latter (9%) and the former significantly more frequently outperforms the latter when ignoring insignificant error differences as shown in the previous paragraph, suggests that in the absence of evidence to the contrary SASCB should be considered the method of choice.

- SASCB follows the performance trend of BOOST.

  SASCB performs very similarly to BOOST in terms of lower/higher error rate than C4.5, although the extent to which the error rate of C4.5 is decreased/increased is different for SASCB and BOOST. This indicates that the Boosting component has a stronger influence on the performance of SASCB than the SASC component. This is not surprising since the Boosting component is the driving mechanism in the outer loop of SASCB. The SASC component helps to produce more diverse classifiers, which most of the time reduces the error rate further.

5. Conclusions

This paper has described a novel classifier committee learning method, SASCB, for decision tree learning. It combines the Boosting technique and the stochastic attribute selection committee technique. SASCB generates different trees to form a committee by both stochastically varying the set of attributes available for creating a test at each decision node and adaptively modifying the distribution of the training set.

Our analysis suggests that the Boosting component has a stronger influence on the performance of SASCB than the SASC component, since the Boosting component is the driving mechanism in the outer loop of SASCB. The SASC component helps SASCB to produce more diverse classifiers, which most of the time reduces the error rate further. The implication of our results is that methods for increasing the model diversity of other base learning algorithms such as rule learners and neural networks used in classifier committee learning algorithms such as Boosting should be investigated and employed.

The results of experiments with a representative collection of natural domains suggest that SASCB gains advantage from both Boost and SASC. It further significantly reduces the error rate of decision tree learning, and decreases, on average, the error rate of Boosting and SASC by 3% and 9% respectively. This further indicates that the Boosting and SASC techniques reduce the error rate of decision tree learning through different agencies.

6. Acknowledgments

The authors are grateful to J. Ross Quinlan for providing C4.5.

References

[1] K. Ali. *Learning Probabilistic Relational Concept Descriptions.* Ph.d. thesis, Dept of Info. and Computer Science, Univ. of California, Irvine, 1996.

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, Boosting, and variants. To appear in *Machine Learning* (available at: http://reality.sgi.com/ronnyk/vote.ps.gz), 1998.

[3] L. Breiman. Arcing classifiers. Technical report, Department of Statistics, University of California, Berkeley, CA (available at: http://www.stat.Berkeley.EDU/users/breiman/), 1996.

[4] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification And Regression Trees*. Belmont, CA: Wadsworth, 1984.

[6] P. Chan, S. Stolfo, and D. Wolpert. *Working Notes of AAAI Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, 1996. (available at http://www.cs.fit.edu/~imlm/papers.html), Portland, Oregon.

[7] T. Dietterich. Machine learning research. *AI Magazine*, 18:97–136, 1997.

[8] T. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[9] T. Dietterich and E. Kong. Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical report, Dept of Computer Science, Oregon State University, Corvallis, Oregon (available at ftp://ftp.cs.orst.edu/pub/tgd/papers/tr-bias.ps.gz), 1995.

[10] P. Domingos. Why does bagging work? a Bayesian account and its implications. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 155–158. AAAI Press, 1997.

[11] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121:256–285, 1996.

[12] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to Boosting. Unpublished manuscript (available at: http://www.research.att.com/~yoav), 1996.

[13] Y. Freund and R. Schapire. Experiments with a new Boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156. San Francisco, CA: Morgan Kaufmann, 1996.

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

[15] S. Kwok and C. Carter. Multiple decision trees. In R. Schachter, T. Levitt, L. Kanal, and J. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 327–335. Elsevier Science, 1990.

[16] C. Merz and P. Murphy. UCI repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: Univ of California, Dept of Info and Computer Science, 1997.

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

[18] J. R. Quinlan. Bagging, Boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730. Menlo Park, CA: AAAI Press, 1996.

[19] R. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[20] R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 322–330. San Francisco, CA: Morgan Kaufmann, 1997.

[21] Z. Zheng and G. Webb. Stochastic attribute selection committees. In *Proceedings of the Eleventh Australian Joint Conference on Artificial Intelligence*. Berlin: Springet-Verlag, 1998.