Pre-publication draft of a paper which appeared in the Proceedings of the 1998 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 98), pp 1133-1140.
Publisher: CSREA Press

Multiple Boosting: A Combination of Boosting and Bagging

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 C98/01)
February, 1998

Abstract *Classifier committee learning approaches have demonstrated great success in increasing the prediction accuracy of classifier learning, which is a key technique for datamining. These approaches generate several classifiers to form a committee by repeated application of a single base learning algorithm. The committee members vote to decide the final classification. It has been shown that Boosting and Bagging, as two representative methods of this type, can significantly decrease the error rate of decision tree learning. Boosting is generally more accurate than Bagging, but the former is more variable than the latter. In addition, Bagging is amenable to parallel or distributed processing, while Boosting is not. In this paper, we study a new committee learning algorithm, namely MB (Multiple Boosting). It creates multiple subcommittees by combining Boosting and Bagging. Experimental results in a representative collection of natural domains show that MB is, on average, more accurate than either Bagging or Boosting alone. It is more stable than Boosting, and is amenable to parallel or distributed processing. These characteristics make MB a good choice for parallel datamining.*

Keywords: Parallel datamining, Boosting, Bagging, Committee learning, Decision tree learning, Machine learning

1 Introduction

Classifier learning is a key technique for datamining. Prediction accuracy and computational requirements are two primary concerns with this type of learning. In order to improve the prediction accuracy of classifiers, classifier committee[1] learning techniques have been developed with great success [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]. 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.

Bagging [Breiman, 1996a] and Boosting [Schapire, 1990; Freund and Schapire, 1996a; 1996b; Freund, 1996; 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 deci-

[1] Committees are also referred to as ensembles [Dietterich, 1997].

sion 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 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], and learning option trees [Kohavi and Kunz, 1997]. A collection of recent research in this area and reviews of related methods can be found in [Chan *et al.*, 1996; Dietterich, 1997; Ali, 1996].

In this paper, we investigate an approach, namely **MB** (**M**ultiple **B**oosting),[2] to generating committees, which is more accurate than Bagging and is more stable than Boosting. MB creates multiple subcommittees by incorporating Bagging into Boosting using the multiboosting technique [Webb, 1998]. We expect that splitting one committee into multiple subcommittees, with each subcommittee being created from a bootstrap sample of the training set, can reduce the variability of Boosting, since the Boosting process is broken down into several small processes. In addition, we expect that introducing Bagging can further improve the accuracy of learned committees, since it increases the diversity and independence of committee members. Moreover, the new algorithm is amenable to parallel and distributed processing.

The next section presents the Boosting and Bagging techniques. Then, Section 3 describes the MB algorithm. Section 4 reports experimental results for evaluating MB. We summarize our findings in the final section.

2 Boosting and Bagging

MB can be considered as a combination of Boosting and Bagging. We briefly discuss Boosting and Bagging in this section. Their classification process is presented in Section 2.3. Further details about Boosting and Bagging are available elsewhere [Schapire, 1990; Quinlan, 1996; Schapire *et al.*, 1997; Bauer and Kohavi, 1998; Breiman, 1996a].

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)

[2] The idea of multiple Boosting was originally proposed by Webb [Webb, 1998].

[Quinlan, 1996] but uses a new Boosting equation as shown in Equation 1, derived from Schapire *et al.* [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,[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 Bagging

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.

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

2.3 Decision Making in BOOST and BAG

At the classification stage, for a given example, both BOOST and BAG 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. When the leaf contains no training examples,[4] BOOST and BAG 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 BAG 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. This voting method corresponds to the method used in the original Bagging [Breiman, 1996a].

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 cat-

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

egorical predictions, corresponds to the original AdaBoost.M1. These three voting methods perform either worse than or similarly to the method that we use here.[5] We will address this issue in Section 4.3.

3 MB: A Combination of Boosting and Bagging

Figure 1 presents the details of the MB algorithm. It results from incorporating BAG into BOOST. MB generates N subcommittees. This process can be parallelized, since the generation of one subcommittee is independent from the generation of another. Each subcommittee contains S decision trees built using the BOOST procedure described in the previous section. The generation of the first subcommittee (or one of the subcommittees if using parallel or distributed processing) starts from the initial training set D with each training instance having the weight 1. The first tree in this subcommittee is the same one as that built by C4.5 using the entire training set. The generation of every other subcommittee starts from a bootstrap sample of D. A bootstrap sample is created by uniformly sampling $|D|$ instances from D with replacement. This is implemented through changing instance weights, resulting in 0 weights for instances not in the sample. Boosting propagates these values ensuring that these instances are not considered when inferring any member of the subcommittee. Note that MB differs from Webb's MULTIBOOST [Webb, 1998] by using bootstrap sampling in place of stochastic weighting at the start of the generation of each subcommittee. This difference is not expected to significantly affect performance. Bootstrap sampling rather than stochastic weighting is employed here to enable direct comparison with Bagging.

[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.

MB(D, S, N)
  INPUT: D: a training set,
  S: the size of each subcommittee,
  N: the number of subcommittees.
  OUTPUT: a committee, H, consisting of N subcommittees with each having S trees.
  Set T, the number of trials, $= S \times N$
  Set instance weight $w_1(x) = 1$ for each x in D
  $t := 1$
  WHILE ($t <= T$)
  { $H_t :=$ **C4.5**(D, w_t)
    $D' :=$ cases in D being misclassified by H_t
    $\epsilon_t = \frac{1}{|D|} \sum_{x \in D'} w_t(x)$
    IF ($\epsilon_t > 1/2$)
      Reset $w_t(x)$ using bootstrap sampling
    ELSE
      IF ((t modulus $S = 0$) or ($\epsilon_t = 0$))
        Set $w_{t+1}(x)$ using bootstrap sampling
        $t := t + 1$
      ELSE
        Calculate $w_{t+1}(x)$ from $w_t(x)$ using Equation 1 and renormalize these weights so that they sum to $|D|$
        $t := t + 1$
  }
  RETURN H

Figure 1: The MB learning algorithm

At the classification stage, all the members of all the subcommittees generated by MB vote to predict a class for a given instance. MB uses the same default voting method as BOOST and BAG, since it, on average, performs better than the other three voting approaches as mentioned in Section 2.3.

4 Experiments

In this section, we empirically evaluate MB to examine whether incorporating Bagging into Boosting can increase stability and average accuracy of learned committees. It is compared with BOOST and BAG. C4.5, the base decision tree learning algorithm of these committee learning algorithms, is used as the base line for the comparison.

4.1 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] 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, except when otherwise indicated. Pruned trees are used for all the algorithms. All of BOOST, BAG, and MB use probabilistic predictions (without voting weights) for voting to decide the final classification in the following subsection. The number of trials (T) is set at 100 for BOOST and BAG. The subcommittee size and the number of subcommittees are set at 5 and 20 respectively, resulting in 100 trees in total for MB.

Table 1: Description of learning tasks

Domain	Size	No. of Classes	No. of Att.	
			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

4.2 Comparing MB with BOOST, BAG, and C4.5

Table 2 shows the error rates of the four algorithms. To facilitate 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 2: Error rates (%)

Domain	C4.5	Boost	Bag	MB
Annealing	7.40	4.90	5.73	4.67
Audiology	21.39	15.41	18.29	15.88
Automobile	16.31	13.42	17.80	16.10
Breast (W)	5.08	3.22	3.37	3.08
Chess (KR-KP)	0.72	0.36	0.59	0.39
Chess (KR-KN)	8.89	3.54	7.80	5.27
Credit (Aust)	14.49	13.91	13.84	12.82
Credit (Ger)	29.40	25.45	24.95	23.90
Echocardiogram	37.80	36.24	33.57	31.68
Glass	33.62	21.09	27.38	24.33
Heart (C)	22.07	18.80	18.45	18.13
Heart (H)	21.09	21.25	20.38	19.20
Hepatitis	20.63	17.67	18.73	17.12
Horse colic	15.76	19.84	15.77	15.90
House votes 84	5.62	4.82	4.71	3.90
Hypo	0.46	0.32	0.45	0.33
Hypothyroid	0.71	1.14	0.71	0.82
Image	2.97	1.58	2.62	1.77
Iris	4.33	5.67	5.00	5.00
Labor	23.67	10.83	14.50	12.33
LED 24	36.50	32.75	31.00	32.00
Letter	12.16	2.95	5.93	3.45
Liver disorders	35.36	28.88	27.43	26.73
Lung cancer	57.50	53.75	42.50	47.08
Lymphography	21.88	16.86	18.50	16.86
NetTalk(Letter)	25.88	22.14	22.98	21.37
NetTalk(Ph)	18.97	16.01	17.33	15.22
NetTalk(Stress)	17.25	11.91	14.97	12.26
Pima	23.97	26.57	23.37	23.31
Postoperative	29.44	38.89	30.00	32.22
Primary tumor	59.59	55.75	55.46	55.02
Promoters	17.50	4.68	9.32	5.64
Sick	1.30	0.92	1.18	1.10
Solar flare	15.62	17.57	15.91	16.31
Sonar	26.43	14.64	21.12	19.68
Soybean	8.49	6.22	6.80	6.66
Splice junction	5.81	4.80	5.18	4.23
Vehicle	28.50	22.40	25.30	24.00
Waveform-21	23.83	18.33	19.67	18.00
Wine	8.96	3.35	5.29	3.07
average	19.18	15.97	16.35	15.42

Table 3: Error rate ratios

Domain	Boost	Bag	MB	MB vs	
	vs C4.5			Boost	Bag
Annealing	**.66**	**.77**	**.63**	.95	**.82**
Audiology	**.72**	**.86**	**.74**	1.03	**.87**
Automobile	**.82**	1.09	.99	1.20	.90
Breast (W)	**.63**	**.66**	**.61**	.96	.91
Chess (KR-KP)	**.50**	.82	**.54**	1.08	**.66**
Chess (KR-KN)	**.40**	.88	**.59**	*1.49*	**.68**
Credit (Aust)	.96	.96	.88	**.92**	**.93**
Credit (Ger)	**.87**	**.85**	**.81**	**.94**	.96
Echocardiogram	.96	.89	**.84**	**.87**	.94
Glass	**.63**	**.81**	**.72**	*1.15*	.89
Heart (C)	**.85**	**.84**	**.82**	.96	.98
Heart (H)	1.01	.97	.91	.90	.94
Hepatitis	.86	.91	**.83**	.97	.91
Horse colic	*1.26*	1.00	1.01	**.80**	1.01
House votes 84	.86	.84	**.69**	**.81**	**.83**
Hypo	**.70**	.98	**.72**	1.03	**.73**
Hypothyroid	*1.61*	1.00	*1.15*	**.72**	1.15
Image	**.53**	.88	**.60**	1.12	**.68**
Iris	1.31	1.15	1.15	.88	1.00
Labor	**.46**	**.61**	**.52**	1.14	.85
LED 24	.90	**.85**	**.88**	.98	1.03
Letter	**.24**	**.49**	**.28**	*1.17*	**.58**
Liver disorders	**.82**	**.78**	**.76**	.93	.97
Lung cancer	.93	**.74**	.82	.88	1.11
Lymphography	**.77**	**.85**	**.77**	1.00	.91
NetTalk(Letter)	**.86**	**.89**	**.83**	**.97**	**.93**
NetTalk(Ph)	**.84**	**.91**	**.80**	**.95**	**.88**
NetTalk(Stress)	**.69**	**.87**	**.71**	1.03	**.82**
Pima	*1.11*	.97	.97	**.88**	1.00
Postoperative	*1.32*	1.02	1.09	**.83**	1.07
Primary tumor	**.94**	**.93**	**.92**	.99	.99
Promoters	**.27**	**.53**	**.32**	1.21	.61
Sick	**.71**	.91	.85	*1.20*	.93
Solar flare	*1.12*	1.02	1.04	**.93**	1.03
Sonar	**.55**	**.80**	**.74**	*1.34*	.93
Soybean	**.73**	**.80**	**.78**	1.07	.98
Splice junction	**.83**	**.89**	**.73**	**.88**	**.82**
Vehicle	**.79**	**.89**	**.84**	*1.07*	.95
Waveform-21	**.77**	**.83**	**.76**	.98	.92
Wine	**.37**	**.59**	**.34**	.92	.58
average	.80	.86	.78	1.00	.89
w/t/l	33/0/7	34/1/5	35/0/5	24/1/15	33/1/6
p. of wtl	$< .0001$	$< .0001$	$< .0001$	.0998	$< .0001$
significant w/t/l	27/8/5	23/17/0	29/10/1	12/22/6	13/27/0
p. of sign. wtl	$< .0001$	$< .0001$	$< .0001$	.1189	.0001

Table 3, **boldface** (*italic*) 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, with MB achieving the lowest average error rate and the greatest average relative error reduction over C4.5.

The average error rate of MB over the 40 domains is 15.42%. It is 15.97% and 16.35% for BOOST and BAG respectively. While the average relative error reductions of BOOST and BAG over C4.5 in the 40 domains are 20% and 14% respectively, it is 22% for MB, the greatest one. 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 confirms 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, MB is more accurate than either BOOST or BAG alone.

MB achieves 11% average relative error reduction over BAG in the 40 domains. The former obtains significantly lower error rates than the latter in 13 out of the 40 domains, but the latter does not achieve significantly lower error rates than the former in any of these domains. A one-tailed sign-test shows that MB is significantly frequently more accurate than BAG in the 40 domains at a level of 0.0001.

The average error rate of MB is 0.55 percentage points lower than that of BOOST in the 40 domains, but their average error ratio is 1.00. It might be thought that MB has no advantage since the average error ratio to BOOST is 1. However, it is noticed that the average *accuracy ratio* of MB against BOOST (the accuracy for MB divided by the corresponding accuracy for BOOST) is 1.01. That is, MB is 1% more accurate than BOOST on average. This discrepancy arises because error ratio favors better performance at low error rates while accuracy ratio rather favors better performance at high error rates. Nevertheless, the one-tailed sign-test shows that the frequency of accuracy increases is not significant over the 40 domains at a level of 0.05.

(4) MB is more stable than BOOST.

MB achieves significantly lower error rates than C4.5 in 29 domains, and significantly higher error rates than C4.5 in only one domain. The biggest error increase of MB over C4.5 is 15%. Given that BOOST makes significant error reductions in 27 domains and significant error increases in 5 domains over C4.5, as well as that the biggest error increase of BOOST over C4.5 is 61%, MB 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.

4.3 Voting Methods

As mentioned in Sections 2.3 and 3, BOOST, BAG, and MB can use four voting methods, namely probabilistic predictions (without

Table 4: Average error rates (%) and ratios over C4.5 of BOOST, BAG, and MB 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
MB	15.42	.78	15.51	.78	16.16	.82	16.20	.83

voting weights), categorical predictions (without voting weights), probabilistic predictions with voting weights, and categorical predictions with voting weights, to make the final classification.[6] Table 4 presents the average error rates and the average error rate ratios over C4.5 of BOOST, BAG, and MB 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 performs better than 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 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 ($p = 0.0401$).

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% higher than (absolute error rate difference) 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 MB, the probabilistic predictions without voting 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 any

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

of the other three methods at a level of 0.01 or better.

In short, these experimental results suggest that using a function of the performance of a tree on the training set as the voting weight may provide very marginal advantage for classifier committee learning algorithms including BOOST, BAG, and MB. 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.

5 Conclusions

In this paper, we present a multiple committee learning algorithm, namely MB, which incorporates Bagging into Boosting. Since the generation of subcommittees is independent from each other, MB is amenable to parallel or distributed processing.

The empirical studies using a representative collection of natural domains show that the probabilistic predictions without voting weights is an appropriate voting method for decision tree committee learning algorithms including Boosting, Bagging, and MB. Its average performance is better than or equal to that of the other voting methods: probabilistic predictions with voting weights, or categorical predictions with or without voting weights.

Experimental results show that MB is more stable than BOOST. It is significantly more accurate than BAG on average. Its average error rate is also lower than that of BOOST. All these results suggest that Multiple Boosting is a better choice than either Boosting or Bagging alone for parallel datamining.

Acknowledgments

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

References

[Ali and Pazzani, 1996] Ali, K.M. and Pazzani, M.J. Error Reduction through Learning Multiple Descriptions. *Machine Learning* 24: 173-202.

[Ali, 1996] Ali, K.M. Learning Probabilistic Relational Concept Descriptions. Ph.D. diss., Dept of Info. and Computer Science, Univ. of California, Irvine.

[Bauer and Kohavi, 1998] Bauer, E. and Kohavi, R. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. Submitted to *Machine Learning* (available at http://reality.sgi.com/ronnyk/vote.ps.gz).

[Breiman, 1996a] Breiman, L. Bagging Predictors. *Machine Learning* 24: 123-140.

[Breiman, 1996b] Breiman, L. Arcing Classifiers. Technical Report (available at http://www.stat.Berkeley.EDU/users/breiman/). Dept of Statistics, Univ of California, Berkeley, CA.

[Chan *et al.*, 1996] 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.

[Dietterich and Bakiri, 1995] Dietterich, T.G. and Bakiri, G. Solving Multiclass Learning Problems via Error-correcting Output Codes. *Journal of Artificial Intelligence Research* 2: 263-286.

[Dietterich and Kong, 1995] 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 (available at ftp://ftp.cs.orst.edu/pub/tgd/papers/tr-bias.ps.gz).

[Dietterich, 1997] Dietterich, T.G. Machine Learning Research. *AI Magazine* 18: 97-136.

[Domingos, 1997] Domingos, P. Why does Bagging Work? a Bayesian Account and its Implications. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, 155-158. AAAI Press.

[Freund, 1996] Freund, Y. Boosting a Weak Learning Algorithm by Majority. *Information and Computation* 121(2): 256-285.

[Freund and Schapire, 1996a] Freund, Y. and Schapire, R.E. A Decision-theoretic Generalization of On-line Learning and an Application to Boosting. Unpublished manuscript (available at http://www.research.att.com/~yoav).

[Freund and Schapire, 1996b] Freund, Y. and Schapire, R.E. Experiments with a New Boosting Algorithm. In Proceedings of the Thirteenth International Conference on Machine Learning, 148-156. San Francisco, CA: Morgan Kaufmann.

[Kohavi, 1995] Kohavi, R. A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, 1137-1143. Morgan Kaufmann.

[Kohavi and Kunz, 1997] Kohavi, R. and Kunz, C. Option Decision Trees with Majority Votes. In Proceedings of the Fourteenth International Conference on Machine Learning, 161-169. San Francisco, CA: Morgan Kaufmann.

[Kwok and Carter, 1990] Kwok, S.W. and Carter, C. Multiple Decision Trees. In Schachter, R.D., Levitt, T.S., Kanal, L.N., and Lemmer, J.F. eds, *Uncertainty in Artificial Intelligence*, 327-335. Elsevier Science.

[Merz and Murphy, 1997] 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.

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

[Quinlan, 1996] Quinlan, J.R. Bagging, Boosting, and C4.5. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, 725-730. AAAI Press.

[Schapire, 1990] Schapire, R.E. The Strength of Weak Learnability. *Machine Learning* 5: 197-227.

[Schapire *et al.*, 1997] Schapire, R.E., Freund, Y., Bartlett, P., and Lee, W.S. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. In Proceedings of the 14th International Conference on Machine Learning, 322-330. Morgan Kaufmann.

[Webb, 1998] Webb, G.I. Idealized Models of Decision Committee Performance and Their Application to Reduce Committee Error. Tech Report (TR C98/11), School of Computing and Mathematics, Deakin University, Australia.