A Comparison of Decision Tree Ensemble Creation Techniques

Robert E. Banfield, *Student Member*, *IEEE*,
Lawrence O. Hall, *Fellow*, *IEEE*,
Kevin W. Bowyer, *Fellow*, *IEEE*, and
W.P. Kegelmeyer, *Member*, *IEEE*

Abstract—We experimentally evaluate bagging and seven other randomization-based approaches to creating an ensemble of decision tree classifiers. Statistical tests were performed on experimental results from 57 publicly available data sets. When cross-validation comparisons were tested for statistical significance, the best method was statistically more accurate than bagging on only eight of the 57 data sets. Alternatively, examining the average ranks of the algorithms across the group of data sets, we find that boosting, random forests, and randomized trees are statistically significantly better than bagging. Because our results suggest that using an appropriate ensemble size is important, we introduce an algorithm that decides when a sufficient number of classifiers has been created for an ensemble. Our algorithm uses the out-of-bag error estimate, and is shown to result in an accurate ensemble for those methods that incorporate bagging into the construction of the ensemble.

Index Terms—Classifier ensembles, bagging, boosting, random forests, random subspaces, performance evaluation.

1 Introduction

BAGGING is one of the older, simpler, and better known techniques for creating an ensemble of classifiers [1]. A number of other randomization-based ensemble techniques have been introduced. Some of the more prominent of these include boosting [2], [3], [4], random subspaces [5], random forests [6], and randomized C4.5 [7]. We present the results of an experimental study aimed at determining the extent to which any of these other techniques offer an increase in accuracy over bagging.

This is the largest such experimental study to date, in terms of the number of experimental data sets and the breadth of different techniques considered. We compare boosting, random subspaces, three variations of random forests, and randomized C4.5 against standard bagging. We present experimental results on a total of 57 different data sets. This includes all the data sets used in previous studies on boosting [3], random subspaces [5], random forests [6], and randomized C4.5 [7], plus two additional data sets.

This is also the most rigorous such study to date, looking at statistical significance based on the typical 10-fold cross-validation evaluation method and contrasting this with significance based on the improved 5×2-fold cross-validation proposed by Dietterich [8] and modified by Alpaydin [9]. A paired t-test on the results of a 10-fold cross-validation is the typical approach used in the literature when statistical significance is reported. Dietterich notes that the 10-fold cross-validation violates the assumptions of the statistical test, in a way that results in an underestimate of the variance, leading to results being declared statistically significant more frequently than they should. Alpaydin notes that Dietterich's method can produce instability based on the order of the cross-validations, and corrects for this using an F-test.

In [10], an approach based on average algorithm rank was argued as the best way to evaluate multiple algorithms on multiple data sets. It allows for a summary decision to be made on statistically significant performance differences over the whole group of data sets and we have applied it for that purpose.

This work extends our previous results [11] in several important respects.

1. Results for boosting are now included in our evaluation.
2. The number of data sets used is greatly increased.
3. A much larger ensemble size is considered.
4. The statistical analysis includes the 5×2-fold approach, as well as the typical 10-fold cross-validation, and an average rank analysis. These extensions have altered some previous conclusions and made some additional insights possible.
5. A method to determine when to stop adding classifiers to an ensemble is introduced.

2 Randomization-Based Ensemble Creation Techniques

We report on an experimental evaluation that looks at bagging as a baseline against which to compare other randomization-based ensemble techniques. The other techniques considered here are boosting, random subspaces, randomized C4.5, and random forests. Bagging, boosting, and random subspaces are general techniques that can be used with any type of base classifier. However, the evaluation reported here focuses on using decision trees as the base classifier.

Bagging creates an ensemble of classifiers by sampling with replacement from the set of training data to create new training sets called "bags" [1]. In the results reported here, as is the case for most work on bagging, the number of items in each bag is the same as the number of items in the set of training data and a separate classifier is trained from each bag. We consider ensembles consisting of up to 1,000 classifiers.

Ho's random subspace technique selects random subsets of the available features to be used in training the individual classifiers in an ensemble [5]. Ho's approach randomly selects one half of the available features for each decision tree and creates ensembles of size 100. In one set of experiments, the random subspace technique gave better performance than either bagging or boosting for a single train/test split for four data sets. Another set of experiments involved 14 data sets that were randomly split into halves for training and testing. Ten random splits were done for each of the 14 data sets. For each data set, the minimum and maximum of the 10 accuracies were deleted and the remaining eight values averaged. Qualitatively, it appears that random subspaces resulted in higher accuracy than either bagging or boosting on about five of the 14 data sets. The differences in accuracy were not evaluated for statistical significance. Ho summarized the results as follows: "The subspace method is better in some cases, about the same or worse in other cases when compared to the other two forest building techniques [bagging and boosting]" [5]. One other conclusion was that "the subspace method is best when the data set has a large number of features and samples, and that it is not good when the data set has very few features coupled with a very small number of samples or a large number of classes" [5].

Breiman's random forest technique blends elements of random subspaces and bagging in a way that is specific to using decision trees as the base classifier [6]. At each node in the tree, a subset of the

- *R.E. Banfield and L.O. Hall are with the Department of Computer Science and Engineering, University of South Florida, ENB118, Tampa, FL 33620-9951. E-mail: {rbanfiel, hall}@csee.usf.edu.*
- *K.W. Bowyer is with the Department of Computer Science and Engineering, University of Notre Dame, South Bend, IN 46556. E-mail: kwb@cse.nd.edu.*
- *W.P. Kegelmeyer is with Sandia National Labs, Biosystems Research, PO Box 969, MS 9951, Livermore, CA 94551-0969. E-mail: wpk@california.sandia.gov.*

Manuscript received 10 Dec. 2005; revised 30 May 2006; accepted 5 June 2006; published online 13 Nov. 2006.
Recommended for acceptance by J. Buhmann.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0695-1205.

TABLE 1
Selected Aspects of This Work Compared with Previous Works

Reference	Number of Datasets	Ensemble Size	Experiments	Results
Ho [5], random subspaces	18	100	half-half split, repeat 10 times, average of middle 8	RS better than bagging, boosting in 5 or 6 of 14 data sets; no statistical test
Breiman [6], random forests	20	RF: 100, boosting: 50	90-10 split, 100 trials	RF better than Adaboost in 11 of 19 data sets, no statistical test
Dietterich [7], random trees	33	RT: 200 bagging: 200 boosting: 100	10-fold cross-val select pruned or unpruned trees	statistically significantly better in 6 of 33 data sets
Freund [3], boosting (Adaboost)	27	100	10-fold cross-val, 10 trials	boosting better than bagging in 12 of 27 data sets, no statistical test
this work	57	1,000	10-fold cross-val, 5x2-fold cross-val, Friedman-Holm test	no stat. sig. improvement over bagging in 38 of 57 data sets; boosting_1000, RF_lgN best; bagging, random subspaces equiv.

available features is randomly selected and the best split available within those features is selected for that node. Also, bagging is used to create the training set of data items for each individual tree. The number of features randomly chosen (from n total) at each node is a parameter of this approach. Following [6], we considered versions of random forests created with random subsets of size 1, 2, and $\lfloor log_2(n) + 1 \rfloor$. Breiman reported on experiments with 20 data sets, in which each data set was randomly split 100 times into 90 percent for training and 10 percent for testing. Ensembles of size 50 were created for Adaboost and ensembles of size 100 were created for random forests, except for the zip code data set, for which ensembles of size 200 were created. Accuracy results were averaged over the 100 train-test splits. The random forest with a single attribute randomly chosen at each node was better than AdaBoost on 11 of the 20 data sets. The random forest with $\lfloor log_2(n) + 1 \rfloor$ attributes was better than AdaBoost on 14 of the 20 data sets. The results were not evaluated for statistical significance.

Dietterich introduced an approach that he termed randomized C4.5 [7]. We will refer to this more generally as random trees. In this approach, at each node in the decision tree, the 20 best tests are determined and one of them is randomly selected for use at that node. With continuous attributes, it is possible that multiple tests from the same attribute will be in the top 20. Dietterich reported on experiments with 33 data sets from the UC Irvine repository. For all but three of the data sets, a 10-fold cross-validation approach was followed. The other three used a train/test split as included in the distribution of the data set. Random tree ensembles were created using both unpruned and pruned (with certainty factor 10) trees, and the better of the two was manually selected for comparison against bagging. Differences in accuracy were tested for statistical significance at the 95 percent level. With this approach, it was found that randomized C4.5 resulted in better accuracy than bagging six times, worse performance three times, and was not statistically significantly different 24 times.

Freund and Schapire introduced a boosting algorithm [3] for incremental refinement of an ensemble by emphasizing hard-to-classify data examples. This algorithm, referred to as AdaBoost.M1, creates classifiers using a training set with weights assigned to every example. Examples that are incorrectly classified by a classifier are given an increased weight for the next iteration. Freund and Schapire showed that boosting was often more accurate than bagging when using a nearest neighbor algorithm as the base classifier, though this margin was significantly diminished when using C4.5. Results were reported for 27 data sets, comparing the performance of boosting with that of bagging using C4.5 as the base classifier. The same ensemble size of 100 was used for boosting and bagging. In general, 10-fold cross-validation was done, repeated for 10 trials, and average error rate reported. For data sets with a defined test set, an average of 20 trials was used with this test set. Boosting resulted in higher accuracy than bagging on 13 of the 27 data sets, bagging resulted in higher accuracy than boosting on 10 data sets, and there were 4 ties. The differences in accuracy were not evaluated for statistical significance.

Table 1 shows a comparative summary of experiments and results of this work with the previously discussed work.

3 Experimental Design

In this work, we used the free open source software package "OpenDT" [13] for learning decision trees in parallel. This program has the ability to output trees very similar to C4.5 release 8 [14], but has added functionality for ensemble creation. In OpenDT, like C4.5, a penalty is assessed to the information gain of a continuous attribute with many potential splits. In the event that the attribute set randomly chosen provides a "negative" information gain, our approach is to randomly rechoose attributes until a positive information gain is obtained, or no further split is possible. This enables each test to improve the purity of the resultant leaves. This approach was also used in the WEKA system [15].

As AdaBoost.M1 was designed for binary classes, we use a simple extension to this algorithm called AdaBoost.M1W [2] which modifies the stopping criteria and weight update mechanism to deal with multiple classes and weak learning algorithms. Our boosting algorithm uses a weighted random sampling with replacement from the initial training set, which is different from a boosting-by-weighting approach where the information gain is adjusted according to the weight of the examples. Freund and Schapire used boosting-by-resampling in [3]. There appears to be no accuracy advantage for boosting-by-resampling or boosting-by-reweighting [16], [17], [18] though Breiman reports increased accuracy for boosting-by-resampling when using unpruned trees [19]. We use unpruned trees because of this and, in general, for increased ensemble diversity [20]. Boosting-by-resampling may take longer to converge than boosting-by-reweighting though.

We have made a modification to the randomized C4.5 ensemble creation method in which only the best test from each attribute is

TABLE 2
Description of Data Sets Attributes and Size

Data Set	num. attr.	num. cont.	num. examples	num. classes	Data Set	num. attr.	num. cont.	num examples	num classes
abalone[3]	8	7	4177	29	mushroom[4]	22	0	8124	2
anneal[2]	38	6	898	6	nursery[3]	8	0	12961	5
audiology[2,4]	69	0	226	24	page	10	10	5473	5
autos[2]	25	15	205	7	pendigits	16	16	10992	10
breast-w[1,2,3,4]	9	9	699	2	phoneme[2]	5	5	5404	2
breast-y[2]	9	0	286	2	pima[1,3,4]	8	8	768	2
bupa[1]	6	6	345	2	primary[2]	17	0	339	22
car[3]	6	0	1728	4	promoters[4]	57	0	106	2
credit-a[2]	15	6	690	2	ringnorm[1]	20	20	300	2
credit-g[1,2,3,4]	20	7	1000	2	sat[1,2,3,4]	36	36	6435	7
crx[4]	15	6	690	2	segment[1,2,3,4]	19	19	2310	7
dna[3]	180	0	3186	3	shuttle[2,3]	9	9	58000	7
ecoli[1]	7	7	325	8	sick[2,4]	29	7	3772	2
glass[1,2,4]	9	9	214	7	sonar[1,2,4]	60	60	208	2
heart-c[2,4]	13	5	303	2	soybean[1,2,4]	35	0	683	19
heart-h[2]	13	5	294	2	soybean-small[4]	35	0	47	4
heart-s[2]	13	5	123	2	splice[2,3,4]	60	0	3190	3
heart-v[2]	13	5	200	2	tic-tac-toe[3]	9	0	958	2
hepatitis[2,4]	19	6	155	2	twonorm[1]	20	20	300	2
horse-colic[2]	22	8	368	2	threenorm[1]	20	20	300	2
hypo[2,4]	25	7	3163	2	vehicle[1,2,3,4]	18	18	846	4
ion[1,4]	34	34	351	2	voting[1,4]	16	0	435	2
iris[2,4]	4	4	150	3	vote1[2,4]	15	0	435	2
krk[2]	6	6	28056	18	vowel[1,4]	10	10	528	11
krkp[2,3,4]	36	0	3196	2	waveform[1,2]	21	21	5000	3
labor[2,4]	16	8	57	2	yeast[3]	8	8	1484	10
led-24	24	0	5000	10	zip[1]	256	256	9298	10
letter[1,2,3,4]	16	16	20000	26					
lrs[3]	93	93	530	10					
lymph[2]	18	3	148	4					

[1] Used in Breiman's Random Forest paper [6].
[2] Used in Dietterich's Random Trees paper [7]
[3] Used in Ho's Random Subspaces paper [5].
[4] Used in Freund & Schapire's Boosting paper [3]

allowed to be among the best set of 20 tests, from which one is randomly chosen. This allows the algorithm to be less prejudiced against discrete attributes when there are a large number of continuous valued attributes. We call it the "random trees B" approach. For this approach, we used a random test from the 20 attributes with maximal information gain.

In the random subspace approach, half ($\lceil n/2 \rceil$) of the attributes were chosen each time. For the random forest approach, we used a single attribute, two attributes, and $\lfloor log_2\, n + 1 \rfloor$ attributes (which will be abbreviated as random forests-lg in the following).

Fifty-seven data sets were used, 52 from the UC Irvine repository [21], credit-g from NIAAD (www.liacc.up.pt/ML), phoneme from the ELENA project (ftp.dice.ucl.ac.be/pub/neural-nets/ELENA/databases), and several synthetic data sets from Breiman for which source code may be found with the Delve package (http://www.cs.utoronto.ca/~delve/data/datasets.html). The data sets, described in Table 2, have from 4 to 256 attributes and the attributes are a mixture of continuous and nominal values.

3.1 Experiments

For each data set, a stratified 10-fold cross-validation was performed. A stratified n-fold cross-validation breaks the data set into n disjoint subsets each with a class distribution approximating that of the original data set. For each of the n folds, an ensemble is trained using $n - 1$ of the subsets, and evaluated on the held out subset. As this creates n nonoverlapping test sets, it allows for statistical comparisons between approaches to be made.

For each data set, we also performed a set of five stratified two-fold cross-validations. In this methodology, the data set is randomly broken into two halves. One half is used in training and the other in testing and vice versa. This validation is repeated five times, each with a new half/half partition. Dietterich's experiments used a t test to evaluate statistical significance [8]. In Alpaydin's method, the t test is abandoned in favor of an F test for reasons of stability [9]. Specifically, rather than using the difference of only one test set, the difference of each test set is considered in the F test used here.

For each approach we use 1,000 trees in our ensemble, though we examine boosting with both 50 and 1,000 trees. Breiman often used only 50 trees in his research [1], [6], and Schapire has used as many as 1,000 [22].

3.2 Statistical Tests

We used three approaches to testing the statistical significance of the observed differences in accuracy. One approach is a t test on the results of a 10-fold cross-validation. This is the most widely used approach for this type of experiment. While the 10 folds of the cross-validation have independent test sets, the training data is highly overlapped across folds, and use of the t test assumes independent trials. Dietterich points out that this results in an elevated level of Type I error, which can be corrected for by his 5×2 cross-validation. This relies on the idea that learning curves rarely cross for algorithms as training set size varies.

We applied the Bonferroni correction, a calculation which raised the critical value necessary for determining significance, in order to compensate for the number of methods used in our experiments. In the Bonferroni correction, the α value of an entire set of n comparisons is adjusted by taking the α value of each individual test as α/n [23]. In our experiments, we define $\alpha = 0.05$ and $n = 7$. In the case of the 10-fold cross-validation, the t-critical value is 3.47 and for the 5×2-fold cross-validation, the F-critical value is 11.66.

A recent paper [10] suggests that the best way to compare multiple algorithms across multiple data sets is to compare their average ranks. In our case, one could rank the algorithm by average accuracy over a cross-validation experiment from 1-the best to 8-the

TABLE 3
Statistical Results for Each Data Set

Data Set	Boosting 1000	Boosting 50	Random Subspaces	Random Trees B	Random Forests-lg	Random Forests-1	Random Forests-2
zip	+/+	+/+	+/+	+/+	+/+	+/	+/+
letter	+/+	+/+	+/+	+/+	+/+		+/+
pendigits	+/+	+/+	+/+	+/	+/+		+/
heart-c			/+	+/	+/		+/
waveform				+/	+/		+/
page				+/			+/
sat	+/			+/			
sonar		+/		+/			
krk	+/+	+/	-/-		+/	-/-	+/+
splice	/+	+/	+/+		+/+	-/-	-/-
autos			+/				
credit-a							/+
lrs					/+		
promoters	+/						
vote1				+/			
breast-w				/-			
led-24		-/	-/-		+/+		+/+
tic-tac-toe	+/+	/+	-/-	-/		-/	
dna	+/		+/	+/		-/-	-/-
breast-y		-/					
horse-colic				-/			
car	+/+	+/+	-/-	-/-		-/-	
nursery	+/+	+/+	-/-	-/-		-/-	
yeast			-/-				
shuttle			-/-				
phoneme			-/-	-/-			
krkp			-/-	-/		-/	
sick			-/	-/		-/-	

Results are displayed as 10-Fold/5×2-Fold where a plus sign designates a statistically significant win and a minus designates a statistically significant loss. Only data sets for which there were significant differences are listed.

TABLE 4
Summary Statistical Table for Each Method Showing Statistical Wins and Losses, the Average Rank Is Also Shown

	Boosting 1000	Boosting 50	Random Subspaces	Random Trees B	Random Forests-lg	Random Forests-1	Random Forests-2
10 Fold Wins	10	8	6	10	8	1	8
10 Fold Losses	0	2	10	7	0	8	2
5x2 Fold Wins	8	6	5	2	6	0	5
5x2 Fold Losses	0	0	9	4	0	6	2
10 Fold average rank[1]	3.96	4.30	5.42	4.53	3.77	4.59	3.67
5x2 Fold average rank[2]	3.34	5.15	5.39	4.52	3.70	4.53	3.32

[1] Average rank for 10 Fold Bagging is 5.76
[2] Average rank for 5x2 Fold Bagging is 6.06

worst. If, for example, two algorithms tied for third, they would each get a rank of 3.5. After obtaining the average ranks the Friedman test can be applied to determine if there are any statistically significant differences among the algorithms for the data sets. If so, the Holm step-down procedure was used to determine which might be statistically significantly different from bagging. It was argued [10] that this is a stable approach for evaluating many algorithms across many data sets and determining overall statistically significant differences.

4 Experimental Results

Table 3 shows the results of our experiments. Statistical wins against bagging are designated by a plus sign and losses by a minus sign. If neither a statistical win nor statistical loss is registered, the table field for that data set is omitted. We separate the results of the 10-fold cross-validation and the 5×2-fold cross-validation with a slash. Table 4 shows a summary of our results.

For **37 of 57** data sets, considering both types of cross-validation, none of the ensemble approaches resulted in a statistically significant improvement over bagging. On one data set, zip, all ensemble techniques showed statistically significant improvement under the 10-fold cross-validation approach. The best ensemble building approaches appear to be boosting-1,000 and random forests-lg. Each scored the most wins against bagging while never losing. For both random subspaces and random forests-1 there were a greater number of statistical losses to bagging than statistical wins. Boosting with only 50 trees and random forests using only two attributes also did well. Random trees-B had a high number of statistical wins in the 10-fold cross-validation but also a high number of losses. Interestingly, in the 5×2-fold cross-validation, it resulted in very few wins and losses.

In comparing the differences between the 10-fold cross-validation and the 5×2-fold cross-validation, the primary difference is the number of statistical wins or losses. Using the 5×2-fold cross-validation method, for only 12 of the 57 data sets was there a statistically significant win over bagging with any ensemble technique. This can be compared to the 10-fold cross-validation where for 18 of the 57 data sets there was a statistically significant win over bagging. Under the 5×2-fold cross-validation, for no data set was every method better than bagging.

The average ranks for the algorithms are shown in Table 4. It was surprising to see that random forests when examining only two randomly chosen attributes had the lowest average rank. Using the

Friedman test followed by the Holm test with a 95 percent confidence level it can be concluded that there was a statistically significant difference between bagging and all approaches except for random subspaces using the average accuracy from a 10-fold cross-validation. Using the 5×2 cross-validation results, there was a statistically significant difference between bagging and all approaches except for boosting 50 classifiers and random subspaces. The approaches were often not significantly more accurate than bagging on individual data sets. However, they were consistently more accurate than bagging.

5 Discussion

Since many papers compare their approaches with bagging and show improvement, it might be expected that one or more of these approaches would be an unambiguous winner over bagging. This was not the case when the results are looked at in terms of statistically significant increases in accuracy on individual data sets. Of the 57 data sets considered, 37 showed no statistically significant improvement over bagging for any of the other techniques, using either the 10-fold or 5×2 cross-validation. However, using the Friedman-Holm tests on the average ranks, we can conclude that several approaches perform statistically significantly better than bagging on average across the group of data sets. Informally, we might say that while the gain over bagging is often small, there is a consistent pattern of gain.

There are three data sets, letter, pendigits, and zip, for which nearly everything improves on the accuracy of bagging. Each of those data sets involves character recognition. We conducted experiments that attempt to increase the diversity of an ensemble of bagged classifiers, hypothesizing that the diversity created by bagging on the letter and pendigits data sets was insufficient to increase the accuracy of the ensemble. This was performed by creating bags of a smaller size than the training set, these sizes ranging from 20 percent to 95 percent in 5 pecent increments. The highest ensemble accuracy obtained on the letter data set, with 95 percent bags, was only marginally higher than the result with 100 percent bags. This difference was not statistically significant. The pendigits data set showed no improvement at any size. Zip was not tested due to running time constraints.

The raw accuracy numbers show that random subspaces can be up to 44 percent less accurate than bagging on some data sets. Data sets that perform poorly with random subspaces likely have attributes which are both highly uncorrelated and each individually important. One such example is the krk (king-rook-king) data set which stores the position of three chess pieces in row#, column# format. If even one of the attributes is removed from the data set, vital information is lost. If half of the attributes are dismissed (e.g., King at A1, Rook at A?, and King at ??) the algorithm will not have enough information and will be forced to guess randomly at the result of the chess game.

Boosting-by-resampling 1,000 classifiers was substantially better than with 50 classifiers. Sequentially generating more boosted classifiers resulted in both more statistically significant wins and fewer statistically significant losses. If processing time permits additional classifiers to be generated, a larger ensemble than 50 is worthwhile.

Random forests using only two attributes obtained a better average rank than random forests-lg in both cross-validation methods but did worse in terms of number of statistically significant improvements. Experimentation with the splice data set resulted in statistically significant wins for random forests-lg and statistically significant losses for random forests-2 with a 6 to 9 percent difference in accuracy. Thus, while testing only two random attributes is likely sufficient, testing additional attributes may prove beneficial on certain data sets. Breiman suggested using out-of-bag accuracy to determine the number of attributes to test [6].

There are other potential benefits aside from increased accuracy. Random forests, by picking only a small number of attributes to test, generates trees very rapidly. Random subspaces, which tests fewer attributes, can use much less memory because only the chosen percentage of attributes needs to be stored. Recall that since random forests may potentially test any attribute, it does not require less memory to store the data set. Since random trees do not need to make and store new training sets, they save a small amount of time and memory over the other methods. Finally, random trees and random forests can only be directly used to create ensembles of decision trees. Bagging, boosting, and random subspaces could be used with other learning algorithms, such as neural networks.

6 An Advantage of Bagging-Based Methods for Ensemble Size

We used an arbitrarily large number of trees for the ensemble in the preceding section. The boosting results, for example, show that an increase in the number of trees provides better accuracy than the smaller ensemble sizes generally used. This suggests a need to know when enough trees have been generated. It also raises the question of whether approaches competitive with boosting-1,000 may (nearly) reach their final accuracy before 1,000 trees are generated. The easiest way of determining when enough trees have been generated would be to use a validation set. This unfortunately results in a loss of data which might otherwise have been used for training.

One advantage of the techniques which use bagging is the ability to test the accuracy of the ensemble without removing data from the training set, as is done with a validation set. Breiman hypothesized that this would be effective [6]. He referred to the error observed when testing each classifier on examples not in its bag as the "out-of-bag" error, and suggested that it might be possible to stop building classifiers once this error no longer decreases as more classifiers are added to the ensemble. The effectiveness of this technique has not yet been fully explored in the literature. In particular, there are several important aspects which are easily overlooked, and are described in the following section.

6.1 Considerations

In bagging, only a subset of examples typically appear in the bag which will be used in training the classifier. Out-of-bag error provides an estimate of the true error by testing on those examples which did not appear in the training set. Formally, given a set T of examples used in training the ensemble, let t be a set of size $|T|$ created by a random sampling of T with replacement, more generally known as a bag. Let s be a set consisting of $T - (T \cap t)$. Since s consists of all those examples not appearing within the bag, it is called the out-of-bag set. A classifier is trained on set t and tested on set s. In calculating the voted error of the ensemble, each example in the training set is classified and voted on by only those classifiers which did not include the example in the bag on which that classifier was trained. Because the out-of-bag examples, by definition, were not used in the training set, they can be used to provide an estimate of the true error.

Only a fraction of the trees in the ensemble are eligible to vote on any given item of training data by its being "out-of-bag" relative to them. For example, suppose out-of-bag error was minimized at 150 trees. These 150 trees are most likely an overestimate of the "true number" because for any example in the data set, it would need to be out-of-bag on 100 percent of the bags in order to have all 150 trees classify that example. Therefore, the OOB results most likely lead to a larger ensemble than is truly needed.

Our experimentation with algorithms to predict an adequate number of decision trees is further complicated by out-of-bag error

TABLE 5
Number of Trees and Test Set Accuracy of the Stopping Criteria for Random Forests and Bagging

Data Set	Algorithm # of Trees rf / bagging	Best OOB # of Trees rf / bagging	Oracle # of Trees rf / bagging	Algorithm Accuracy rf / bagging	Best OOB Accuracy rf / bagging	Oracle Accuracy rf / bagging
credit-g	113 / 92	296 / 645	25 / 913	75.7 / 74.6	75.8 / 75.2	76.8 / 75.8
hypo	53 / 73	100 / 305	48 / 502	99.11 / 99.21	99.15 / 99.24	99.15 / 99.30
krkp	216 / 35	1997 / 159	161 / 13	99.06 / 99.41	98.84 / 99.47	99.09 / 99.56
led-24	204 / 290	2000 / 946	132 / 887	75.02 / 74.52	75.00 / 74.78	75.36 / 74.90
pendigits	181 / 215	911 / 449	1518 / 838	99.02 / 98.43	99.04 / 98.46	99.09 / 98.53
phoneme	118 / 189	1464 / 377	123 /269	90.49 / 90.03	90.49 / 90.01	90.58 / 90.18
ringnorm	243 / 192	1841 / 988	947 / 961	96.35 / 95.60	96.43 / 95.69	96.49 / 95.72
sat	291 / 204	1828 / 969	786 / 926	91.67 / 91.14	91.72 / 91.19	91.80 / 91.22
segment	217 / 167	1862 / 555	1822 / 721	98.18 / 97.53	98.18 / 97.62	98.31 / 97.66
sick	93 / 64	93 / 64	1990 / 13	98.33 / 99.05	98.33 / 99.05	98.41 / 99.18
splice	342 / 129	1863 / 536	1414 / 609	96.30 / 95.05	96.58 / 95.14	96.65 / 95.17
twonorm	311 / 419	1586 / 985	1863 / 429	96.96 / 96.86	97.07 / 96.78	97.14 / 96.99
waveform	182 / 185	1534 / 876	1562 / 920	84.98 / 83.86	85.28 / 84.22	85.36 / 84.36

estimate quirks on data sets with a small number of examples. Small data sets (number of examples < 1,000) can often have a very low error estimate with a rather small number of decision trees (50 to 100), but then the addition of more trees results in a greater error rate in both the out-of-bag error and the test set error, as might be shown in a 10-fold cross-validation. This behavior is contrary to many experiments which have shown that test set error steadily decreases with an increasing number of classifiers until it plateaus. We speculate that this is a result of instability in the predictions leading to a "lucky guess" by the ensemble for such data sets. Since the decision to stop building additional classifiers is more effective, in a time-saving sense, for large data sets, we believe it is more important to concentrate on data sets with a larger number of examples.

We have developed an algorithm which appears to provide a reasonable solution to the problem of deciding when enough classifiers have been created for an ensemble. It works by first smoothing the out-of-bag error graph with a sliding window in order to reduce the variance. We have chosen a window size of 5 for our experiments. After the smoothing has been completed, the algorithm takes windows of size 20 on the smoothed data points and determines the maximum accuracy within that window. It continues to process windows of size 20 until the maximum accuracy within that window no longer increases. At this point, the stopping criterion has been reached and the algorithm returns the ensemble with the maximum raw accuracy from within that window. The algorithm is shown in Algorithm 1.

Algorithm 1 Algorithm for deciding when to stop building classifiers

```
1: $SlideSize \Leftarrow 5$, $SlideWindowSize \Leftarrow 5$, $BuildSize \Leftarrow 20$
2: $A[n] \Leftarrow$ Raw Ensemble accuracy with $n$ trees
3: $S[n] \Leftarrow$ Average Ensemble accuracy with $n$ trees over the
   previous SlideWindowSize trees
4: $W[n] \Leftarrow$ Maximum smoothed value
5: repeat
6:    Add ($BuildSize$) more trees to the ensemble
7:    $NumTrees = NumTrees + BuildSize$
      //Update $A[]$ with raw accuracy estimates obtained from
      out-of-bag error
8:    for $x \Leftarrow NumTrees - BuildSize$ to $NumTrees$ do
9:       $A[x] \Leftarrow$ VotedAccuracy($Tree_1 \ldots Tree_x$)
10:   end for
      //Update $S[]$ with averaged accuracy estimates
11:   for $x \Leftarrow NumTrees - BuildSize$ to $NumTrees$ do
12:      $S[x] \Leftarrow$ Average($A[x - SlideSize] \ldots A[x]$)
13:   end for
      //Update maximum smoothed accuracy within window
14:   W[$NumTrees/BuildSize - 1$]
      $\Leftarrow \max(S[NumTrees - BuildSize] \ldots S[NumTrees])$
15: until ($W[NumTrees/BuildSize - 1] \leq$
    $W[NumTrees/BuildSize - 2]$)
16: Stop at tree $argmax_j(A[j] | j \in [NumTrees - 2 *$
    $BuildSize] \ldots [NumTrees - BuildSize])$
```

6.2 Experiments

We compare the stopping points and the resulting test set accuracy of ensembles built out to 2,000 trees using Random Forests-lg and a 10-fold cross-validation. For this comparison we examine 1) the stopping point of our algorithm, 2) the stopping point by taking the minimum out-of-bag error over all 2,000 trees, and 3) an oracle algorithm which looks at the lowest observed error on the test set over the 2,000 created trees (as trees are added sequentially). Thirteen of the previously used data sets with greater than 1,000 examples are used. The results are shown in Table 5.

For most data sets, the out-of-bag error continues to decrease long into the training stage. This often does not result in any improvement of test set performance. Across all 13 data sets the total gain by using the minimum out-of-bag error rather than our algorithm was only 0.06 percent on average. Comparing our algorithm to the oracle, the accuracy loss is less than 0.25 percent per data set. In comparing the number of trees used, our method uses many fewer trees than the other methods. On average, we use 1,140 fewer trees compared to the minimum out-of-bag error and 755 fewer trees compared to the oracle method. While these numbers are clearly influenced by the maximum number of trees chosen to build, it is also evident that looking at the maximum out-of-bag accuracy causes the algorithm to continue building a large number of trees.

We have also tested this method on the bagged trees without the use of random forests. We generated half (1,000) the number of the trees used in the previous experiment in order to shorten the previously observed large over estimation on the number of trees using the minimum out-of-bag error alone and to reduce the training time. The results for this experiment are shown in Table 5. The use of our algorithm results in an average net loss of 0.12 percent per data set compared to the minimum out-of-bag error, while using 431 fewer trees. Compared to the oracle method, there is a net loss of 0.25 percent per data set (consistent with the previous experiment) while using 442 fewer trees.

TABLE 6
Test Set Accuracy Results Using a Third of the Trees Chosen in Table 5

Data Set	Original Algorithm Accuracy rf / bagging	1/3 Algorithm Accuracy rf / bagging	Original Max OOB Accuracy rf / bagging	1/3 Max OOB Accuracy rf / bagging
credit-g	75.70 / 74.60	75.60 / 73.80	75.80 / 75.20	75.80 / 74.90
hypo	99.11 / 99.21	99.11 / 99.27	99.15 / 99.24	99.11 / 99.24
krkp	99.06 / 99.41	98.88 / 99.53	98.84 / 99.47	98.81 / 99.47
led-24	75.02 / 74.52	74.92 / 74.54	75.00 / 74.78	75.00 / 74.48
pendigits	99.02 / 98.43	99.02 / 98.38	99.04 / 98.46	99.02 / 98.40
phoneme	90.49 / 90.03	90.17 / 89.93	90.49 / 90.01	90.32 / 89.90
ringnorm	96.35 / 95.60	96.01 / 95.45	96.43 / 95.69	96.35 / 95.60
sat	91.67 / 91.14	91.52 / 90.89	91.72 / 91.19	91.76 / 90.97
segment	98.18 / 97.53	98.31 / 97.45	98.18 / 97.62	98.27 / 97.49
sick	98.33 / 99.05	98.28 / 99.07	98.33 / 99.05	98.28 / 99.07
splice	96.30 / 95.05	95.77 / 95.05	96.58 / 95.14	96.27 / 94.95
twonorm	96.96 / 96.86	96.78 / 96.58	97.07 / 96.78	96.95 / 96.83
waveform	84.98 / 83.86	84.36 / 83.58	85.28 / 84.22	85.02 / 83.96

Based on these results, we believe it is possible to choose an acceptable stopping point while the ensemble is being built. In experiments with our algorithm, it has not shown itself to be overly sensitive to the parameters of the sliding window size and the building window size. On average, the number of trees built in excess for the purpose of choosing the stopping point in our algorithm, will be half of the building window size.

When bagging a data set, the probability of any particular example being included in the bag is slightly less than two-thirds, meaning only about one-third of the examples are out-of-bag. Put another way, for each example in the training set, only about one-third of the trees in the ensemble vote on that example. Therefore, the number of trees we have chosen to stop at may be as many as three times the amount necessary for equivalent performance on a test set consisting of all unseen examples. For this reason, we include the accuracy results obtained by using a random one-third of the number of trees chosen to stop with in the previous experiments. These results are shown in Table 6. Figs. 1 and 2 demonstrate the relationship of out-of-bag error and test set error for a given number of trees in the full ensemble. Fig. 2 is a worst-case result, with oob error decreasing but overall error being minimal early and higher with more trees before stabilizing.

Looking at the accuracy with one-third of the number of trees shows mixed results. Though there are some data sets unaffected by the change, other data sets, especially the larger sized ones, benefit from the greater number of trees. We believe that our algorithm, which stops at the first window at which accuracy no longer increases, compensates for what might otherwise require three times the number of trees to decide.

7 Conclusions

This paper compares a variant of the randomized C4.5 method introduced by Dietterich [7], random subspaces [5], random forests [6], AdaBoost.M1W [2], and bagging. A 10-fold cross validation and 5×2-fold cross validation are used in the comparison. The accuracy of the various ensemble building approaches was compared with bagging using OpenDT to build unpruned trees. The comparison was done on 57 data sets. This is the largest comparison of ensemble techniques that we know of, in terms of number of data sets or number of techniques. This is also the most rigorous comparison, in the sense of employing the cross-validation test suggested by Alpaydin in addition to the standard 10-fold cross-validation and the Friedman-Holm test on the average rank.

We found that some of the well-known ensemble techniques rarely provide a statistically significant advantage over the accuracy achievable with standard bagging on individual data sets. We found that boosting-by-resampling results in better accuracy with a much larger ensemble size than has generally

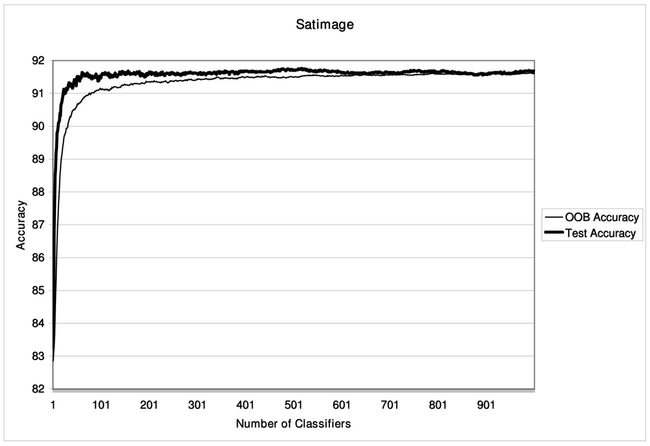

Fig. 1. Out-of-bag accuracy versus test set accuracy results as classifiers are added to the ensemble for satimage.

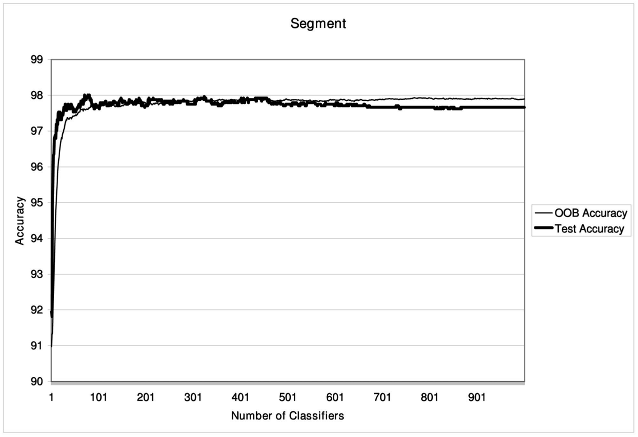

Fig. 2. Out-of-bag accuracy versus test set accuracy results as classifiers are added to the ensemble for segment.

been used, and that at this larger ensemble size it does offer some performance advantage over bagging. However, the increase in accuracy is statistically significant in only a fraction of the data sets used. Random forests-lg and random forests-2 show some improvement in performance over bagging. The accuracy improvement with these random forests algorithms is perhaps not quite as big as with boosting-1,000, however they have the advantage that the trees can be created in parallel.

An evaluation approach using the average ranking (by cross-validation accuracy) of the algorithms on each data set [10] has recently been argued to be the best approach for comparing many algorithms across many data sets. When we calculated the average ranks and then used the Friedman test followed by the Holm test, boosting 1,000, randomized trees, and random forests were statistically significantly better than bagging using the 5×2-fold cross-validation accuracies. With the 10-fold cross-validation accuracies, boosting-50 was also statistically significantly better than bagging. We conclude that for any given data set the statistically significantly better algorithms are likely to be more accurate, just not by a significant amount on that data set. So, performance/accuracy trade-offs may make sense in some cases.

We also showed a way to automatically determine the size of the ensemble. The stopping criteria we presented showed that it is possible to intelligently stop adding classifiers to an ensemble using out-of-bag error, as hypothesized by Breiman. Our experiments show this clearly applies to bagging and random forests-lg, which makes use of bagging. In particular, our results demonstrate that it is possible to stop much earlier than the minimum out-of-bag error would dictate, and still achieve good accuracy from the ensemble.

The raw accuracy results for the 10-fold and the 5×2-fold cross-validations are contained in an appendix. The Appendix can be found at http://computer.org/tpami/archives.htm.

ACKNOWLEDGMENTS

This research was partially supported by the US Department of Energy through the ASCI (DVS) Data Discovery Program, Contract number: DE-AC04-76DO00789 and the US National Science Foundation under grant EIA-0130768. The authors would like to thank Remy Losaria for her help with the 5×2-fold cross-validation experiments.

REFERENCES

[1] L. Breiman, "Bagging Predictors," *Machine Learning,* vol. 24, pp. 123-140, 1996.
[2] G. Eibl and K. Pfeiffer, "How to Make AdaBoost.M1 Work for Weak Base Classifiers by Changing Only One Line of the Code," *Proc. 13th European Conf. Machine Learning,* pp. 72-83, 2002.
[3] Y. Freund and R. Schapire, "Experiments with a New Boosting Algorithm," *Proc. 13th Nat'l Conf. Machine Learning,* pp. 148-156, 1996.
[4] R. Schapire, "The Strength of Weak Learnability," *Machine Learning,* vol. 5, no. 2, pp. 197-227, 1990.
[5] T. Ho, "The Random Subspace Method for Constructing Decision Forests," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 8, pp. 832-844, Aug. 1998.
[6] L. Breiman, "Random Forests," *Machine Learning,* vol. 45, no. 1, pp. 5-32, 2001.
[7] T. Dietterich, "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization," *Machine Learning,* vol. 40, no. 2, pp. 139-157, 2000.
[8] T.G. Dietterich, "Approximate Statistical Test for Comparing Supervised Classification Learning Algorithms," *Neural Computation,* vol. 10, no. 7, pp. 1895-1923, 1998.
[9] E. Alpaydin, "Combined 5×2 cv F Test for Comparing Supervised Classification Learning Algorithms," *Neural Computation,* vol. 11, no. 8, pp. 1885-1892, 1999.
[10] J. Demsar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *J. Machine Learning Research,* vol. 7, pp. 1-30, 2006.
[11] L. Hall, K. Bowyer, R. Banfield, D. Bhadoria, W. Kegelmeyer, and S. Eschrich, "Comparing Pure Parallel Ensemble Creation Techniques against Bagging," *Proc. Third IEEE Int'l Conf. Data Mining,* pp. 533-536, 2003.
[12] R.E. Banfield, L.O. Hall, K.W. Bowyer, and W.P. Kegelmeyer, "A Statistical Comparison of Decision Tree Ensemble Creation Techniques," *Proc. 2006 Int'l Conf. Systems, Man, and Cybernetics,* 2006, to appear.
[13] R. Banfield, OpenDT, http://opendt.sourceforge.net/, 2005.
[14] J. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1992.
[15] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools with Java Implementations.* Morgan Kaufmann, 1999.
[16] E. Bauer and R. Kohavi, "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants," *Machine Learning,* vol. 36, nos. 1-2, pp. 105-139, 1999.
[17] L. Breiman, "Arcing Classifiers," *Annals of Statistics,* vol. 26, no. 2, pp. 801-824, 1998.
[18] Y. Freund and R. Schapire, "Discussion of the Paper 'Arcing Classifiers' by Leo Breiman," *Annals of Statistics,* vol. 26, no. 2, pp. 824-832, 1998.
[19] L. Breiman, "Rejoinder to the Paper 'Arcing Classifiers' by Leo Breiman," *Annals of Statistics,* vol. 26, no. 2, pp. 841-849, 1998.
[20] R.E. Banfield, L.O. Hall, K.W. Bowyer, and W.P. Kegelmeyer, "A New Ensemble Diversity Measure Applied to Thinning Ensembles," *Proc. Fifth Int'l Workshop Multiple Classifier Systems,* pp. 306-316, 2003.
[21] C. Merz and P. Murphy, *UCI Repository of Machine Learning Databases,* Dept. of CIS, Univ. of California, Irvine, http://www.ics.uci.edu/~mlearn/MLRepository.html, 2006.
[22] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee, "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods," *Proc. 14th Int'l Conf. Machine Learning,* pp. 322-330, 1997.
[23] R. Johnson and D. Wichern, *Applied Multivariate Statistical Analysis,* third ed. Prentice-Hall, 1992.