

A Comparison of Ensemble Creation Techniques

Robert E. Banfield, Lawrence O. Hall, Kevin W. Bowyer¹, Divya Bhadoria,
W. Philip Kegelmeyer² and Steven Eschrich

Department of Computer Science & Engineering
University of South Florida
Tampa, Florida 33620-5399

¹ Computer Science & Engineering
384 Fitzpatrick Hall
Notre Dame, IN 46556

² Sandia National Labs, Biosystems Research Department, PO Box 969, MS
9951 Livermore, CA 94551-0969, USA

{rbanfiel,hall,dbhadori,eschrich}@csee.usf.edu,kwb@cse.nd.edu,wpk@ca.sandia.gov

Abstract

We experimentally evaluate bagging and six other randomization-based approaches to creating an ensemble of decision-tree classifiers. Bagging uses randomization to create multiple training sets. Other approaches, such as Randomized C4.5 apply randomization in selecting a test at a given node of a tree. Then there are approaches, such as random forests and random subspaces, that apply randomization in the selection of attributes to be used in building the tree. On the other hand boosting, as compared here, incrementally builds classifiers by focusing on examples misclassified by existing classifiers. Experiments were performed on 34 publicly available data sets. While each of the other six approaches has some strengths, we find that none of them is consistently more accurate than standard bagging when tested for statistical significance.

1 Introduction

Bagging [1], Adaboost.M1W [2, 3, 4], three variations of random forests [5], one variation of Randomized C4.5 [6] (which we will call by the more general name “random trees”), and random subspaces [7] are compared. With the exception of boosting, each ensemble creation approach compared here can be distributed in a simple way across a set of processors. This makes them suitable for learning from very large data sets [8, 9, 10] because each classifier in an ensemble can be built at the same time if processors are available. Their classification accuracy is evaluated through a series of 10-fold cross validation experiments on 34 data sets taken mostly from the UC Irvine repository [11]. In this work we use the open source software package “OpenDT” [12] for learning decision trees in parallel. This program has the ability to output trees very similar to C4.5 release 8 [13], but has added functionality for ensemble creation.

Previous experimental results [14] show that each of the ensemble creation techniques gives a statistically significant, though small, increase in accuracy over a single decision tree. However in head-to-head comparisons with bagging, none of the ensemble building methods was generally statistically significantly more accurate.

This work extends our previous work [14] by including the results for boosting, using ANOVA to better understand the statistical significance of the results, and increasing the number and size of the data sets used. We have also increased the number of classifiers in each ensemble, with the exception of boosting, to 1000. By using a greater number of decision trees in the ensemble and larger size data sets, our conclusions about several of the methods have changed.

2 Ensemble Creation Techniques Evaluated

Ho's random subspace method of creating a decision forest utilizes the random selection of attributes or features in creating each decision tree. Ho used a randomly chosen 50% of the attributes to create each decision tree in an ensemble and the ensemble size was 100 trees.

Ho found the random subspace approach was better than bagging and boosting for a single train/test data split for four data sets taken from the stat log project [15]. Fourteen other data sets were used by splitting them into two halves randomly. Each half was used as a training set with the other half used as a test set. This was done 10 times for each of the data sets. The maximum and minimum accuracy results were deleted and the other eight runs were averaged. There was no evaluation of statistical significance. The conclusion was that random subspaces was better for data sets with a large number of attributes. Ho's method tended to not be as good with a smaller number of attributes and a small number of examples, or a small number of attributes and a large number of classes. This approach is interesting for large data sets with a significant number of attributes because it requires less time and memory to build each of the classifiers.

Breiman's random forest approach to creating an ensemble also utilizes a random choice of attributes in the construction of each CART decision tree [16, 5]. However, a random selection of attributes occurs at each node in the decision tree. Potential tests from these random attributes are evaluated and the best one is chosen. So, it is possible for each of the attributes to be utilized in the tree. The number of random attributes chosen for evaluation at each node is a variable in this approach. Additionally, bagging is used to create the training set for each of the trees. We utilized random subsets of size 1, 2 and $\lfloor \log_2(n) + 1 \rfloor$ where n , is the number of attributes.

Random forest experiments were conducted on 20 data sets and compared with Adaboost on the same data sets in [5]. Ensembles of 100 decision trees were built for the random forests and 50 decision trees for Adaboost. For the zip-code data set 200 trees were used. A random 10% of the data was left out of the training set to serve as test data. This was done 100 times and the results averaged. The random forest with a single attribute randomly chosen at each node was better than Adaboost on 11 of the 20 data sets. There was no evaluation of statistical significance. It was significantly faster to build the ensembles using random forests.

Dietterich introduced an approach which he called Randomized C4.5 [6],

which comes under our more general description of random trees. In this approach, at each node in the decision tree the 20 best tests are determined and the actual test used is randomly chosen from among them. With continuous attributes, it is possible that multiple tests from the same attribute will be in the top 20. All tests (in C4.5) must be kept to determine the best 20, which can make this approach memory intensive when there are many continuous attributes which have many values that can be used in a binary test.

Dietterich experimented with 33 data sets from the UC Irvine repository. For all but three of them a 10-fold cross validation approach was followed. The best result from a pruned or unpruned ensemble was reported. Pruning was done with a certainty factor of 10. The test results were evaluated for statistical significance at the 95% confidence level. It was found that Randomized C4.5 was better than C4.5 14 times and equivalent 19 times. It was better than bagging with C4.5 6 times, worse 3 times and equivalent 24 times. From this, it was concluded that the approach tends to produce an equivalent or better ensemble than bagging. It has the advantage that you do not have to create multiple instances of a training set.

3 Algorithm Modifications

We describe our implementation of random forests and a modification to Dietterich’s randomized C4.5 method. 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 re-choose 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 [17].

We have made a modification to the Randomized C4.5¹ ensemble creation method in which only the best test from each attribute is allowed to be among the best set of twenty features 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.

4 Experimental Results

Thirty-four data sets, 32 from the UC Irvine repository [11], credit-g from NIAAD (www.liacc.up.pt/ML) and phoneme from the ELENA project. The data sets, described in Table 1, have from 4 to 69 attributes and the attributes are a

¹On a code implementation note, we allow trees to be grown to single example leaves, which we call pure trees. MINOBS is set to one (which means a test will be attempted any time there are two or more examples at a node), tree collapsing is not allowed and dynamic changes in the minimum number of examples in a branch for a test to be used are not allowed.

Table 1: Description of data sets attributes and size.

Data Set	# attributes	# continuous attributes	# examples	# classes
anneal	38	6	898	6
audiology	69	0	226	24
autos	25	15	205	7
breast-w	9	9	699	2
breast-y	9	0	286	2
credit-a	15	6	690	2
credit-g	20	7	1000	2
glass	9	9	214	7
heart-c	13	5	303	2
heart-h	13	5	294	2
heart-s	13	5	123	2
heart-v	13	5	200	2
hepatitis	19	6	155	2
horse-colic	22	8	368	2
hypo	25	7	3163	2
ion	34	34	351	2
iris	4	4	150	3
krkp	36	0	3196	2
labor	16	8	57	2
led-24	24	0	5000	10
letter	16	16	20000	26
lymph	18	3	148	4
page	10	10	5473	5
pendigits	16	16	10992	10
phoneme	5	5	5404	2
pima	8	8	768	2
primary	17	0	339	22
satimage	36	36	6435	7
sick	29	7	3772	2
sonar	60	60	208	2
soybean	35	0	683	19
vehicle	18	18	846	4
voting	15	0	435	2
waveform	21	21	5000	3

mixture of continuous and nominal values. Ensemble size was 50 for the boosting approach, and 1000 trees for each of the other approaches. While 1000 trees are more than the original authors suggested, this number was chosen so that more than enough trees were present in the ensemble. Unlike boosting, Breiman has shown these other techniques do not overfit as more classifiers are added to the ensemble [5].

For the random trees B approach, we used a random test from the 20 attributes with maximal information gain. In the random subspace approach of Ho, half ($\lceil n/2 \rceil$) of the attributes were chosen each time. For the random forest approach, we used a single attribute, 2 attributes and $\lceil \log_2 n + 1 \rceil$ attributes (which will be abbreviated as Random Forests-lg in the following).

For each data set, a 10-fold cross validation was done. For each fold, an ensemble is built by each method and tested on the held out data. This allows for statistical comparisons between approaches to be made. Each ensemble consists solely of unpruned trees. ANOVA was first used to determine which data sets showed statistically significant differences at a specified confidence level. Subsequently, a paired t-test was used to determine, at the same confidence level, whether a particular ensemble approach is better or worse than bagging.

Table 2 shows the comparative results at the 99% confidence interval. All ensemble creation techniques to the right of the bagging column in the table utilized bagging in creating training sets. For 30 of the data sets none of the ensemble approaches could produce a statistically significant improvement over bagging. On two data sets, all techniques showed accuracy improvement (as indicated by a boldface number). The (slightly) best ensemble building approaches appears to be random forests-lg which is better than bagging four times and random forests-2 which is also better four times. Random forests-lg is only worse than bagging once (indicated by a number in italics), while random forests-2 is worse twice. Boosting had the least wins at two, while losing to bagging once. Random subspaces lost to bagging four times and registered only three wins. Random trees B and random forests-1 were better 3 times and worse 1 time and 2 times respectively.

We can create a summary score for each ensemble algorithm by providing 1 point for a win, and 1/2 point for a tie. At the 99% confidence level the top performing ensemble methods are random forests-lg (18.5 points) and random forests-2/Random Trees B (18 points). All other approaches score 17.5 points or 16.5 (random subspaces).

An interesting question is how would these approaches rank if the average accuracy, regardless of significance, was the only criterion. Once again that random forests-lg and random forests-2 appear the best (24.5 and 23 points respectively). Random forests-1 on the other hand performs the worst (18.5 points). Random subspaces (21.5 points) performs much better in this study, beating both boosting and random trees B (each with 19 points). Clearly, utilizing statistical significance tests changes the conclusions that one would make given these experimental results. It is worth noting that all scores are above 17 which means they are each better than growing a bagged ensemble on average.

Table 2: The average raw accuracy results. Boldface indicates statistically significantly better than bagging at the 99% confidence interval and italics means significantly worse. Results of a Borda count are provided with higher values signifying better performance. Summary scores are also reported.

Data set	Boosting	Random Subspaces	Random Trees B	Bagging	Random Forests-lg	Random Forests-1	Random Forests-2
anneal	99.33	99.78	99.78	99.22	99.33	99.67	99.78
audiology	79.57	81.74	78.26	80.43	81.74	76.09	78.26
autos	87.12	87.60	82.31	89.02	86.14	82.79	83.29
breast-w	96.43	96.29	96.43	96.00	96.57	96.86	96.71
breast-y	68.97	73.10	74.48	71.72	72.76	74.83	74.14
credit-a	85.36	86.96	86.67	86.09	86.81	87.10	86.96
credit-g	75.10	76.70	74.20	74.60	76.90	73.70	75.90
glass	77.27	77.73	79.55	74.55	75.91	78.18	79.09
heart-c	82.90	82.26	83.55	78.39	81.94	81.61	81.94
heart-h	96.33	96.67	96.33	97.00	96.33	94.67	96.67
heart-s	96.15	93.08	91.54	94.62	93.85	93.08	91.54
heart-v	82.50	82.00	86.00	82.50	84.50	86.00	86.00
hepatitis	90.63	90.63	91.25	91.88	94.38	90.00	91.88
horse-Colic	98.38	98.38	94.59	97.03	97.57	93.78	96.22
hypo	99.87	99.91	99.91	99.81	99.94	99.46	99.91
ion	95.28	93.89	93.61	93.61	93.61	93.89	93.89
iris	94.67	94.00	94.67	94.67	94.67	94.00	94.00
krkp	99.56	<i>95.75</i>	<i>98.72</i>	99.66	<i>99.47</i>	<i>97.94</i>	<i>99.13</i>
labor	81.48	100.0	100.0	100.0	100.0	100.0	100.0
led-24	<i>71.43</i>	<i>69.44</i>	72.41	73.57	74.93	74.27	74.77
letter	96.74	97.03	96.44	94.90	96.84	95.66	96.81
lymph	82.67	80.00	86.67	78.67	84.00	84.00	85.33
page	96.39	97.21	97.34	97.19	97.32	97.21	97.39
pendigits	99.21	99.30	99.25	98.59	99.25	99.02	99.14
phoneme	<i>91.46</i>	83.70	90.37	91.42	91.26	91.02	91.35
pima	74.29	74.55	76.49	76.75	76.75	75.45	75.97
primary	39.71	53.24	49.71	49.41	51.18	50.00	50.29
sat	91.89	92.19	92.24	91.06	92.08	91.26	91.72
sick	99.10	<i>97.54</i>	98.73	99.05	98.99	<i>97.91</i>	<i>98.60</i>
sonar	81.43	82.38	85.24	77.14	81.90	82.38	83.81
soybean	91.59	95.80	94.93	93.19	94.35	93.91	94.78
vehicle	76.94	75.76	74.59	74.35	74.59	73.53	74.47
vote	95.23	95.45	94.77	95.91	95.91	95.00	96.14
waveform	84.21	85.27	85.55	84.01	85.01	85.59	85.41
Borda count	128	150	152	118	167	117	166
Summary							
Better	2	3	3		4	3	4
Worse	1	4	1		1	2	2
Same	31	27	30		29	28	28
Score	17.5	16.5	18		18.5	17.5	18

To complete our investigation of the performance of these ensemble creation techniques we list the average accuracy results over ten folds and provide a Borda count. The Borda count [18] is calculated by assigning a rank to each of the proposed methods (first place, second place, etc.). The first place method obtains N points, second place takes $N - 1$ points, and so on, where N is the number of methods compared. The sum of those values across all data sets is the Borda count, as shown in Table 2, where greater values correspond to more accurate methods.

Again we see random forests-lg and random forests-2 taking the lead, with Borda counts of 167 and 166, respectively. Random trees B and random subspaces obtain the next best scores of 152 and 150. It is difficult to say how many points constitutes a “significant” win, however boosting (128), bagging (118), and random forests-1 (117) certainly have a non-trivial number of points less.

5 Discussion

5.1 Random Forests and Bagging

Since the random forest approach uses bagging to create the training sets for the trees of their ensembles, one might expect that the two algorithms share the same wins and losses while the other methods do not. This turns out not to be the case. On the krkp data set, random forests are statistically significantly less accurate than bagging, as are random subspaces and random trees. Likewise in the three cases where each version of random forests is statistically significantly more accurate than bagging, random subspaces and random trees are also more accurate; on two of those data sets boosting is more accurate. An interesting experiment would be to measure the accuracy of random forests without bagging the training set since this could lead to a decrease in the running time. Random forests are already much faster than bagging since less attributes need to be tested at every possible node in the tree.

5.2 Comparison Against Prior Results in the Literature

Our accuracy results compare with those published by Breiman in [5] for both boosting and random forests. Of the 11 data sets common to each work, our implementation of random forests-1 is more accurate eight times and our implementation of boosting is more accurate seven times than what is shown in [5]. These variations are small, possibly influenced by the use of two different splitting criteria functions (the gini index for CART and information gain ratio for OpenDT).

In Dietterich’s Randomized C4.5 experiments, he chose to report the best of the pruned (C4.5 certainty factor of 10) and unpruned ensembles arguing that the decision to prune might always be correctly determined by doing cross validation on the training set. Of the 11 data sets for which Dietterich chose to use unpruned trees and which appear in our paper, the OpenDT implemen-

tation was more accurate seven times. There are some data sets for which our unpruned ensemble is much more accurate than Dietterich’s reported pruned ensemble (which, by transitivity, is more accurate than his unpruned ensemble). OpenDT handles missing attributes as a preprocessing step, replacing them prior to learning. C4.5 replaces missing attributes at each node in the tree with likely values from the reduced data set at each split, and assigns a penalty to the gain ratio of the attribute. The former method shows much better accuracy for ensembles on some data sets, and is the reason for the large discrepancies.

5.3 Other Ensemble Methods vs. Bagging

At the outset of the study, it was expected that one or more of these approaches would be an unambiguous winner over bagging in terms of accuracy. This was not the case statistically. Of the 34 data sets examined, the maximum number of statistical significance wins was four (with one loss). While the Borda count shows that, given several data sets, techniques such as random forests can show an accuracy improvement over bagging, for any particular data set, this accuracy improvement is not reliable.

There are other potential benefits aside from increased accuracy performance though. Random forests, by picking only a small number of attributes to test, generates trees very rapidly. Random subspaces, which tests less attributes, can also use much less memory because only the chosen percentage of attributes needs to be stored. Recall that since random forests may potentially split on any attribute, it does not require any 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.

Within the confines of this experiment, boosting has the unique ability to specialize itself on the hard to learn examples. Unfortunately this makes the algorithm highly susceptible to noise. As several of the data sets used here are known to be noisy, boosting is at a disadvantage in these experiments. If led-24, a synthetic data set with artificial noise, is removed from the experiment, boosting would have zero losses at the 99% confidence interval.

Finally, random trees and random forests can only be directly used to create ensembles of decision trees. As with bagging, boosting and random subspaces could be utilized with other learning algorithms such as neural networks.

6 Summary

This paper compares several methods of building ensembles of decision trees. In particular a variant of the randomized C4.5 method introduced by Dietterich [6] (which we call random trees B), random subspaces [7], random forests [5], Adaboost.M1W, and bagging are compared. All experiments used a 10-fold cross validation approach to compare average accuracy. The accuracy of the various ensemble building approaches was compared with bagging using OpenDT to

build unpruned trees. The comparison was done on 34 data sets with 32 taken from the UC Irvine repository [11] and the others publicly available.

The ensemble size was 1000 trees for each of the ensemble creation techniques except boosting which used 50. The ensemble size of boosting was chosen to match what had been used in previous work [5]. The ensemble size of the remaining techniques was chosen to allow plenty of classifiers in the ensemble. Statistical significance tests were done to determine whether each of the ensemble methods was statistically significantly more than or less accurate than bagging.

None of the approaches was unambiguously always more accurate than bagging. Random forests generally have better performance, and are much faster to build. The accuracy of the random subspace approach fluctuated, showing mediocre results statistically, but fairly good results generally. It is notable that a random forest built utilizing only two randomly chosen attributes for each test in the decision tree was among the most accurate classification methods.

Acknowledgments: This research was partially supported by the Department of Energy through the ASCI Views Data Discovery Program, Contract number: DE-AC04-76DO00789 and the National Science Foundation under grant EIA-0130768.

References

- [1] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.
- [2] G. Eibl and K.P. Pfeiffer. How to make adaboost.m1 work for weak base classifiers by changing only one line of the code. In *Proceedings of the Thirteenth European Conference on Machine Learning*, pages 72–83, 2002.
- [3] M. Collins, R.E. Schapire, and Y. Singer. Logistic regression, AdaBoost and Bregman distances. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 158–169, 2000.
- [4] R.E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.
- [5] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- [6] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.
- [7] T.K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.
- [8] G. Hulten and P. Domingos. Learning from infinite data in finite time. In *Advances in Neural Information Processing Systems 14*, pages 673–680, Cambridge, MA, 2002. MIT Press.

- [9] K.W. Bowyer, N.V. Chawla, Jr. T.E. Moore, L.O. Hall, and W.P. Kegelmeyer. A parallel decision tree builder for mining very large visualization datasets. In *IEEE Systems, Man, and Cybernetics Conference*, pages 1888–1893, 2000.
- [10] N.V. Chawla, T.E. Moore, L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, and C. Springer. Distributed learning with bagging-like performance. *Pattern Recognition Letters*, 24:455–471, 2003.
- [11] C.J. Merz and P.M. Murphy. *UCI Repository of Machine Learning Databases*. Univ. of CA., Dept. of CIS, Irvine, CA. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- [12] Robert Banfield. The opendt project. Technical report, The OpenDT Project, <http://www.csee.usf.edu/~rbanfiel/OpenDT.html>, 2003.
- [13] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1992. San Mateo, CA.
- [14] L.O. Hall, K.W. Bowyer, R.E. Banfield, D. Bhadoria, W.P. Kegelmeyer, and Steven Eschrich. Comparing pure parallel ensemble creation techniques against bagging. In *The Third IEEE International Conference on Data Minin*, pages 533–536, 2003.
- [15] P. Brazdil and J.Gama. The statlog project- evaluation / characterization of classification algorithms. Technical report, The STAT-LOG Project- Evaluation / Characterization of Classification Algorithms, <http://www.ncc.up.pt/liacc/ML/statlog/>, 1998.
- [16] L. Breiman, J.H. Friedman, R.A. Olshen, and P.J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA., 1984.
- [17] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, 1999.
- [18] J. C. de Borda. *Memoire sur les elections au scrutin*. Historie de l’Academie Royale des Sciences, Paris, 1781.