Machine Learning, 40, 229–242, 2000

Randomizing Outputs to Increase Prediction Accuracy

LEO BREIMAN leo@stat.berkeley.edu
Statistics Department, University of California, Berkeley, CA 94720, USA

Editor: William W. Cohen

Abstract. Bagging and boosting reduce error by changing both the inputs and outputs to form perturbed training sets, growing predictors on these perturbed training sets and combining them. An interesting question is whether it is possible to get comparable performance by perturbing the outputs alone. Two methods of randomizing outputs are experimented with. One is called output smearing and the other output flipping. Both are shown to consistently do better than bagging.

Keywords: ensemble, randomization, output variability

1. Introduction

In recent research in combining predictors, it has been recognized that the success in combining low-bias predictors such as trees and neural nets has been through methods that reduce the variability in the predictor due to training set variability. Assume that the training set consists of N independent draws from the same underlying distribution. Conceptually, training sets of size N can be drawn repeatedly and the same algorithm used to construct a predictor on each training set. These predictors will vary, and the extent of the variability is an important factor in the generalization prediction error.

Given training set $\{(y_n, \mathbf{x}_n), n = 1, \ldots, N\}$ where the y's are either class labels or numerical values, the most common way of reducing variability is by perturbing the training set to produce alternative training sets, growing a predictor on each perturbed training set, and either averaging the outcomes (regression) or letting them vote for the most popular class (classification). Examples are bagging (Breiman, 1996b) boosting (Freund & Schapire, 1995, 1996) and arcing (Breiman, 1998a).

However, in work (Breiman, 1996b) on subset selection in linear regression, I found the somewhat surprising result that adding noise to the y's while leaving the input $\mathbf{x}$-vectors unchanged worked just as well as bagging. This paper extends those earlier results to non-linear contexts in both regression and classification and also tries to understand why perturbing outputs only works well. Although we use trees in the experimental results, the ideas have more general validity.

In Section 2, we set up the general expressions for prediction error in regression and classification. In these expressions, we try to isolate a component that measures the contribution due to the y-variability only holding the inputs constant, and another component that measures the residual effect of input-variability. This can be done in a straightforward way in regression, but more implicitly in classification.

In Section 2 synthetic data is used in a manner that gives Monte Carlo estimates of the variability in the decision tree learner CART (Breiman et al., 1984) caused by the separate input and output components. The conclusions are that the output variability dominates the input variability in regression but not in classification. However, introducing extra artificial variability into the outputs reduces the error in classification considerably.

This suggests that if extra random variation is put into the outputs (given a single training set), leaving the inputs fixed, producing a sequence of perturbed training sets, then the predictors grown on these, averaged or voted, may be comparable in accuracy to methods that perturb both inputs and outputs.

Section 3 discusses possible methods for perturbing outputs. In regression, the situation is pretty clear. Adding Gaussian noise to the outputs works fairly well. Formulating a J-class classification problem in terms of J multiple outputs, where the jth output is one if the output label is j, and the other outputs are zero, puts it into a regression-like multiple output context. Now Gaussian noise can be added to each output independently of the others. We refer to the adding of Gaussian noise to the outputs as *Output Smearing*.

The most obvious way of perturbing the output in classification is to alter some of the class labels. In this procedure, we have found that it is important to keep the class compositions relatively invariant. Then the extent of the change is measured by a single real parameter called the flip rate and we call the procedure *Output Flipping*.

Section 4 gives the results of out experimental work with output smearing on a variety of data sets. Generally, output smearing works better than bagging but not as well as the Adaboost algorithm (Freund & Schapire, 1995, 1996). Section 5 gives the experimental results on flipping. It's error rates are comparable to output smearing. But unlike smearing where "one size fits all", it is sensitive to the size of the flip rate. Finally Section 6 contains some conclusions and remarks.

2. Output variability and prediction error

Denote a training set $T = \{(y_n, \mathbf{x}_n), n = 1, \ldots, N\}$. Assume the $(y_n, \mathbf{x}_n)$ are independently drawn from the same underlying probability distribution P and let $(Y, \mathbf{X})$ be a random vector having the distribution P and independent of the instances in the training set. Given a training set, we assume that we have an algorithm that will operate on the training set producing a function $f(\mathbf{x}, T)$ whose value is the predicted output for input vector $\mathbf{x}$.

2.1. Regression

Take the mean-squared prediction error to be defined as

$$PE(f(\cdot, T)) = E_{Y,\mathbf{X}}(Y - f(\mathbf{X}, T))^2 \tag{1}$$

where the subscripts $Y, \mathbf{X}$ denote expectation with respect to $Y, \mathbf{X}$ holding everything else fixed. To briefly review the work in Geman, Bienenstock, and Doursat (1992) and Breiman (1996a), we can always decompose Y as

$$Y = f^*(\mathbf{X}) + \varepsilon \tag{2}$$

where we refer to $f^*(\mathbf{X})$ as the structural part of Y and ε, the noise component, has the property that $E(\varepsilon \mid \mathbf{X}) \equiv 0$. Substituting (2) into (1) gives

$$PE(f(\cdot, T)) = E\varepsilon^2 + E_{\mathbf{X}}(f^*(\mathbf{X}) - f(\mathbf{X}, T))^2$$

We are interested in the average performance of the algorithm over replicated training sets of size N. Define

$$\bar{P}E(f) = E_T PE(f(\cdot, T))$$

and

$$\bar{f}(\mathbf{x}) = E_T f(\mathbf{x}, T)$$

Then some algebra results in:

$$E_{T,\mathbf{X}}(f^*(\mathbf{X}) - f(\mathbf{X}, T))^2 = E_{\mathbf{X}}(f^*(\mathbf{X}) - \bar{f}(\mathbf{X}))^2 + E_{T,\mathbf{X}}(f(\mathbf{X}, T) - \bar{f}(\mathbf{X}))^2 \quad (3)$$

The first term is a bias term denoted by $B^2(f)$. It measures how much the average of $f(\mathbf{x}, T)$ over learning sets differs from the structural part of y given by $f^*(\mathbf{x})$. The second is the variance term $V(f)$ which is a measure of the fluctuation of $f(\mathbf{x}, T)$ around its average $\bar{f}(\mathbf{x})$. So (3) gives the decomposition

$$\bar{P}E(f) = \sigma^2 + B^2(f) + V(f) \quad (4)$$

where σ^2 is the noise variance. Now, denote the training set by $(\mathbf{Y}', \mathbf{X}')$ where $\mathbf{Y}'$ is the N-long vector of outputs, and $\mathbf{X}'$ the N-long array of input vectors, and let

$$\bar{f}(\mathbf{x}, \mathbf{X}') = E_{\mathbf{Y}}(f(\mathbf{x}, \mathbf{Y}', \mathbf{X}')) \quad (5)$$

That is, $\bar{f}(\mathbf{x}, \mathbf{X}')$ is gotten by holding the inputs constant and integrating over the output distribution. In the identity

$$V(f) = E_{T,\mathbf{X}}(f(\mathbf{X}, T) - \bar{f}(\mathbf{X}, \mathbf{X}'))^2 + E_{\mathbf{X},\mathbf{X}'}(\bar{f}(\mathbf{X}, \mathbf{X}') - \bar{f}(\mathbf{X}))^2 \quad (6)$$

we identify the first term as the variability due to the outputs and the 2nd as the residual variance due to the inputs and write

$$V(f) = V_{\mathrm{O}}(f) + V_{\mathrm{I}}(f). \quad (7)$$

The output variability term can be thought of this way: fix the inputs $\mathbf{X}'$ and compute the variation around the mean as outputs $\mathbf{Y}'$ vary over their conditional distribution given $\mathbf{X}'$. Then average this variation over the distribution of $\mathbf{X}'$.

2.2. Classification

The prediction error is defined as the misclassification rate:

$$PE(f(\cdot, T)) = P_{Y,\mathbf{X}}(Y \neq f(\mathbf{X}, T)) \tag{8}$$

The right hand side of (8) can be written as

$$E_{\mathbf{X}} P_Y(Y \neq f(\mathbf{X}, T) \mid \mathbf{X}) \tag{9}$$

Writing

$$P_Y(Y = f(\mathbf{X}, T) \mid \mathbf{X} = \mathbf{x}) = \sum_j P(Y = j \mid \mathbf{x}) I(f(\mathbf{x}, T) = j) \tag{10}$$

where $I(\cdot)$ is the 0-1 indicator function and taking expectation of (10) with respect to T gives

$$P_{T,Y}(Y = f(\mathbf{X}, T) \mid \mathbf{X} = \mathbf{x}) = \sum_j P(Y = j \mid \mathbf{x}) P_T(f(\mathbf{x}, T) = j). \tag{11}$$

To simplify notation, write

$$P(j \mid \mathbf{x}) = P(Y = j \mid \mathbf{x}), \qquad P(j \mid f, \mathbf{x}) = P_T(f(\mathbf{x}, T) = j)$$

and let $j^*(\mathbf{x}) = \arg\max P(j \mid \mathbf{x})$. Then

$$P_{T,Y}(Y \neq f(\mathbf{X}, T) \mid \mathbf{X} = \mathbf{x}) = 1 - P(j^* \mid \mathbf{x}) + \sum_j (P(j^* \mid \mathbf{x}) - P(j \mid \mathbf{x})) P(j \mid f, \mathbf{x}) \tag{12}$$

leading to:

$$\bar{P}E(f) = PE^* + E_{\mathbf{X}}\left(\sum_j (P(j^* \mid \mathbf{X}) - P(j \mid \mathbf{X})) P(j \mid f, \mathbf{X})\right) \tag{13}$$

where PE^* is the Bayes rate and the second term, which is non-negative, is the excess over the Bayes rate resulting from the use of the non-optimal classifier f. Let $\hat{j}(\mathbf{x}) = \arg\max P(j \mid f, \mathbf{x})$. That is, if the classifiers based on a large set of replicate training sets voted, the plurality of votes at input $\mathbf{x}$ would go to $\hat{j}(\mathbf{x})$. Then, the second term in (13) can be written as the sum of

$$E_{\mathbf{X}}(P(j^* \mid \mathbf{X}) - P(\hat{j} \mid \mathbf{X})) P(\hat{j} \mid f, \mathbf{X})) \tag{14}$$

and

$$E_{\mathbf{X}}\left(\sum_{j\neq\hat{j}}(P(j^*\,|\,\mathbf{X})-P(j\,|\,\mathbf{X}))P(j\mid f,\mathbf{X})\right) \tag{15}$$

The first term (14) we call the *bias* (B). If $j^*\neq\hat{j}$ then the class that got the most votes at $\mathbf{x}$ is not the optimal choice. Thus, at $\mathbf{x}$ the classifier is systematically wrong. The second term we call the *spread*(S) rather than variance, since it does not have the properties usually associated with the variance in regression.

Classifiers that have a large range of models to fit to the data usually have small bias. Their error comes from the spread. That is, at an input $\mathbf{x}$, while $j^*=\hat{j}$, there are too many votes for classes other than $\hat{j}$. Thus, for low bias classifiers, the key to increasing accuracy is in reducing the spread while keeping the bias low.

The idea behind reducing the spread is this—consider the classifier $f(\mathbf{x})=j'(\mathbf{x})$. That is, we assume we can generate endless replicate training sets of size N and define $f(\mathbf{x})$ to be the class getting the plurality vote at $\mathbf{x}$. Then $f(\mathbf{x})$ has zero spread, and its bias increases to $E_{\mathbf{X}}(P(j^*\,|\,\mathbf{X})-P(\hat{j}\,|\,\mathbf{X}))$. But if $j^*=\hat{j}$ for most $\mathbf{x}$, than the bias term remains small. So if we could generate a large number of replicate training sets, then we could drive the spread to zero.

But generating a large number of replicate data sets is difficult. Bagging tries to imitate this, but the bootstrapped training sets are a rough approximation. Suppose that instead, the inputs are held fixed and replicate sets of outputs are generated. The resulting classifier is

$$\tilde{j}(\mathbf{x},\mathbf{X}')=\arg\max_j P_{\mathbf{Y}'}(f(\mathbf{x},(\mathbf{Y}',\mathbf{X}'))=j). \tag{16}$$

Using this classifier will cut down on the spread, although not as much as using $j'(\mathbf{x})$. Denote by $\Delta\bar{PE}$ the decrease in $\bar{PE}$ gotten by using $j'(\mathbf{x})$ instead of the original classifier, by B the bias of this predictor, by $\Delta_{\mathrm{O}}\bar{PE}$ the reduction using $\tilde{j}(\mathbf{x},\mathbf{X}')$, and let the residual change due to input variability be $\Delta_{\mathrm{I}}\bar{PE}=\Delta\bar{PE}-\Delta_{\mathrm{o}}\bar{PE}$. Therefore, we have the decomposition:

$$\bar{PE}=PE^*+B+\Delta_{\mathrm{O}}\bar{PE}+\Delta_{\mathrm{I}}\bar{PE} \tag{17}$$

where $\Delta_{\mathrm{O}}\bar{PE}$ and $\Delta_{\mathrm{I}}\bar{PE}$ are measures of the relative importance of the output and input variability.

3. Output variability in synthetic data

The contribution of output variability to the prediction error is difficult to measure unless the underlying structure of the data is known. With synthetic data many replications are possible, either of the training set or of the outputs. We define and use synthetic data to give estimates of the output and input contributions and use unpruned CART as our prediction algorithm.

Table 1. Contributions to *PE* (%).

Data set	Noise var.	Bias	Output var.	Input var.
Friedman #1	8.3	25.3	47.8	18.7
Friedman #2	48.0	0.0	51.2	0.5
Friedman #3	24.4	10.7	49.1	15.5

3.1. Regression

Three synthetic data sets were used in this experiment. The structure of these three data sets was introduced in Friedman (1991). They are also used and described in Breiman (1996a) and referred to as Friedman #1, #2, #3. All use 200 instances in the training set. Friedman #1 has 10 inputs. The other two have 4.

From Section 2, we have

$$\bar{PE} = Ee^2 + B^2 + V_{\mathrm{O}} + V_{\mathrm{I}}$$

In the present Monte Carlo experiments which generated all entries in Table 1, a 15000 member evaluation set was generated to estimate the Y, $\mathbf{X}$ expectations. The training sets are of size 200 and 50 of them were generated and averaged over in the computations for each data set. Table 1 gives the percentage of the contributions to the total prediction error.

In the first and third data sets, the output variance was about 75% of the total variance. In the second it was 99%.

3.2. Classification

Recall that

$$\bar{PE} = PE^* + B + \Delta_{\mathrm{O}}\bar{PE} + \Delta_{\mathrm{I}}\bar{PE}$$

Using three synthetic two-class data sets defined in Breiman (1998) and called twonorm, threenorm, and ringnorm, a Monte Carlo experiment was carried out on each of these data sets to evaluate the components of the prediction error.

Each of these data sets was used to produce 100 training sets of 300 with equal probability of each class. To get the predictor based on voting only over the outputs, for each input vector $\mathbf{x}$ in the training set, the probability $p(\mathbf{x})$ of class #1 was computed. In the iterations with that training set, each time a coin was flipped with probability $p(\mathbf{x})$ of heads. If it came up heads, class label #1 was assigned, otherwise class label #2. Then after 100 iterations with a training set, holding the inputs fixed, a vote was taken and the most popular class assigned as the predictor.

An evaluation set of 15,000 was generated to estimate the components. Table 2 gives the estimates of the components of $\bar{PE}$.

Table 2. Components of $\bar{PE} \times 100$.

Data set	$\bar{PE}$	PE^*	Bias	$\Delta_O \bar{PE}$	$\Delta_I \bar{PE}$
twonorm	23.1	2.2	2.0	7.5	11.5
threenorm	34.7	10.7	2.6	13.7	7.6
ringnorm	25.0	1.4	4.0	4.6	14.9

Table 3. Percent of probabilities less than th.

Data sets	th = .1	th = .01	th = .001
twonorm	92.5	79.3	60.3
threenorm	64.4	33.8	7.3
ringnorm	96.0	86.2	64.9

This table shows that for the 1st and 3rd data sets the output variability has about half of the effect of the input variability. To illustrate the reason, the percent of instances in the training set such that $\min(p(\mathbf{x}), 1 - p(\mathbf{x})) < \text{th}$ was computed for each data set for th = .1, .01, .001 and given in Table 3 (averaged over 10 training sets).

The large percentages of small values of $\min(p(\mathbf{x}), 1-p(\mathbf{x}))$ in data sets #1 and #3 reduces the output variability. For instance, the outputs such that $\min(p(\mathbf{x}), 1 - p(\mathbf{x})) < .01$ will rarely get changed in 100 iterations. Only the instances not in the first column will have appreciable variability. In ringnorm this is only 4% of the instances.

To push the point to the extreme, it's possible to have data sets where the classes are perfectly separated so that the Bayes rate is zero. Then output variability will be zero. But, surprisingly, this does not have the consequence that randomizing outputs will not significantly reduce prediction error. To illustrate, instead of flipping outputs on the basis of their true probability, they are flipped with a constant probability .25 irrespective of the input value.

More specifically, in each iteration, a training set is generated from the underlying probability for the synthetic data. From this training set, 100 new training sets of size 300 are generated by randomly flipping outputs with probability .25. Each of these training sets is used to grow a classifier. The 100 classifiers grown this way will have a different structure than the classifiers grown using training sets drawn from the true underlying probability.

We can compute the components of error of these new classifiers in terms of prediction for the original input-output relationship by using a large test set generated from the true underlying input-output distribution.

There are some interesting changes from the results in Table 2. The first column is the average prediction error of the individual trees. These are higher in Table 4 because they have been trained on data with substantial noise added to the outputs. The next change is that the bias has been significantly reduced. The biggest effect is in the 4th column which measures the decrease in error due to voting over the 100 classifier outputs. This is the

Table 4. Components of $\bar{PE} \times 100$ (new flipping scheme).

Data set	$\bar{PE}$	PE*	Bias	$\Delta_O \bar{PE}$	$\Delta_I \bar{PE}$
twonorm	33.1	2.2	0.6	27.7	2.6
threenorm	40.6	10.8	0.3	20.8	6.0
ringnorm	34.1	1.4	1.4	28.3	3.8

largest component in Table 4, but had moderate to minor effect in Table 2. Finally, the 5th column shows that the effect of input variability has dropped considerably.

These results raise the interesting possibility that although the errors in the individual classifiers may be raised by introducing noise into the output variables, voting over the ensemble of classifiers produced this way may produce substantial increases in accuracy. In fact, introducing an appreciable amount of artificial output variability produces, as we will see in the following sections, as large or larger reduction in error rate than is given by bagging, which works on both inputs and outputs.

4. Experimental results from output smearing

We ran output smearing on a variety of data sets, both regression and classification. In both situations, a simple method was used to provide extra output variability. Yet the results were generally better than bagging.

4.1. Regression

The procedure used here was to first compute the sample standard deviation of the outputs in the data set. A more robust estimate was formed by doing a second pass which rejected from the standard deviation computation any output more than 2.5 original standard deviations from the original mean. Then new outputs were generated as:

$$y'_n = y_n + z_n \cdot \text{sd}, \quad n = 1, \ldots, N$$

where sd is the standard deviation estimate and the $\{z_n\}$ are independent unit normals. A maximal tree is grown using the new set of outputs. This is repeated 100 times. Then, for any test instance the predicted output is given by the average over the predictions of these 100 trees. The data sets we used in the experiment are briefly described in Table 5.

For the first three data sets, we estimated generalization error by leaving out a randomly selected 10% of the instances, constructing the 100 trees on the remaining 90% and using the left-out 10% as a test set. This was repeated 100 times and the test set mean-squared errors averaged.

The robotarm data (supplied by Michael Jordan) has a separate test set of 5000 instances. The 100 trees were constructed on the 15000 member training set and the error estimated using the test set. For the last three synthetic data sets, in each run a 200 instance training

Table 5. Data set summaries.

Data set	Size	No. inputs
Ozone	330	8
Housing	506	12
Servo	167	4
Robotarm	15000	12
Friedman #1	200	10
Friedman #2	200	4
Friedman #3	200	4

Table 6. Mean-square error estimates.

Data set	Bagging	Smearing
Ozone	18.1	17.4
Housing	10.6	10.3
Servo	98.3	89.7
Robotarm	4.72	4.64
Friedman #1	6.23	5.01
Friedman #2	21.4e3	22.2e3
Friedman #3	25.1e-3	23.3e-3

set and 2000 instance test set were generated. The 100 trees were built on the training set and evaluated using the test set. This was repeated 50 times and the results averaged. The error estimates are given in Table 6 and compared with the use of bagging using the same procedures for error estimation and 100 bagged trees per run. Except for one synthetic data set, smearing produces lower error rates than bagging. The reduction is not spectacular, but consistent.

4.2. Classification

To emulate output variability in classification, classification was turned into a multiple output regression problem. If there were J classes, there are J outputs. If the class of the nth instance was j, then the jth output is one with zeroes in the other outputs. The splits were based on minimizing the total sum-of-squares. For class label 0-1 outputs this reduces to the Gini criterion.

Given J-class data, a standard deviation measure is computed for each class. If p_j is the proportion of the instances in class j, then define

$$\text{sd}_j = 2\sqrt{p_j(1 - p_j)}.$$

Table 7. Data set summaries.

Data set	Size	No. inputs	No. classes	Test set
sonar	208	60	2	10%
glass	214	9	6	10%
breast (Wis)	699	9	2	10%
ionosphere	351	34	2	10%
soybean	683	35	19	10%
vehicle	846	18	4	10%
vowel	990	10	11	10%
letters	15000	16	26	5000
dna	2000	60	3	1186
satellite	4435	36	6	2000
digit	7291	256	10	2007
wave	300	21	3	3000
twonorm	300	20	2	3000
threenorm	300	20	2	3000
ringnorm	300	20	2	3000

The new outputs are given by

$$y'_{j,n} = y_{j,n} + z_{j,n} \cdot \mathrm{sd}_j \quad j = 1, \ldots, J \quad n = 1, \ldots, N$$

where the $\{z_{j,n}\}$ are independent unit normals. The predictions of the trees built using smeared outputs are no longer 0-1. A class prediction is made based on which output is the largest.

As in regression, 100 trees are built based on 100 sets of smeared outputs. Given a new test instance, these trees vote and the predicted class is the one having the plurality of the votes. The data set summarized in Table 7 were used in the experiment:

For the first 8 data sets, 10% was left out in each run generating 100 trees and used as a test set. The results were averaged over 100 runs. For the next 4 data sets, there were only one run generating 100 trees. The test set was then used to get the error estimate. The last four data sets are synthetic, with equal probabilities of each class. For these, a training set of 300 and test set of 3000 were newly generated for each run. The error was averaged over 50 runs. The results are given in Table 8 and compared with bagging whose error estimates are derived using the same procedure as for smearing.

For comparison, the results for CART are also listed. These were gotten by leaving out 10% in the first 8 data sets, using 10-fold cross-validation to determine the size of the tree grown on all of the data (except for the 10% test set) and then running the test set down the designated tree. This was repeated 100 times and the results averaged. Thus, 1100 trees were grown for each of the first 8 data sets. For the next 4 data sets a single CART tree was selected using 10-fold cross-validation and the test set used to

Table 8. Misclassification error (%).

Data set	Bagging	Smearing	CART
sonar	20.1	16.3	29.2
glass	23.2	22.6	29.8
breast (Wis)	4.1	3.4	5.7
ionosphere	7.9	6.9	14.5
soybean	6.8	5.6	9.3
vehicle	15.4	15.8	30.1
vowel	8.0	4.4	20.9
letters	6.4	5.2	13.2
dna	5.1	5.1	6.2
satellite	10.0	9.2	14.0
digit	10.5	9.9	16.5
waveform	19.5	18.6	29.8
twonorm	6.9	5.2	23.2
threenorm	19.5	18.1	33.9
ringnorm	9.9	7.0	22.1

estimate the error. For the synthetic data sets, the procedure used a 300 instance training set and 3000 instance training set with a cross-validated tree selected using the training set only.

Smearing is consistently better than bagging—sometimes significantly better.

5. Experimental results from flipping outputs

Flipping outputs refers to changing the class label of an input. In our experiments, it was found that accuracy was improved by flipping outputs so that the class proportions remained about the same. That is, if class j outputs had a certain probability of being changed, then the changes of the other class outputs into class j would keep the proportion of class j instances in the training set about equal to the original proportion of class j instances.

To do this, the following random flipping regime was used: let $c(k)$ be the proportion of class k labels in the training set, and denote by $p(j \mid k)$ the probability that if the class label of an instance is k, we will flip it into label j. Then set the value of a parameter w and let

$$p(j \mid k) = w^* c(j), \quad j \neq k$$
$$p(k \mid k) = 1 - w(1 - c(k)).$$

Some algebra shows that the only invariant probability distribution under these transitions is $\{c(k)\}$. If the flip rate fr is defined to be the proportion of instances that have their output

Table 9. Misclassification error (%).

Data set	Smearing	Flipping	Flip rate
sonar	16.3	17.3	.20
glass	22.6	23.1	.25
breast(Wis)	3.4	3.7	.15
ionosphere	6.9	7.5	.20
soybean	5.6	6.2	.25
vehicle	15.8	15.5	.25
vowel	4.4	4.0	.50
letters	5.2	4.7	.45
dna	5.1	4.9	.40
satellite	9.2	9.3	.30
digit	9.9	8.9	.40
waveform	18.6	18.5	.30
twonorm	5.2	5.2	.25
threenorm	18.1	18.3	.20
ringnorm	7.0	5.7	.25

flipped, then

$$w = \frac{\mathrm{fr}}{(1 - sq)} \tag{18}$$

where $sq = \sum_j c(j)^2$. To see this, note that the proportion of class k labels that get flipped is $w(1 - c(k))$. The total proportion that get flipped is the sum of $w(1 - c(k))c(k)$. Putting this sum equal to fr and solving for w gives (18).

Unlike smearing outputs where one size fits all (more or less) the success of flipping outputs depends on the value of the flip rate selected. We tried values of the flip rate going from .10 to .50 in increments of .05 and report the lowest error rate over 100 iterations of leave out 10% or test set evaluations. This is done in Table 9 which also compares the results to smearing and gives the value of the flip rate used.

It's pretty much a dead heat between smearing and flipping. There are some interesting error decreases with flipping on the vowel, letters, digit, and ringnorm data sets. But having to choose the optimum value of the flip rate makes flipping less attractive.

6. Remarks

Since the advent of bagging there have been questions as to whether similar results could be gotten from just randomizing the outputs—in particular, by flipping the outputs. Freund and Schapire (1998) list this as an interesting unsolved problem in combining predictors.

I put this off for a while because it seemed that the selection of which outputs to flip was a difficult problem and I couldn't see a way around it. I would have rejected the idea of giving each output a flip probability that depended only on the class proportions in the training set as being too simplistic to work well. But using the synthetic data sets, I investigated whether knowledge of the output probabilities could be combined with the flip method used in Section 5 to further increase accuracy. I could find no combination that gave improvement.

It's interesting that while the true output variability may be small, adding substantial output variability gives a decrease in error rates as large or larger than bagging. We do have an clue as to why this works better than bagging. The theoretical and experimental results concerning bagging show that while it reduces variance it has little effect on bias, and may even increase it a little. In the experiments on synthetic data in Section 3, randomizing the outputs by flipping gave significant decreases in bias. The bias-reduction capability may explain the improvement.

A natural follow-up is to try and combine bagging and output randomization. I tried combining flipping with bagging and got small unexciting improvements in most cases. The essential problem in combining classifiers is in growing a suitably diverse ensemble on base classifiers, and that there are many ways of doing this work. For instance Dietterich (1998) found that randomizing split selections in growing trees gave results often better than bagging.

None of the above methods give as small an error as does the Adaboost algorithm (Freund & Schapire, 1995, 1996) or half&half bagging (Breiman, 1998b) whose reasons for working still remains a bit of a mystery (see Breiman, 1997). For instance, Adaboost, run on the 15 data sets used above, averages a 5% lower misclassification rate than smearing.

I have been pointed by a referee to two other studies that have looked at the effect of adding noise to the outputs. One (An, 1996) is an analytic study of the effects of adding noise when neural nets for regression are trained. He concludes that "zero-mean and constant variance noise added to the desired output have no effect on generalization." This conclusion refers to the effect of small added output noise applied to the training of a single model. The other (Grossman & Lapedes, 1993) uses flipping outputs on neural net classifiers to develop diagnostics for overfitting-underfitting of a single model. The goal of this present paper is considerably different than either.

References

An, G. (1996). The effects of adding noise during backpropagation training on generalization performance. *Neural Computation*, *6*, 643–674.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, *26*(2), 123–140.

Breiman, L. (1996b). The heuristics of instability in model selection. *Annals of Statistics*, *24*, 2350–2383.

Breiman, L. (1997). Prediction games and arcing algorithms. Technical Report 504, Statistics Department, University of California at Berkeley. Available at www.stat.berkeley.edu

Breiman, L. (1998a). Arcing classifiers (with discussion). *Annals of Statistics*, *26*, 801–849.

Breiman, L. (1998b). Half and half bagging and hard boundary points. Technical Report 534, Statistics Dept. Univ. of Calif. at Berkeley.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Chapman and Hall.

Dietterich, T. (1998). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 1–22.

Freund, Y. & Schapire, R. (1997). A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Freund, Y. & Schapire, R. (1996). Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference* (pp. 148–156).

Freund, Y. & Schapire, R. (in press). Discussion of "Arcing Classifiers" by L. Breiman. *Annals of Statistics*.

Friedman, J. (1991). Multivariate adaptive regression splines (with discussion). *Annals of Statistics*, *19*, 1–141.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, *4*, 1–58.

Grossamn, T. & Lapedes, A. (1993). Use of bad training data for better predictions. *NIPS*, *6*, 343–350.

Received May 14, 1998
Accepted July 12, 1999
Final manuscript July 12, 1999