Pre-publication draft of a paper which appeared in the Proceedings of the Tenth Australian Joint Conference on Artificial Intelligence (AI 98), pp 321-332.
Publisher: Springer-Verlag

Stochastic Attribute Selection Committees

Zijian Zheng and Geoffrey I. Webb

School of Computing and Mathematics
Deakin University, Geelong
Victoria 3217, Australia
{zijian,webb}@deakin.edu.au

Technical Report (TR C98/08)
March, 1998

Abstract. Classifier committee learning methods generate multiple classifiers to form a committee by repeatedly applying a single base learning algorithm. The committee members vote to decide the final classification. Two such methods, Bagging and Boosting, have shown great success with decision tree learning. They create different classifiers by modifying the distribution of the training set. This paper studies a different approach: the Stochastic Attribute Selection Committee learning method with decision tree learning. It generates classifier committees by stochastically modifying the set of attributes but keeping the distribution of the training set unchanged. An empirical evaluation of a variant of this method, namely SASC, in a representative collection of natural domains shows that the SASC method can significantly reduce the error rate of decision tree learning. On average SASC is more accurate than Bagging and less accurate than Boosting, although a one-tailed sign-test fails to show that these differences are significant at a level of 0.05. In addition, it is found that, like Bagging, SASC is more stable than Boosting in terms of less frequently obtaining significantly higher error rates than C4.5 and obtaining lower error rate increases over C4.5. Moreover, like Bagging, SASC is amenable to parallel and distributed processing while Boosting is not.

Keywords: machine learning, decision tree learning, inductive learning, committee learning, datamining

1 Introduction

Classifier committees have been the focus of much recent attention (Freund, 1996; Freund and Schapire, 1996a; 1996b; Quinlan, 1996; Breiman, 1996a; 1996b; Dietterich and Kong, 1995; Ali, 1996; Chan, Stolfo, and Wolpert, 1996; Ali and Pazzani, 1996; Schapire, Freund, Bartlett, and Lee, 1997; Domingos, 1997; Bauer and Kohavi, 1998). With this type of approach, a set of classifiers is generated using a single base learning algorithm to form a committee. The committee members vote to decide the final classification.

Given a training set described using a set of attributes, conventional classifier learning algorithms such as decision tree learning algorithms (Quinlan, 1993; Breiman, Friedman, Olshen, and Stone, 1984) 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, the committee has a lower error rate than the individual classifiers.

Bagging (Breiman, 1996a) and Boosting (Schapire, 1990; Freund, 1996; Freund and Schapire, 1996a; 1996b; Schapire *et al.*, 1997), as two representative methods of this type, can significantly decrease the error rate of decision tree learning (Quinlan, 1996; Freund and Schapire, 1996b; Bauer and Kohavi, 1998). 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 stochastically changes the distribution of the training set and uses equal weight voting. Although Boosting is generally more accurate than Bagging, the performance of Boosting is more variable than that of Bagging (Quinlan, 1996; Bauer and Kohavi, 1998). Given an integer T as the committee size, both Boosting and Bagging need approximately T times as long as their base learning algorithm does for learning a single classifier. However, Bagging has an advantage over Boosting. That is, it is amenable to parallel and distributed processing while Boosting is not, since the generation of each committee member, a classifier, is independent for the former while it must occur sequentially for the latter. This makes Bagging appropriate for parallel machine learning and datamining.

While much recent attention has focused on Boosting and Bagging, other classifier committee learning approaches have also been developed, including generating multiple trees by manually changing learning parameters (Kwok and Carter, 1990), error-correcting output codes (Dietterich and Bakiri, 1995), generating different classifiers by randomizing the base learning process (Dietterich and Kong, 1995; Ali, 1996), learning option trees (Buntine, 1990; Kohavi and Kunz, 1997), training a committee of neural networks by manually selecting attribute subsets (Cherkauer, 1996; Tumer and Ghosh, 1996), learning naive Bayesian classifier committees by randomly choosing attribute subsets (Zheng, 1998), creating Gaussian classifier committees by varying attribute sets (Asker and Maclin, 1997), and creating committees for first-order learning by adding random selection of conditions to FOIL (Ali and Pazzani, 1996). Finally, dif-

ferent base learning algorithms can be used for learning different classifiers in committees (Wolpert, 1992; Zhang, Mesirov, and Waltz, 1992). A collection of recent research in this area and reviews of related methods can be found in Chan, Stolfo, and Wolpert (1996), Dietterich (1997), and Ali (1996).

In contrast to Bagging and Boosting, this paper studies an alternative approach to generating different classifiers to form a committee, namely **SASC** (**S**tochastic **A**ttribute **S**election **C**ommittees). SASC builds different classifiers by stochastically modifying the set of attributes considered during induction, while the distribution of the training set is kept unchanged. Two variants of this method which have been proposed are Dietterich and Kong's (1995) randomization trees and Ali's (1996) decision tree ensembles. The former generates decision tree committees through repeatedly applying a base learning algorithm that is derived by modifying C4.5 (Quinlan, 1993) to choose randomly among the top 20 tests in terms of information gain ratio at each decision node. The latter employs a very similar technique. At each decision node when building a tree, it chooses a test from those tests whose entropy is within 1/0.8 of the entropy of the best test. The probability that a test, s, in this set is chosen is proportional to 1/Entropy(s). Dietterich and Kong (1995) compare the randomization tree method with Bagging but only in 5 domains. Ali (1996) only compares the decision tree ensemble method with its base decision tree generator.

In this paper, we use another variant of this stochastic approach for decision tree committee learning to empirically compare the SASC approach with Boosting and Bagging. Although these three variants may perform differently, we do not expect that their performance differences are significant. The objective of this study is to empirically investigate the advantages and disadvantages of committee learning methods by stochastically changing attribute sets over committee learning methods by varying the distribution of the training set. We do not claim that the variant used here is better than the other two variants mentioned above. Indeed, our (untested) expectation is that all the three variants will offer comparable performance.

The following section describes the new variant of the SASC method. The implemented algorithm is called SASC. Section 3 reports experiments for evaluating the SASC algorithm. Finally, we conclude with future work.

2 SASC for Decision Tree Learning

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 (Quinlan, 1993). 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.

C4.5SAS(*Att*, *D*, *P*)
INPUT: *Att*: a set of attributes,
D: a training set represented using *Att* and classes,
P: a probability value.
OUTPUT: a pruned tree.
C := the majority class in *D*
RawTree := **Grow-Tree-SAS**(*Att*, *D*, *C*, *P*)
PrunedTree := **Prune-Tree**(*RawTree*, *Att*, *D*)
RETURN *PrunedTree*

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

We use C4.5 (Quinlan, 1993) with the modifications described below as the base classifier learning algorithm in our stochastic attribute selection committee learning algorithm, SASC, although any conventional decision tree learning algorithm can be used. When building a decision node, by default C4.5 uses the information gain ratio to search for the best attribute to form a test (Quinlan, 1993). To force C4.5 to generate different trees using the same training set, we modify C4.5 by stochastic restriction of the attributes available for selection at a decision node. This is implemented by using a probability parameter P.[1] 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 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

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

```
Grow-Tree-SAS(Att, D, C, P)
   INPUT: Att: a set of attributes,
          D: a training set,
          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)
        IF (Test_best is reasonable (with a positive test evaluation function value))
        {  Use Test_best to partition D into n subsets D_1, D_2, ..., D_n, 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, C, P),
              Grow-Tree-SAS(Att, D_2, C, P),
              ... ,
              Grow-Tree-SAS(Att, D_n, C, P)
        }
        ELSE
           RETURN a leaf node labeled with C
      }
   }
```

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

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 as the tree generated by C4.5. It is worth mentioning that SASC is amenable to parallel and distributed processing, since the generation of each tree in SASC is independent from that of another.

At the classification stage, for a given example, SASC makes 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.[2] With this method, each decision tree returns a distribution over classes that the example belongs to. This is performed by tracing the exam-

[2] The default setting of the SASC algorithm.

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, consisting of T trees.
H_1 := **C4.5SAS**(Att, D, 1)
FOR each t from 2 to T
{ H_t := **C4.5SAS**(Att, D, P) }
RETURN H

Fig. 3. The SASC learning algorithm

ple 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 (Quinlan, 1993). When the leaf contains no training examples,[3] 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, SASC is different from C4.5. It estimates 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 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 predicts 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 later.[4] These three voting methods perform either worse than or similarly to the method that we use here.[5] This issue will be addressed in Section 3.4.

[3] For multi-branch trees created by C4.5, some leaves may contain no training instances (Quinlan, 1993).

[4] The weight of each training example at any trial t is 1 when computing α_t in SASC.

[5] Quinlan (1996) uses categorical predictions with the confidence with which a tree classifies a test instance as the weight of this tree for voting in the Boosted C4.5 algorithm. In effect, this treatment is similar to using probabilistic predictions without weights discussed here.

3 Experiments

In this section, we use experiments in a representative collection of natural domains to explore whether the SASC algorithm can significantly reduce the error rate of C4.5. SASC is also compared with a Boosting algorithm, namely BOOST, and a Bagging algorithm, namely BAG. The reason for comparing SASC with Boosting and Bagging is, as mentioned before, to explore the advantages and disadvantages of generating decision tree committees by stochastically changing the set of attributes over that by changing the distribution of the training set with respect to reducing the error rate of decision tree learning. At the current stage, the prediction accuracy is our primary concern about classifier committee learning.

3.1 The Comparison Algorithms: BOOST and BAG

BOOST is our implementation of the Boosting algorithm with decision tree learning. It follows the Boosted C4.5 algorithm (AdaBoost.M1) (Quinlan, 1996) but uses a new Boosting equation as shown in Equation 1, derived from Schapire *et al.* (1997).

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 (Quinlan, 1993). 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,[6] 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.

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

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.

The primary idea of Bagging (Breiman, 1996a) is to generate a committee of classifiers with each from a bootstrap sample of the original training set. BAG, our implementation of Bagging, uses C4.5 (Quinlan, 1993) as its base classifier learning algorithm.

Given a committee size T and a training set D consisting of m instances, BAG generates $T - 1$ bootstrap samples with each being created by uniformly sampling m instances from D with replacement. It, then, builds one decision tree using C4.5 from each bootstrap sample. Another tree is created from the original training set.

[6] This step is limited to $10 \times T$ times.

As SASC, BOOST and BAG also have four voting methods to decide the final classification. The first one is using the probabilistic predictions produced by all H_ts, without voting weights. The second one is using the categorical predictions provided by all H_ts, without voting weights. This voting method corresponds to the method used in the original Bagging (Breiman, 1996a). The other two methods are the same as these two but each tree H_t is given a weight α_t for voting. The last of these alternatives, weighted voting of categorical predictions, corresponds to the original AdaBoost.M1 (Schapire, 1990; Freund and Schapire, 1996a; 1996b; Freund, 1996; Schapire *et al.*, 1997). The first method is set as the default in BOOST and BAG, since it performs either better than or similarly to the others on average. We will empirically compare these four voting methods later in this section.

3.2 Experimental Domains and Methods

Forty natural domains from the UCI machine learning repository (Merz and Murphy, 1997) are used. They include all the domains used by Quinlan (1996) for studying Boosting and Bagging. Table 1 summarizes the characteristics of these domains, including dataset size, the number of classes, the number of continuous attributes, and the number of discrete attributes. 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 attributes.

In every domain, two stratified 10-fold cross-validations (Kohavi, 1995) are carried out for each algorithm. The result of each algorithm in each domain reported 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 except when otherwise indicated. All of BOOST, BAG, and SASC use probabilistic predictions (without voting weights) for voting to decide the final classification in the following subsection. Schapire *et al.* (1997) 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 for BOOST, BAG, and SASC. The probability of each attribute being selected into the subset (the parameter P) is set at the default, 33%, for SASC.

3.3 Comparing SASC with BOOST, BAG, and C4.5

Table 2 shows the error rates of the four algorithms. To facilitate the pairwise comparisons among these algorithms, error ratios are derived from Table 2 and presented in Table 3. 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 3, **boldface** (*italic*)

Table 1. Description of learning tasks

Domain	Size	No. of Classes	No. of Attributes	
			Cont	Discr
Annealing	898	6	6	32
Audiology	226	24	0	69
Automobile	205	7	15	10
Breast cancer (W)	699	2	9	0
Chess (KR-KP)	3169	2	0	36
Chess (KR-KN)	551	2	0	39
Credit (Aust)	690	2	6	9
Credit (Ger)	1000	2	7	13
Echocardiogram	131	2	6	1
Glass	214	6	9	0
Heart (C)	303	2	13	0
Heart (H)	294	2	13	0
Hepatitis	155	2	6	13
Horse colic	368	2	7	15
House votes 84	435	2	0	16
Hypo	3772	5	7	22
Hypothyroid	3163	2	7	18
Image	2310	7	19	0
Iris	150	3	4	0
Labor	57	2	8	8
LED 24	200	10	0	24
Letter	20000	26	16	0
Liver disorders	345	2	6	0
Lung cancer	32	3	0	56
Lymphography	148	4	0	18
NetTalk(Letter)	5438	163	0	7
NetTalk(Phoneme)	5438	52	0	7
NetTalk(Stress)	5438	5	0	7
Pima	768	2	8	0
Postoperative	90	3	1	7
Primary tumor	339	22	0	17
Promoters	106	2	0	57
Sick	3772	2	7	22
Solar flare	1389	2	0	10
Sonar	208	2	60	0
Soybean	683	19	0	35
Splice junction	3177	3	0	60
Vehicle	846	4	18	0
Waveform-21	300	3	21	0
Wine	178	3	13	0

Table 2. Error rates (%)

Domain	C4.5	Boost	Bag	Sasc
Annealing	7.40	4.90	5.73	5.85
Audiology	21.39	15.41	18.29	18.73
Automobile	16.31	13.42	17.80	14.35
Breast (W)	5.08	3.22	3.37	3.44
Chess (KR-KP)	0.72	0.36	0.59	0.67
Chess (KR-KN)	8.89	3.54	7.80	9.26
Credit (Aust)	14.49	13.91	13.84	14.71
Credit (Ger)	29.40	25.45	24.95	25.10
Echocardiogram	37.80	36.24	33.57	37.01
Glass	33.62	21.09	27.38	25.27
Heart (C)	22.07	18.80	18.45	16.65
Heart (H)	21.09	21.25	20.38	18.88
Hepatitis	20.63	17.67	18.73	18.40
Horse colic	15.76	19.84	15.77	17.39
House votes 84	5.62	4.82	4.71	4.59
Hypo	0.46	0.32	0.45	0.46
Hypothyroid	0.71	1.14	0.71	0.76
Image	2.97	1.58	2.62	2.06
Iris	4.33	5.67	5.00	5.00
Labor	23.67	10.83	14.50	18.83
LED 24	36.50	32.75	31.00	29.00
Letter	12.16	2.95	5.93	3.74
Liver disorders	35.36	28.88	27.43	29.90
Lung cancer	57.50	53.75	42.50	45.83
Lymphography	21.88	16.86	18.50	18.48
NetTalk(Letter)	25.88	22.14	22.98	21.98
NetTalk(Ph)	18.97	16.01	17.33	18.03
NetTalk(Stress)	17.25	11.91	14.97	12.44
Pima	23.97	26.57	23.37	23.76
Postoperative	29.44	38.89	30.00	28.89
Primary tumor	59.59	55.75	55.46	54.72
Promoters	17.50	4.68	9.32	7.09
Sick	1.30	0.92	1.18	1.42
Solar flare	15.62	17.57	15.91	15.70
Sonar	26.43	14.64	21.12	16.32
Soybean	8.49	6.22	6.80	5.42
Splice junction	5.81	4.80	5.18	4.50
Vehicle	28.50	22.40	25.30	25.12
Waveform-21	23.83	18.33	19.67	19.83
Wine	8.96	3.35	5.29	4.48
average	19.18	15.97	16.35	16.10

Table 3. Error rate ratios

Domain	Boost	Bag	Sasc	Sasc vs	
	vs C4.5			Boost	Bag
Annealing	**.66**	**.77**	**.79**	1.19	1.02
Audiology	**.72**	**.86**	**.88**	*1.22*	1.02
Automobile	**.82**	1.09	.88	1.07	**.81**
Breast (W)	**.63**	**.66**	**.68**	1.07	1.02
Chess (KR-KP)	**.50**	.82	.93	*1.86*	1.14
Chess (KR-KN)	**.40**	.88	1.04	*2.62*	*1.19*
Credit (Aust)	.96	.96	1.02	1.06	1.06
Credit (Ger)	**.87**	**.85**	**.85**	.99	1.01
Echocardiogram	.96	.89	.98	1.02	1.10
Glass	**.63**	**.81**	**.75**	*1.20*	.92
Heart (C)	**.85**	**.84**	**.75**	.89	**.90**
Heart (H)	1.01	.97	.90	.89	.93
Hepatitis	.86	.91	**.89**	1.04	.98
Horse colic	*1.26*	1.00	1.10	.88	1.10
House votes 84	.86	.84	.82	.95	.97
Hypo	**.70**	.98	1.00	*1.44*	1.02
Hypothyroid	*1.61*	1.00	1.07	**.67**	1.07
Image	**.53**	.88	**.69**	*1.30*	**.79**
Iris	1.31	1.15	1.15	.88	1.00
Labor	**.46**	**.61**	.80	*1.74*	1.30
LED 24	.90	**.85**	**.79**	**.89**	.94
Letter	**.24**	**.49**	**.31**	*1.27*	**.63**
Liver disorders	**.82**	**.78**	**.85**	1.04	1.09
Lung cancer	.93	**.74**	**.80**	.85	1.08
Lymphography	**.77**	**.85**	.84	1.10	1.00
NetTalk(Letter)	**.86**	**.89**	**.85**	.99	**.96**
NetTalk(Ph)	**.84**	**.91**	**.95**	*1.13*	*1.04*
NetTalk(Stress)	**.69**	**.87**	**.72**	1.04	**.83**
Pima	*1.11*	.97	.99	**.89**	1.02
Postoperative	*1.32*	1.02	.98	**.74**	.96
Primary tumor	**.94**	**.93**	**.92**	.98	.99
Promoters	**.27**	**.53**	**.41**	1.51	.76
Sick	**.71**	.91	1.09	*1.54*	*1.20*
Solar flare	*1.12*	1.02	1.01	**.89**	.99
Sonar	**.55**	**.80**	**.62**	1.11	**.77**
Soybean	**.73**	**.80**	**.64**	.87	**.80**
Splice junction	**.83**	**.89**	**.77**	.94	**.87**
Vehicle	**.79**	**.89**	**.88**	*1.12*	.99
Waveform-21	**.77**	**.83**	**.83**	1.08	1.01
Wine	**.37**	**.59**	**.50**	1.34	.85
average	0.80	.86	.84	1.13	.98
w/t/l	33/0/7	34/1/5	32/1/7	16/0/24	21/1/18
p. of wtl	$< .0001$	$< .0001$	$< .0001$	.1341	.3746
significant w/t/l	27/8/5	23/17/0	23/17/0	5/24/11	9/28/3
p. of sign. wtl	$< .0001$	$< .0001$	$< .0001$	.1051	.0730

font, for example for BOOST vs C4.5, indicates that BOOST is significantly more (less) accurate than C4.5. The second last row in Table 3 presents the numbers of wins, ties, and losses between the error rates of the corresponding two algorithms in the 40 domains, and the significance levels of a one-tailed pairwise sign-test on these win/tie/loss records. The last row presents similar comparison results but treating insignificant wins and losses as ties.

From Tables 2 and 3, we have the following four observations.

(1) All the three committee learning algorithms can significantly reduce the error rate of the base tree learning algorithm.

The average error rate of C4.5 in the 40 domains is 19.18%. BOOST, BAG, and SASC reduce the average error rate to 15.97%, 16.35%, and 16.10% respectively. The average relative error reductions of these three committee learning algorithms over C4.5 in the 40 domains are 20%, 14%, and 16% respectively. A one-tailed pairwise sign-test shows that all these error reductions are significant at a level better than 0.0001.

(2) BOOST is more accurate than BAG on average, but BAG is more stable than BOOST in terms of less frequently obtaining significantly higher error rates than C4.5. This is consistent with previous findings (Quinlan, 1996; Bauer and Kohavi, 1998).

The average error rate of BAG in the 40 domains is 16.35%, while it is 15.97% for BOOST, 8% relatively lower, on average, than that of BAG. However, a one-tailed sign-test fails to show that the frequency of error reductions of BOOST over BAG in the 40 domains is significant (w/t/l = 24/0/16, $p = 0.1341$).

While BOOST significantly reduces the error of C4.5 in 27 out of the 40 domains, it also significantly increases the error of C4.5 in 5 domains. BAG achieves significantly lower error rates than C4.5 in 23 domains, but does not obtain any significantly higher error rate than C4.5 in any of these domains.

(3) On average, SASC is more accurate than BAG, but less accurate than BOOST.

SASC achieves 2% average relative error reduction over BAG in the 40 domains. The former obtains significantly lower error rates than the latter in 9 out of the 40 domains, and significantly higher error rates in 3 domains. SASC demonstrates its advantage over BAG in terms of lower error rate, although a one-tailed sign-test fails to show that the frequency of the error reductions in the 40 domains is significant at a level of 0.05.

The average error ratio of SASC over BOOST is 1.13 in the 40 domains. It might be thought a serious disadvantage of SASC that the average error ratio compared to BOOST is greater than 1. However, it is noticed that the average *accuracy ratio* of SASC against BOOST (the accuracy for SASC divided by the corresponding accuracy for BOOST) is 1.00. That is, on average SASC is as accurate as BOOST. This discrepancy arises because error ratio favors better performance at low error rates while accuracy ratio rather favors better performance at high error rates. The one-tailed sign-test shows that the frequency of error decreases is not significant over the 40 domains at a level of 0.05. In addition, it is found that SASC is likely to outperform BOOST when BOOST cannot obtain large error reductions over C4.5, for example, when the reduction is less than or equal to

Table 4. Average error rates (%) and ratios over C4.5 of BOOST, BAG, and SASC with different voting methods in the 40 domains

	prob.		cat.		prob. & w		cat. & w	
	err.	ratio	err.	ratio	err.	ratio	err.	ratio
BOOST	15.97	.80	16.38	.83	15.87	.80	16.16	.81
BAG	16.35	.86	16.33	.86	16.76	.87	16.75	.88
SASC	16.10	.84	16.41	.86	16.64	.86	16.44	.85

15%.

(4) SASC is more stable than BOOST

The performance stability of SASC is very similar to that of BAG, although SASC is more accurate than BAG on average. Both of them obtain significantly lower error rates than C4.5 in 23 domains and significantly higher error rates than C4.5 in no domains, whereas BOOST is significantly more accurate than C4.5 in 27 domains, but is significantly less accurate in 5 domains. In addition, the highest relative error increase of both SASC and BAG over C4.5 in the 40 domains is 15% which is not significant. It is 61% for BOOST, which is significant. These results indicate that, like BAG, SASC is more stable than BOOST in terms of less frequently obtaining significantly higher error rates than C4.5 and obtaining lower error rate increases over C4.5. Our analysis suggests that the stochastic component of SASC and BAG contributes to their stable behavior.

3.4 Voting Weights

As mentioned before, BOOST, BAG, and SASC can use four voting methods, namely probabilistic predictions (without voting weights), categorical predictions (without voting weights), probabilistic predictions with voting weights, and categorical predictions with voting weights, to make the final classification.[7] Table 4 presents the average error rates and the average error rate ratios over C4.5 of BOOST, BAG, and SASC with different voting methods in the 40 domains.

While the previous research on Boosting uses the categorical predictions with voting weights as voting method (the last method in the table) (Freund, 1996; Freund and Schapire, 1996b; Quinlan, 1996; Bauer and Kohavi, 1998), we found that the probabilistic predictions without voting weights (the first method in the table) method outperforms it with respect to either lower average error rate or higher average relative error reduction over C4.5. The average relative error reduction of the latter over the former is 2% in the 40 domains. However, a one-tailed sign-test fails to show that this error reduction is significant ($p = 0.0662$). Note that the probabilistic predictions with voting weights method achieves the lowest average error rate in the 40 domains among the four voting methods for

[7] The voting weight of a tree without training errors is set at a big value 1000000 for BOOST, BAG, and SASC in the experiments reported in this subsection.

BOOST. Nevertheless, the average error rate ratio of the probabilistic predictions without voting weights method against the probabilistic predictions with voting weights method is 0.99 in the 40 domains. A one-tailed sign-test shows that their differences are not significant ($p = 0.3601$). Therefore, we choose probabilistic predictions without voting weights as the voting method for BOOST in the experiments reported in the previous subsection. The categorical predictions without voting weights method obtains the highest average error rate and the lowest average relative error reduction over C4.5 in the 40 domains among the four voting methods for BOOST. It performs significantly worse than the probabilistic predictions without voting weights method using a one-tailed sign-test ($p = 0.0401$). Substituting the probabilistic predictions with voting weights results (the lowest average error rate for BOOST) for the BOOST results presented in Tables 2 and 3 does not significantly affect any of the comparative outcomes with respect to SASC (error ratio = 1.12, w/t/l = 16/1/23, $p = 0.1684$).

Bagging uses the categorical predictions without voting weights method for voting in the previous research (Breiman, 1996a; Quinlan, 1996; Bauer and Kohavi, 1998). Our experiments show that this voting method achieves the lowest average error rate in the 40 domains among the four methods for BAG. The probabilistic predictions without voting weights method obtains an average error rate just 0.02 percentage points higher than that of this method for BAG. A two-tailed sign-test shows that their differences in error rate are not significant ($p = 0.7283$) in the 40 domains. In addition, their average relative error reductions over C4.5 are the same in the 40 domains, and the average error ratio between them is 1.00. The other two voting methods perform worse than these two for BAG. A one-tailed sign-test shows that the probabilistic predictions without voting weights method is, on average, significantly more accurate than the categorical predictions with voting weights method for BAG in the 40 domains ($p = 0.0038$). A one-tailed sign-test shows that the error difference between any other pair of these four methods is not significant at a level of 0.05 for BAG. Therefore, BAG uses the probabilistic predictions without voting weights method in the previous subsection, which is the same as for BOOST.

For SASC, the probabilistic predictions without weights method achieves the lowest average error rate and the highest average relative error reduction over C4.5 in the 40 domains among the four voting methods. A one-tailed sign-test shows that this method is, on average, significantly more accurate than the categorical predictions with voting weights method ($p = 0.0288$). The error difference between any other pair of these four voting methods for SASC over the 40 domains is not significant at a level of 0.05 using a one-tailed sign-test.

In short, these experimental results suggest that using a function of the performance of a tree on the training set as the voting weight provide very marginal advantage for classifier committee learning algorithms including BOOST, BAG, and SASC. It seems that classifier committee learning algorithms with probabilistic predictions without weights for voting perform reasonably well, better than or equal to with the other voting methods on average. This is consistent with Quinlan's (1996) findings. We argue that when using categorical predictions for

voting, the information of the extent to which each committee member supports its vote (categorical prediction) is lost. If this information is taken into account, the committee may make a different decision. For example, committee member A gives 30%, 30%, and 40% support for decision D_1, D_2, and D_3 respectively. It is 25%, 35%, and 40% for member B; 10%, 80%, and 10% for member C. Suppose the committee has only these three members. If the voting with categorical predictions is used, the decision is D_3, since two members support it but with very low support. If taking account of detailed support from each member, the decision should be D_2, since it gets the highest total support from all the committee members. Therefore, the probabilistic prediction voting method can utilize this information, thus helping the committee to make better decisions.

4 Conclusions and Future Work

In this paper, we have studied the Stochastic Attribute Selection Committee learning method with decision tree learning by using a variant of it, namely SASC. This approach generates different trees by stochastically varying the set of attributes available for creating a test at each decision node, but keeping the distribution of the training set unchanged. Like Bagging, SASC is amenable to parallel and distributed processing while Boosting is not. This gives SASC an advantage over Boosting for parallel machine learning and datamining. Since Breiman (1996b) argues that the adaptive property makes Boosting work well, introducing some adaptive property to the change of the attribute set might further reduce the error rate of the SASC method. However, adaptive SASC may not be amenable to parallel processing.

We have conducted a set of experiments in a wide variety of natural domains. From the experiments, we found that using probabilistic predictions without weights for voting is more appropriate and provides more advantages for committee learning, including Boosting, Bagging, and SASC, (at least not worse) than using categorical predictions with or without a function of the performance of individual classifiers as voting weight. The results showed that on average SASC is more accurate than Bagging and less accurate than Boosting, although a one-tailed sign-test shows that these differences are not significant at a level of 0.05. In addition, it is found that, like Bagging, SASC is more stable than Boosting in terms of less frequently obtaining significantly higher error rates than C4.5 and obtaining lower error rate increases over C4.5. Our analysis suggests that the stochastic component of SASC and Bagging contributes to their stable behavior.

Acknowledgements

The authors are grateful to J. Ross Quinlan for providing C4.5. Thanks to Kai Ming Ting for his helpful comments.

References

Ali, K.M.: *Learning Probabilistic Relational Concept Descriptions.* PhD. Thesis, Dept of Info. and Computer Science, Univ. of California, Irvine (1996).

Ali, K.M. and Pazzani, M.J.: Error reduction through learning multiple descriptions. *Machine Learning* **24** (1996) 173-202.

Asker, L. and Maclin, R.: Ensembles as a sequence of classifiers. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence.* San Francisco, CA: Morgan Kaufmann (1997) 860-865.

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

Breiman, L.: Bagging predictors. *Machine Learning* **24** (1996a) 123-140.

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

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

Buntine, W.: *A Theory of Learning Classification Rules.* PhD. Thesis, School of Computing Science, University of Technology, Sydney (1990).

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

Cherkauer, K.J.: Human expert-level performance on a science image analysis task by a system using combined artificial neural networks. Chan, P., Stolfo, S., and Wolpert, D. (eds) *Working Notes of AAAI Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms* (available at http://www.cs.fit.edu/~imlm/papers.html), Portland, Oregon (1996) 15-21.

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

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

Dietterich, T.G.: Machine learning research. *AI Magazine* **18** (1997) 97-136.

Domingos, P.: Why does Bagging work? a Bayesian account and its implications. *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining.* AAAI Press (1997) 155-158.

Freund, Y.: Boosting a weak learning algorithm by majority. *Information and Computation* **121** (1996) 256-285.

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

Freund, Y. and Schapire, R.E.: Experiments with a new Boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning.* San Francisco, CA: Morgan Kaufmann (1996b) 148-156.

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

Kohavi, R. and Kunz, C.: Option decision trees with majority votes. *Proceedings of the Fourteenth International Conference on Machine Learning*. San Francisco, CA: Morgan Kaufmann (1997) 161-169.

Kwok, S.W. and Carter, C.: Multiple decision trees. Schachter, R.D., Levitt, T.S., Kanal, L.N., and Lemmer, J.F. (eds) *Uncertainty in Artificial Intelligence*. Elsevier Science (1990) 327-335.

Merz, C.J. and Murphy, P.M.: 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).

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

Quinlan, J.R.: Bagging, Boosting, and C4.5. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press (1996) 725-730.

Schapire, R.E.: The strength of weak learnability. *Machine Learning* **5** (1990) 197-227.

Schapire, R.E., Freund, Y., Bartlett, P., and Lee, W.S.: Boosting the margin: A new explanation for the effectiveness of voting methods. *Proceedings of the Fourteenth International Conference on Machine Learning*. Morgan Kaufmann (1997) 322-330.

Tumer, K. and Ghosh, J.: Error correction and error reduction in ensemble classifiers. *Connection Science* **8** (1996) 385-404.

Wolpert, D.H.: Stacked generalization. *Neural Networks* **5** (1992) 241-259.

Zhang, X., Mesirrov, J.P., and Waltz, D.L.: Hybrid system for protein secondary structure prediction. *Journal of Molecular Biology* **225** (1992) 1049-1063.

Zheng, Z.: Naive Bayesian classifier committees. To appear in *Proceedings of the Tenth European Conference on Machine Learning (ECML-98)*. Berlin: Springet-Verlag (1998).