HALF&HALF BAGGING AND HARD BOUNDARY POINTS

Leo Breiman
Statistics Department
University of California
Berkeley, CA 94720
leo@stat.berkeley.edu

Technical Report 534
Statistics Department

September 1998

Half&half bagging is a method for generating an ensemble of classifiers and combining them that does not resemble any method proposed to date. It is simple and intuitive in concept and its accuracy is very competitive with Adaboost. Certain instances that are used repeatedly turn out to be located in the boundaries between classes and we refer to these as hard boundary points. The effectiveness of half&half bagging leads to the conjecture that the accuracy of any combination method is based on its ability to locate the hard boundary points.

1. **Introduction**

Half&half bagging is a method for producing combinations of classifiers having low generalization error. The basic idea is straightforward and intuitive--suppose k classifiers have been constructed to date. Each classifier was constructed using some weighted subset of the original training set. To construct the next training set, randomly select an example **e**. Run **e** down that subgroup of the k classifiers that did not use **e** in their training sets. Total the unweighted votes of this subgroup and use the plurality vote to predict the classification of **e**. If **e** is misclassified, put it in group MC. If not, into group CC. Stop when the sizes of both MC and CC are equal to M, where $2M \leq N$, and N is the size of the original training set. Usually, CC is filled first but the sampling continues until MC reaches the same size.

The basic idea, then, of half&half bagging to is use purely random sampling to form a new training set that is half full of examples misclassified up to the present, and half full of correctly classified examples. The new twist is that the classified-misclassified judgment for **e** is based only on the classifiers built on previous training sets not containing **e.** The size of the training sets constructed this way is 2M, and M is set by the user. Generally, the larger M the smaller the generalization error. In the experiments reported on here, we universally take M=N/4. Each classifier constructed has an equal vote.

The Adaboost algorithm (Freund and Schapire, [1995]) produces combinations of classifiers with remarkably low generalization error (Bauer and Kohavi[1998], Dietterich [1998], Drucker and Cortes[1996], Freund and Schapire [1996], Quillan[1996], Breiman[1998]). The reasons for this are obscure because the algorithm was initially designed to drive the training error to zero as quickly as possible, and not necessarily to produce low generalization error. But the generalization error keeps decreasing after the training error is zero. Adaboost is a complex algorithm, so it is difficult to understand why it does so well. An explanation has been advanced (Schapire et.al.[1997]) but it appears incomplete (Breiman[1997])

In contrast, the concept of half&half is intuitively clear and simple--just keep working on training sets consisting of half misclassified and half correctly classified examples. Therefore, it is a bit surprising that half&half produces test set error rates very competitive with Adaboost, and on balance, even does a little bit better. This result lets us see more clearly what is essential to a good combining algorithm--it forms training sets where misclassified examples are weighted more heavily, but still keeps in a proportion of correctly classified examples.

In Section 2 the primary experimental results are given. Section 3 defines hard boundary points and gives some graph and tables to illustrate where they live and how they are distinguished. In Section 4 we note and give empirical data to show that half&half bagging can give accurate results when used on small fractions of the training set. Section 5 contains some remarks on the light that half&half sheds on why some methods of combining work so well.

2. Experimental Results on Accuracy

Our experiments use 10 smaller sized data sets from the UCI repository, 4 larger sets separated into training and test sets, and 4 synthetic data sets. Table 1 gives a brief summary.

Table 1 Data Set Summary

Data Set	Train Size	Test Size	Inputs	Classes
glass	214	--	9	6
breast cancer	699	--	9	2
diabetes	768	--	8	2
sonar	208	--	60	2
vowel	990	--	10	11
ionosphere	351	--	34	2
vehicle	846	--	18	4
soybean	685	--	35	19
German credit	1000 `	--	24	2
image	2310	--	19	7
letters	15000	5000	16	26
sat-images	4435	2000	36	6
dna	2000	1186	60	3
zip-code	7291	2007	256	10
waveform	300	3000	21	3
twonorm	300	3000	20	2
threenorm	300	3000	20	2
ringnorm	300	3000	20	2

On each of the 10 smaller sized data sets, the following procedure was used: a random 10% of the data was set aside. On the remaining data half&half was run combining 100 trees. The set aside 10% was then put down the combined predictor to get a test set error. This was repeated 100 times and the test set errors averaged. The same procedure was followed for the Adaboost runs all of which are based on combining 50 trees.

In the runs on the larger data sets, the half&half results for the first three sets were based on combining 100 trees--the zip-code procedure combined 200. For Adaboost, 50 trees were combined for the first three and 100 for zip-code. The synthetic data was described in Breiman[1996] and also used in Schapire et.al.[1997]. There were 50 runs. In each run, a new training set of size 300 and test set of size 3000 were generated. In half&half 100 trees were combined in each run--50 in Adaboost.

The reason that the half&half runs combined twice as many trees as did the Adaboost runs was that half&half was generally slower to converge than Adaboost (see Section 3). But since half&half constructs trees on training sets half the size of the ones Adaboost uses, the cpu times are similar.

The results of these runs are reported in Table 2.

`Table 2 Test Set Errors (%)

Data Set	Adaboost	Half&Half
glass	22.0	21.1
breast cancer	3.2	3.5
diabetes	26.6	24.0
sonar	15.6	14.4
vowel	4.1	3.5
ionosphere	6.4	6.6
vehicle	23.2	24.1
soybean	5.8	6.1
German credit	30.6	23.4
image	7.7	2.2
letters	3.4	3.6
sat-images	8.8	8.6
dna	4.2	3.8
zip-code	6.2	5.9
waveform	17.8	17.8
twonorm	4.9	4.7
threenorm	18.8	17.4
ringnorm	6.9	6.1

The differences are small except on the German credit and image data sets where half&half is significantly more accurate than Adaboost. We do not understand why Adaboost does not do well on these two data sets.

3. **Hard Boundary Points**

After each classifier in the series is constructed, it is possible to form an imitation of the test set combined classifier. For each instance, look only at the votes of the classifiers constructed to date using training sets not containing the instance. Classify each instance by plurality vote and call this the *out-of-bag classifier*. Then an error is made on instance if the out-of-bag classification does not agree with the true class assignment.

Unlike the situation in straight bagging (see Tibshiani [1996], Wolpert and Macready[1996]) the out-of-bag classifier gives a biased estimate of the generalization error. As each new tree is constructed, it relies on the current out-of-bag classifications. This creates a dependence between the classifiers constructed and the out-of-bag classifications. The out-of-bag error estimates for the synthetic data sets tends to be close to the test set error estimates. But for the four larger data sets listed in Table 1 it tends to be above the test set error estimates by up to 40% to 60%

Some of the instances characterized as errors by the out-of-bag classifier are used over and over again in constructing the new classifiers. We call these instances *hard boundary points*.. They generally are on or near the boundary between classes and are consistently misclassified by the out-of-bag classifier for two reasons. One is that they are off-side. That is, the most likely class given the **x** location is not the data assigned class. Unless the instance is an outlier, this happens close to the boundary. The other reason is that they are close to the boundary and given the shape of the boundary the classifiers consistently put them on the wrong side. Figures 1, 2, and 3 plot the hard boundary points for the two-dimensional versions of the twonorm, threenorm and ringnorm data.

The hard boundary points are marked by the fact that when they are used in a training set, they are in the misclassified half. To illustrate, Figure 4 has graphs constructed using the first six data sets listed in Table 1. What is graphed for each instance in the data is the proportion of times that when it was in a training set, it was in the misclassified half of the training set. In the six data sets the hard boundary points are the upper band of instances that are invariably misclassified. The bulk of the instances are in the lower band. Depending on the data set, they are out-of-bag misclassified 30& to 50% of the times they show up in a training set.

The way that the hard boundary points effects the classifiers varies from data set to data set. To shed some light on this Table 3 was constructed using 300 iterations each for the six data sets graphed in Figure 4. The first column is the number of hard boundary points. The second column is the number of hard boundary points as a percent of the total number of instances in the data. The third column is the average percent of the training sets that each hard boundary points turned up in. The fourth column is the average percent of the misclassified half of the training set that consists of hard boundary points.

Table 3 Properties of Hard Boundary Points (HBP)

data set	# HBP	% HBP	freq. HBP	%HBP/half
glass	33	15.4	63.2	46.0
breast	20	2.8	98.8	12.4
diabetes	120	15.6	63.0	46.1
sonar	22	10.5	68.9	36.1
vowel	39	3.9	91.4	19.0
ionosphere	19	5.4	92.7	20.8

If data sets have large overlapping boundaries with a high intrinsic error rate, then there will be a large proportion of hard boundary points. With a low intrinsic error rate, there will be fewer hard boundary points. With fewer hard boundary points, the 3rd column shows that they tend to show up in almost every misclassified half of the training sets. The 4th column shows the obvious--with a smaller fraction of hard boundary points, they have a smaller representation in the training sets.

To see if the hard boundary points really make a difference, we set up a filter that keeps them out of the training sets. For any instance, after its 5th appearance in a training set, if the proportion of out-of-bag misclassifications becomes larger than .95, the instance cannot be used in any future training sets. Table 4 compares the test sets errors with this filter to the results without it for the first six data sets.

Table 4 Test Set Error (%) With and Without Hard Boundary Points.

data set	error without HBP	error with HBP
glass	26.8	21.1
breast	4.5	3.5
diabetes	24.8	24.0
sonar	24.5	14.4
ionosphere	7.9	6.6
vowel	16.3	3.5

Filtering out the hard boundary points can have a significant effect on the test set error. Note that this is true even for the breast, ionosphere and vowel data that have small numbers of hard boundary points. It makes little difference for the diabetes data, which has a large number of hard boundary points, but blows up the error for the glass and sonar data which also have large numbers of hard boundary points. These results show that the hard boundary points are an essential ingredient in accurate classification.

4. **Using smaller training sets.**

Half&half bagging also works well when the size of the training sets (2M) is small compared to N. For instance, in databases with N in the order of 10,000, 2M=N/50 produces respectable accuracy. This experimental result leads to a method for forming predictions in large databases (Breiman[1998]).

To illustrate this, half&half bagging was run on all the data sets listed in Table 1 using training sets 1/10 the size of the original data, i.e. 2M=N/10. In a bit of overkill, 500 iterations were used. Otherwise, the structure of the experiments were the same as reported above. The resulting error estimates are given in Table 6 where the error estimates for 2M=N/2 are reproduced from Table 2 for comparison.

Table 6 Test Set Errors Using Smaller Training Sets (%)

Data Set	2M=N/10	2M=N/2
glass	29.4	21.1
breast cancer	3.8	3.5
diabetes	23.6	24.0
sonar	17.7	14.4
vowel	5.4	3.5
ionosphere	6.8	6.6
vehicle	24.4	24.1
soybean	6.7	6.1
German credit	21.4	23.4
image	1.8	2.2
letters	3.7	3.6
sat-images	8.8	8.6
dna	3.8	3.8
zip-code	6.3	5.9
waveform	16.0	17.8
twonorm	4.2	4.7
threenorm	17.4	17.4
ringnorm	7.2	6.1

Except for the small glass data set, there are only slight increases in error rates when using one-tenth of the data to grow the trees instead of one-half. For a few data sets. the error is reduced. To get some understanding of what was going on, I looked at the hard boundary points for the synthetic data sets.

The hard boundary points selected using N/10 are generally a superset of those selected using N/2. The trees constructed using N/10 are smaller and will find more boundary points that are difficult to classify correctly. For instance, using the threenorm data, the number of hard boundary points increases by about 50%. But a large majority of the N/2 hard boundary points are included in the N/10 hard boundary points.

Since using N/2 usually gives slightly better accuracy then the smaller training sets, an obvious question is why not use all of the data, i.e. why not take 2M=N. Experimentally, 2M=N does not give as low as error rates as 2M=N/2. I believe the reason is that using 2M=N does not give enough diversity in the choice of the instances classified correctly by the pseudo-error measure.

5 <u>What is important in constructing ensembles of classifiers?</u>

Half&half bagging differs from Adaboost and other arcing algorithms (Breiman[1998a]). At the beginning of each new iteration, the instances are divided into two sets. Half of the new training set consists of random draws from one set, the other half from random draws from the second set. The first set has an appreciable number of hard boundary points--the other set consists of instances that are currently classified correctly. The tree classifier classifies every instance in the training set correctly, since it is grown to one example per terminal node.

The hard boundary points therefore have a significant influence on the structure of the tree. In terms of the boundary between classes, it is pinned down by the heavy representation of the hard boundary points but rocks back a forth with the need to correctly classify the randomly chosen coprrectly classified points, and the points in the misclassified set that are not hard boundary points.

In classification, the essential thing is to get the boundary between classes right. We don't know, apriori, which instances in the original training set are near the boundary. . If we did, we could weight these instances upward to force the classifier combination to get the boundary right. The hard boundary points are on the boundary or off-side and half&half bagging identifies them, weights them up, and thus forces the combination to construct an accurate boundary .

The hard boundary points are analogous to support vectors {Vapnik [995]}. The success of half&half bagging shows that the key to accurate combinations is in the placement of the hard boundary points, and not necessarily in getting high margins. In an interesting paper by Ratsch et.al [1998], Adaboost is analyzed and those instances having the minimum margin values are called *support patterns*. A two dimensional data set is used to show that the support pattern points in Adaboost mostly coincide with the support vectors points produced when a support vector machine is applied to the data. We believe that the support patterns produced by Adaboost will often coincide with the half&half hard boundary points.

References

Bauer, E. and Kohavi, R.[1998]An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants, Machine Learning 1-33.

Breiman, L. [1997} Prediction Games and Arcing Algorithms, Technical Report 504, Statistics Department, UCB

Breiman. L. [1998a] Arcing Classifiers, Annals Of Statistics, 26, 801-849

Breiman, L.[1998b] Pasting votes together for classification in large data bases on-line, to appear--Machine Learning

Dietterich, T. [1998] An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. Machine Learning, 1-22

Drucker, H. and Cortes, C. [1996] Boosting decision trees, Neural Information Processing 8, Morgan-Kaufmann, 479-485

Freund, Y. and Schapire, R. [1995] A decision-theoretic generalization of on-line learning and an application to boosting. to appear--Journal of Computer and System Sciences

Freund, Y. and Schapire, R. [1996] Experiments with a new boosting algorithm, Machine Learning: Proceedings of the Thirteenth International Conference, 148-156.

Quinlan, J.R.[1996] Bagging, Boosting, and C4.5, Proceedings of AAAI'96 National Conference on Artificial Intelligence, 725-730

Ratsch, G., Onoda, T., Muller, K.-R., [1998] Soft margins for Adaboost

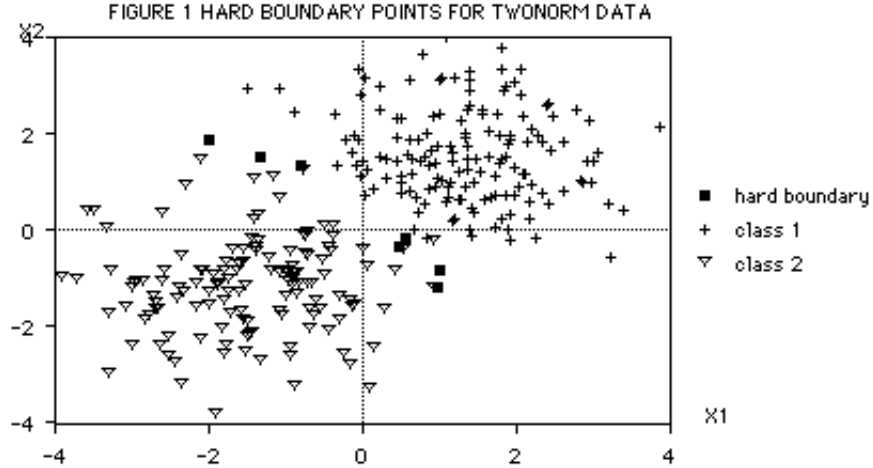
Schapire, R.,Freund, Y., Bartlett, P., and Lee, W[1997] Boosting the Margin, to appear--Annals of Statistics

Tibshirani, R.[1996] Bias, Variance, and Prediction Error for Classification Rules, Technical Report, Statistics Department, University of Toronto

Vapnik, V. [1995] The Nature of Statistical Learning Theory, Springer

Wolpert, D.H. and Macready, W.G.[1996] An Efficient Method to Estimate `Bagging's Generalization Error, to appear--Machine Learning

FIGURE 1 HARD BOUNDARY POINTS FOR TWONORM DATA



FIRURE 2 HARD BOUNDARY POINTS FOR THREENORM DATA

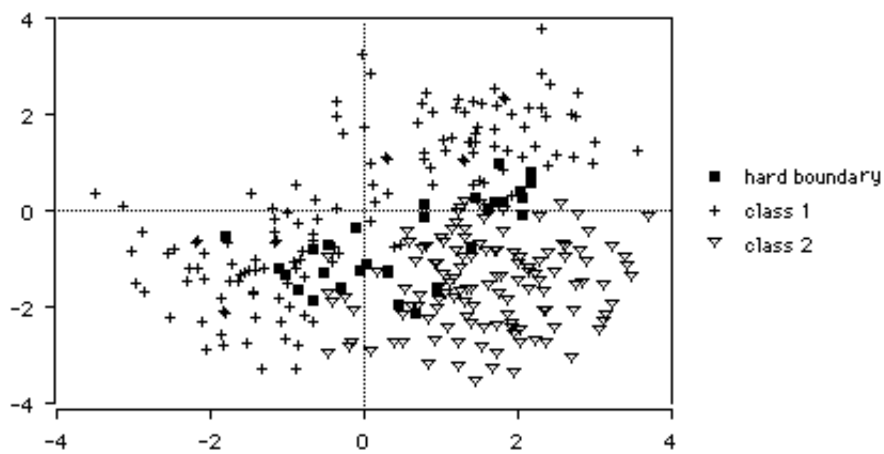


FIGURE 3 HARD BOUNDARY POINTS FOR RINGNORM DATA

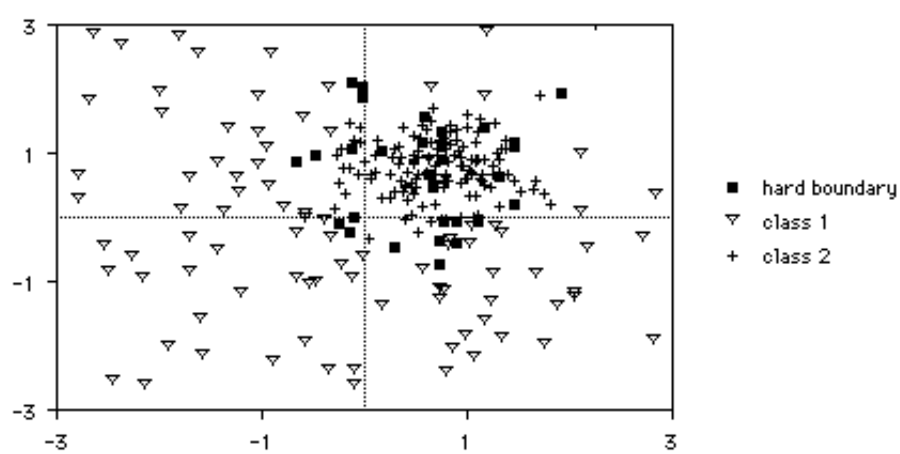


FIGURE 4 PROPORTION OF OUT-OF-BAG MISCLASSIFICATIONS