RANDOM FORESTS--RANDOM FEATURES

Leo Breiman
Statistics Department
University of California
Berkeley, CA 94720

Technical Report 567
September 1999

Abstract

Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large. The error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them. Using a random selection of features to split each node yields error rates that compare favorably to Adaboost, but are more robust with respect to noise. Internal estimates monitor error, strength, and correlation and these are used to show the response to increasing the number of features used in the splitting. These ideas are al;so applicable to regression.

1. Random Forests

1.1 Introduction

Significant improvements in classification accuracy have resulted from growing an ensemble of trees and letting them vote for the most popular class. In order to grow these ensembles, often random vectors are generated that govern the growth of each tree in the ensemble. An early example is bagging (Breiman [1996]) where to grow each tree a random selection (without replacement) is made from the examples in the training set.

Another example is random split selection (Detterich[1998]) where at each node the split is selected at random from among the K best splits. Breiman [1999] generates new training sets by randomizing the outputs in the original training set. An other approach is to select the training set from a random set of weights on the examples in the training set. In an important paper on written character recognition, Amit and Geman [1997] define a large number of features and search over a random selection of these for the best split at each node. This latter paper has been influential in my thinking.

The common element in all of these procedures is that for the kth tree, a random vector Θ_k is generated, independent of the past random vectors $\Theta_{1, \ldots ,}\Theta_{k-1}$ but with the same distribution, and a tree is grown using the training set and Θ_k, resulting in a classifier $h(\mathbf{x},\Theta_k)$ where $\mathbf{x}$ is an input vector. For instance, in bagging, the random vector Θ is generated as the counts in N boxes resulting from N darts thrown at random at the boxes, where N is number of examples in the training set. In random split selection Θ consists of a number of independent random integers between 1 and K.

After a large number of trees is generated, they vote for the most popular class. We call these procedures **random forests.**

Definition 1.1 *A random forest is a classifier consisting of a collection of tree-structured classifiers* $\{h(\mathbf{x},\Theta_k), k=1, \ldots\}$ *where the* $\{\Theta_k\}$ *are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input* $\mathbf{x}$.

1.1 Outline of Paper

Section 2 gives some theoretical background for random forests. Use of the Strong Law of Large Numbers shows that they always converge so that overfitting is not a problem. We give a simplified and extended version of the Amit and Geman [1997] analysis to show that the accuracy of of a random forest depends on the strength of the individual tree classifiers and a measure of the dependence between them.

Section 3 introduces forests using the random selection of features at each node to determine the split. An important question is how many features to select at each node. For guidance, internal estimates of the generalization error, classifier strength and dependence are computed. These are called out-of-bag estimates and are reviewed in Section 4. Sections 5 and 6 give empirical results for two different forms of random features. The first uses random selection from the original inputs--the second uses random linear combinations of inputs. The results compare favorably to Adaboost.

The results turn out to be insensitive to the number of features selected to split each node. Usually, selecting one or two features gives near optimum results. To explore this and relate it to strength and correlation, an empirical study is carried out in Section 7.

Adaboost has no random elements and grows an ensemble of trees by successive reweightings of the training set where the current weights depend of the past history of the ensemble formation. But just as a deterministic random number generator can give a good imitation of randomness, my belief is that in its later stages Adaboost is emulating a random forest. Evidence for this conjecture is given in Section 7.

Important recent problems, i.e.. medical diagnosis and document retrieval , often have the property that there are many input variables, often in the hundreds or thousands, with each one containing only a small amount of information. A single tree classifier will then have accuracy only slightly better than a random choice of class. But combining trees grown using random features can produce improved accuracy. In Section 8 we experiment on a simulated data set with 1000 input variables, 1000 examples in the training set and a 4000 example test set. Accuracy comparable to the Bayes rate is achieved.

Section 9 looks at random forests for regression. A bound for the mean squared generalization error is derived which shows that the decrease in error from the individual trees in the forest depends of the correlation between residuals and the mean squared error of the individual trees. Empirical results for regression are in Section 10. Concluding remarks are given in Section 11.

2 Characterizing the Accuracy of Random Forests

2.1 Random Forests Converge

Given an ensemble of classifiers $h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_K(\mathbf{x})$, and with the training set drawn at random from the distribution of the random vector Y,**X** define the margin function as

$$mg(\mathbf{X},Y) = av_k I(h_k(\mathbf{X})=Y) - \max_{j \neq Y} av_k I(h_k(\mathbf{X})=j).$$

where $I(\bullet)$ is the indicator function. The margin measures the extent to which the average number of votes at $\mathbf{X},Y$ for the right class exceeds the average vote for any other class. The generalization error is given by:

$$PE^* = P_{\mathbf{X},Y}(mg(\mathbf{X},Y) < 0)$$

where the subscripts $\mathbf{X},Y$ indicate that the probability is over the $\mathbf{X},Y$ space.

In random forests $h_k(\mathbf{X}) = h(\mathbf{X},\Theta_k)$. For a large number of trees , it follows from the Strong Law of Large Numbers and the tree structure that:

Theorem 1.2 *As the number of trees increases, for almost surely all sequences* $\Theta_1, \ldots$ PE^* *converges to*

$$P_{\mathbf{X},Y}(P_\Theta(h(\mathbf{X},\Theta)=Y) - \max_{j \neq Y} P_\Theta(h(\mathbf{X},\Theta)=j) < 0) \tag{1}$$

Proof: see Appendix I.

This result explains why random forests do not overfit as more trees are added, but produce a limiting value of the generalization error.

2.2 Strength and Correlation

For random forests, an upper bound can be derived for the generalization error in terms of two parameters that are measures of how accurate the individual classifiers are and of the dependence between them. The interplay between these two gives the foundation for understanding the workings or random forests. We build on the analysis in Amit and Geman[1997]

Definition 2.1. *The margin function for a random forest is*

$$mr(\mathbf{X},Y) = P_\Theta(h(\mathbf{X},\Theta)=Y) - \max_{j \neq Y} P_\Theta(h(\mathbf{X},\Theta)=j)$$

and the strength of the set of classifiers $\{h(\mathbf{x},\Theta)\}$ *is*

$$s = E_{\mathbf{X},Y}\, mr(\mathbf{X},Y)$$

Assuming $s \geq 0$, Chebychev's inequality gives

$$PE^* \leq \mathrm{var}(mr)/s^2 \tag{2}$$

The variance of mr can be bounded by a more interesting expression. Let

$$\hat{j}(\mathbf{X},Y) = \arg\max_{j \neq Y} P_\Theta(h(\mathbf{X},\Theta) = j)$$

so

$$\begin{aligned} mr(\mathbf{X},Y) &= P_\Theta(h(\mathbf{X},\Theta)=Y) - P_\Theta(h(\mathbf{X},\Theta)=\hat{j}(\mathbf{X},Y)) \\ &= E_\Theta[I(h(\mathbf{X},\Theta)=Y) - I(h(\mathbf{X},\Theta)=\hat{j}(\mathbf{X},Y))]. \end{aligned}$$

<u>Definition 2.2</u> *The raw margin function is;*

$$rmg(\Theta,\mathbf{X},Y) = I(h(\mathbf{X},\Theta)=Y) - I(h(\mathbf{X},\Theta)=\hat{j}(\mathbf{X},Y))$$

For any function f the identity

$$[E_\Theta f(\Theta)]^2 = E_{\Theta,\Theta'} f(\Theta) f(\Theta')$$

holds where Θ,Θ' are independent with the same distribution, implying that

$$mr(\mathbf{X},Y)^2 = E_{\Theta,\Theta'}\, rmg(\Theta,\mathbf{X},Y)\, rmg(\Theta',\mathbf{X},Y) \tag{3}$$

Using (3) gives

$$\begin{aligned} \mathrm{var}(mr) &= E_{\Theta,\Theta'}(\mathrm{cov}_{\mathbf{X},Y}\, rmg(\Theta,\mathbf{X},Y)\, rmg(\Theta',\mathbf{X},Y)) \\ &= E_{\Theta,\Theta'}(\rho(\Theta,\Theta')\, sd(\Theta)\, sd(\Theta')) \end{aligned} \tag{4}$$

where $\rho(\Theta,\Theta')$ is the correlation between $rmg(\Theta,\mathbf{X},Y)$ and $rmg(\Theta',\mathbf{X},Y)$ holding Θ,Θ' fixed and $sd(\Theta)$ is the standard deviation of $rmg(\Theta,\mathbf{X},Y)$ holding Θ fixed. Then,

$$\begin{aligned} \mathrm{var}(mr) &= \bar{\rho}(E_\Theta sd(\Theta))^2 \\ &\leq \bar{\rho} E_\Theta \mathrm{var}(\Theta) \end{aligned} \tag{5}$$

where $\bar{\rho}$ is the mean value of the correlation. Write

$$E_{\Theta}\operatorname{var}(\Theta) \le E_{\Theta}(E_{\mathbf{X},Y} rmg(\Theta,\mathbf{X},Y))^2 - s^2$$
$$\le 1 - s^2.$$

Theorem 2.3 *An upper bound for the generalization error is given by*

$$PE^* \le \bar{\rho}(1-s^2)/s^2 \qquad (6)$$

Although the bound is likely to be loose, it fulfills the same suggestive function for random forests as VC-type bounds do for other types of classifiers. It shows that the two ingredients involved in the generalization error for random forests are the strength of the individual classifiers in the forest, and the correlation between them in terms of the raw margin functions. The c/s2 ratio is the correlation divided by the square of the strength. In understanding the functioning of random forests, this ratio will be a helpful guide.

Definition 5.3 *The* c/s2 *ratio for a random forest is defined as*

$$c/s2 = \bar{\rho}/s^2$$

There are simplifications in the two class situation. The margin function is

$$mr(\mathbf{X},Y) = 2P_{\Theta}(h(\mathbf{X},\Theta)=Y) - 1$$

The requirement that the strength is positive becomes the familiar $E_{\mathbf{X},Y}P_{\Theta}(h(\mathbf{X},\Theta)=Y) > .5$. The raw margin function is $2I(h(\mathbf{X},\Theta)=Y) - 1$and the correlation is between $I(h(\mathbf{X},\Theta)=Y)$ and $I(h(\mathbf{X},\Theta')=Y)$.

For more than two classes, the measure of strength defined above depends on the forest as well as the individual trees since it is the forest that determines $\hat{j}(\mathbf{X},Y)$. Another approach is possible. Write

$$PE^* = P_{\mathbf{X},Y}(P_{\Theta}(h(\mathbf{X},\Theta)=Y) - \max_{j \ne Y} P_{\Theta}(h(\mathbf{X},\Theta)=j) < 0)$$
$$\le \sum_j P_{\mathbf{X},Y}(P_{\Theta}(h(\mathbf{X},\Theta)=Y) - P_{\Theta}(h(\mathbf{X},\Theta)=j) < 0).$$

Define

$$s_j = E_{\mathbf{X},Y}(P_{\Theta}(h(\mathbf{X},\Theta)=Y) - P_{\Theta}(h(\mathbf{X},\Theta)=j))$$

to be the strength of the set of classifiers $\{h(\mathbf{x},\Theta)\}$ relative to class j. Note that this definition of strength does not depend on the forest. Using Chebyshev again leads to

$$PE^* \leq \sum_j \mathrm{var}(P_\Theta(h(\mathbf{X},\Theta)=Y) - P_\Theta(h(\mathbf{X},\Theta)=j))/s_j^2 \qquad (7)$$

and using identities similar to those used in deriving (5), the variances in (7) can be expressed in terms of average correlations. We did not use estimates of the quantities in (7) in our empirical study, but think they would be interesting in a many class problem.

3. Using Random Features

Some random forests reported on in the literature have consistently lower generalization error than others. For instance, random split selection (Dieterrich[1998]) does better than bagging. Breiman's introduction of random noise into the outputs (Breiman [1998c] also does better. But none of these three forests do as well as Adaboost (Freund and Schapire[1996]) or other arcing algorithms that work by perturbing the training set (see Breiman [1998b], Dieterrich [1998]), Bauer and Kohavi [1999]).

To improve accuracy, the randomness injected has to minimize correlation while maintaining strength. The forests studied here consist of using randomly selected inputs or combinations of inputs at each node to grow each tree. The resulting forests give accuracy that compare favorably with Adaboost. This class of procedures has desirable characteristics;

i) Its accuracy as good as Adaboost and sometimes better.
ii) Its relatively robust to outliers and noise.
iii) Its faster than bagging or boosting.
iv) It gives a useful internal estimates of error, strength, correlation
v) Its simple.

Amit and Geman [1997] grew shallow trees for handwritten character recognition using random selection from a large number of geometrically defined features to define the split at each node. Although my implementation is different and not problem specific, It was their work that provided the start for my ideas

3.1 Using Out-Of-Bag Estimates to Monitor Error, Strength, and Correlation

In my experiments with random forests, bagging (Breiman[1996]) is used in tandem with random feature selection. Each new training set is drawn, with replacement, from the original training set. Then a tree is grown on the new training set using random feature selection. The trees grown are not pruned.

There are two reasons for using bagging. The first is that the use of bagging seems to enhance accuracy when random features are used. The second is that bagging can be used to give ongoing estimates of the generalization error of the combined ensemble of trees, as well as estimates for the strength and correlation.

Assume a method for constructing a classifier from any training set. Given a specific training set T form bootstrap training sets T_k, construct classifiers h(**x**, T_k) and let these vote to form the bagged predictor. For each y,**x** in the training set, aggregate the votes only over those classifiers for which T_k does not containing y,**x.** Call this the out-of-bag classifier. Then the out-of-bag estimate for the generalization error is the error rate of the out-of-bag classifier on the training set.

Tibshirani [1996] and Wolpert and Macready[1996], proposed using out-of-bag estimates as an ingredient in estimates of generalization error. Wolpert and Macready worked on regression type problems and proposed a number of methods for estimating the generalization error of bagged predictors. Tibshirani used out-of-bag estimates of variance to estimate generalization error for arbitrary classifiers. The study of error estimates for bagged classifiers in Breiman[1996b], gives empirical evidence to show that the out-of-bag estimate is as accurate as using a test set of the same size as the training set.

In each bootstrap training set, about one-third of the instances are left out. Therefore, the out-of-bag estimates are based on combining only about one-third as many classifiers as in the ongoing main combination. Since the error rate decreases as the number of combinations increases, the out-of-bag estimates will tend to overestimate the current error rate. To get unbiased out-of-bag estimates, it is necessary to run past the point of where the test set error converges. But unlike cross-validation, where bias is present but its extent unknown, the out-of-bag estimates are unbiased.

Strength and correlation can also be estimated using out-of-bag methods. This gives internal estimates which are helpful in understanding classification accuracy and how to improve it. The details are given in Appendix II.

4.<u>**Random Forests Using Random Input Selection**</u>

The simplest random forest with random features is formed by selecting at random at each node a small group of input variables to split on. Grow the tree to maximum size and do not prune. Denote this procedure by Forest-RI. The size F of the group is fixed. Two values of F were tried. The first used only one randomly selected variable, i.e. F=1. The second took F to be the first integer less than $\log_2 M+1$, where M is the number of inputs.

Our experiments use 13 smaller sized data sets from the UCI repository, 4 larger sets separated into training and test sets, and 4 synthetic data The first 10 sets were selected because I had used them often in past research. 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
ecoli	336	--	7	8
votes	435	--	16	2
liver	345	--	6	2
letters	15000	5000	16	26
sat-images	4435	2000	36	6
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 13 smaller sized data sets, the following procedure was used: a random 10% of the data was set aside. On the remaining data random forests was run twice combining 100 trees--once with F=1, and the other with F=int($\log_2 M$+1). The set aside 10% was then put down each forest to get a test set error for both. The test set error selected corresponded to the lower value of the out-of-bag estimate in the two runs. This was repeated 100 times and the test set errors averaged. The same procedure was followed for the Adaboost runs which are based on combining 50 trees.

In the runs on the larger data sets, the random forest 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 random forests 100 trees were combined in each run--50 in Adaboost. The results of these runs are given in Table 2. The 2nd column are the results selected from the two group sizes by means of lowest out-of-bag error. The 3rd column is the test set error using just one random feature to grow the trees.

Table 2 Test Set Errors (%)

Data Set	Adaboost	Forest-RI	
		Selection	Single Input
glass	22.0	20.6	21.2
breast cancer	3.2	2.9	2.7
diabetes	26.6	24.2	24.3
sonar	15.6	15.9	18.0
vowel	4.1	3.4	3.3
ionosphere	6.4	7.1	7.5
vehicle	23.2	25.8	26.4
soybean	5.8	6.0	6.5
German credit	23.5	24.4	26.2
image	1.6	2.1	2.7
ecoli	14.8	12.8	13.0
votes	4.8	4.1	4.6
liver	30.7	25.1	24.7
letters	3.4	3.5	4.7
sat-images	8.8	8.6	10.5
zip-code	6.2	6.3	7.8
waveform	17.8	17.2	17.3
twonorm	4.9	3.9	3.9
threenorm	18.8	17.5	17.5
ringnorm	6.9	4.9	4.9

The error rates using random input selection compare favorably with Adaboost. The comparison might be even more favorable if the search is over more values of F instead of the preset two. But the procedure is not overly sensitive to the value of F. The average absolute difference between the error rate using F=1 and the higher value of F is less than 1% . The difference is most pronounced on the three large data sets.

The single variable test set results were included because in some of the data sets, using a single random input variable did better than using several. In the others, results were only slightly better than use of a single variable. It was surprising that using a single randomly chosen input variable to split on at each node could produce good accuracy.

Random input selection can be much faster than either Adaboost or Bagging. A simple analysis shows that the ratio of R2I compute time to the compute time of unpruned tree construction using all variables is $F^*\log_2(N)/M$ where F is the number of variables used in Forest-RI, N is the number of instances, and M the number of input variables. For zip-code data, using F=1, the ratio is .025, implying that Forest-RI is 40 times faster. An empirical check confirmed this difference. A Forest- RI run (F=1) on the zip-code data takes 4.0 minutes on a 250 Mhz Macintosh to generate 100 trees compared almost three hours for Adaboost. For data sets with many variables, the compute time difference may be important.

5. Random Forests Using Linear Combinations of Inputs

If there are only a few inputs, say M, taking F an appreciable fraction of M might lead an increase in strength but higher correlation. Another approach consists of defining more features by taking random linear combinations of a number of the input variables. That is, a feature is generated by specifying L, the number of variables to be combined. At a given node, L variables are randomly selected and added together with coefficients that are uniform random numbers on [-1,1]. F linear combinations are generated, and then a search is made over these for the best split. This procedure is called Forest-RC .

We use L=3 and F=2,8 with the choice for F being decided on by the out-of-bag estimate. We selected L=3 because with O($M^{3)}$ different combinations of the input variables, larger values of F should not cause much of a rise in correlation while increasing strength. If the input variables in a data set are incommensurable, they are normalized by subtracting means and dividing by standard deviations, where the means and standard deviations are determined from the training set. The test set results are given in Table 3 where the 3rd column contains the results for F=2.

Table 2 Test Set Errors (%)

Data Set	Adaboost	Forest-RC Selection	Forest-RC Two Features
glass	22.0	24.4	23.5
breast cancer	3.2	3.1	2.9
diabetes	26.6	23.0	23.1
sonar	15.6	13.6	13.8
vowel	4.1	3.3	3.3
ionosphere	6.4	5.5	5.7
vehicle	23.2	23.1	22.8
soybean	5.8	5.8	6.1
German credit	23.5	22.8	23.8
image	1.6	1.6	1.8
ecoli	14.8	12.9	12.4
votes	4.8	4.1	4.0
liver	30.7	27.3	27.2
letters	3.4	3.4	4.1
sat-images	8.8	9.1	10.2
zip-code	6.2	6.2	7.2
waveform	17.8	16.0	16.1
twonorm	4.9	3.8	3.9
threenorm	18.8	16.8	16.9
ringnorm	6.9	4.8	4.6

Except for the three larger data sets, use of F=8 is superflous--F=2 achieves close to the minimum. On the larger data sets, F=8 gives better results. Further discussion is given in Section 6. Forest-RC does exceptionally well on the synthetic data sets. Overall, it compares more favorably to Adaboost than Forest-RI,

In Tables 1 and 2 there are some entries for which the selected entry is less than the one input value or with Forest-RC, less than the two feature value. The reason this happens is that when the error rates corresponding to the two values of F are close together, then the out-of-bag estimates will select a value of F almost at random.

6. **Empirical Results on Strength and Correlation**

The purpose of this section is to look at the effect of strength and correlation on the generalization error. Another aspect that we wanted to get more understanding of was the lack of sensitivity in the generalization error to the group size F. To conduct an empirical study of the effects of strength and correlation in a variety of data sets, out-of-bag estimates of the strength and correlation were used.

We begin by running Forest-RI on the Sonar data (60 inputs, 208 examples) using from 1 to 50 inputs. In each iteration, 10% of the data was split off as a test set. Then F, the number of random inputs selected at each node, was varied from 1 to 50. For each value of F, 100 trees were grown to form a random forest and the terminal values of test set error, strength, correlation, etc. recorded. Eighty iterations were done, each time removing a random 10% of the data for use as a test set, and all results averaged over the 80 repetitions. Altogether, 400,000 trees were grown in this experiment.

The top graph of Figure 1, plots the values of strength and correlation vs. F. The result is fascinating. Past about F=4 the strength remains constant--adding more inputs does not help. But the correlation continues to increase. The second graph plots the test set errors and the out-of-bag estimates of the error against F. The averaged test set errors are fairly variable. The out-of-bag estimates are more stable. Both show the same behavior--a small drop from F=1 out to F about 4-8, and then a general gradual increase. This increase in error tallies with the beginning of the constancy region for the strength.

Figure 2 has plots for similar runs on the breast data set where features consisting of random combinations of three inputs are used. The number of these features was varied from 1 to 25. Again, the correlation shows a slow rise, while after the strength stays virtually constant, so that the minimum error is at F=1. The surprise in these two figures is the relative constancy of the strength. Since the correlations are slowly but steadily increasing, the lowest error occurs when only a few inputs or features are used.

Since the larger data sets seemed to have a different behavior than the smaller, we ran a similar experiment on the satellite data set. The number of features, each consisting of a random sum of two inputs, was varied from 1 to 25, and for each 100 classifiers were combined. The results are shown in Figure 3. The results differ from those on the smaller data sets. Both the correlation and strength show a small but steady increase. The error rates show a slight decease. We conjecture that with larger and more complex data sets, the strength continues to increase longer before it plateaus out.

Our results indicate that better random forests have lower correlation between classifiers and higher strength, The randomness used in tree construction has to aim for weak dependence while maintaining reasonable strength. This conclusion has been hinted at in previous work. Dietterich [1998] has measures of dispersion of an ensemble and notes that more accurate ensembles have larger dispersion. Freund [personal communication] believes that one reason why Adaboost works so well is that at each step it tries to decouple the next classifier from the current one.

7. Conjecture: Adaboost is a Random Forest

Various classifiers can be modified to use both a training set and a set of weights on the training set. Consider the following random forest: a large collection of K different sets of non-negative sum-one weights on the training set are defined. Denote these weights by $\mathbf{w}(1),\mathbf{w}(2),\ldots\mathbf{w}(K)$. Corresponding to these weights are probabilities $p(1),p(2),\ldots p(K)$ whose sum is one. Draw from the integers 1, ..., K according to these probabilities. The outcome is Θ. If $\Theta = k$ grow the classifier $h(\mathbf{x},\Theta)$ using the training set with weights $\mathbf{w}$(k).

In its original version, Adaboost (Freud and Schapire[]) is a deterministic algorithm that selects the weights on the training set for input to the next classifier based on the misclassifications in the previous classifiers. In our experiment, random forests were produced as follows: Adaboost was run 75 times on a data set producing sets of non-negative sum-one weights $\mathbf{w}(1),\mathbf{w}(2),\ldots\mathbf{w}(50)$ (the first 25 were discarded). The probability for the kth set of weights is set proportional to $Q(\mathbf{w}_k)=\log((1-error(k))/error(k))$ where $error(k)$ is the $\mathbf{w}$(k) weighted training set error of the kth classifier. Then the forest is run 250 times.

This was repeated 100 times on a few data set, each time leaving out 10% as a test set and then averaging the test set errors. On each data set, the Adaboost error rate was very close to the forest error rate. A typical result is on the Wisconsin Breast Cancer data where Adaboost produced an average of 2.91% error and the forest produced 2.94%.

In the Adaboost algorithm, $\mathbf{w}(k+1)=\phi(\mathbf{w}(k))$ where ϕ is a function determined by the base classifier. Denote the kth classifier by $h(\mathbf{x},\mathbf{w}_k)$. The vote of the kth classifier is weighted by $Q(\mathbf{w}_k)$, so that the normalized vote for class j at $\mathbf{x}$ equals

$$\sum_k I(h(\mathbf{x},\mathbf{w}_k)=j)Q(\mathbf{w}_k)/\sum_k Q(\mathbf{w}_k). \tag{7}$$

For any function f defined on the weight space, define the operator $\mathbf{T}f(\mathbf{w})=f(\phi(\mathbf{w}))$. We conjecture that $\mathbf{T}$ is ergodic with invariant measure

$\pi(d\mathbf{w})$. Then (7) will converge to $E_{Q\pi}[I(h(\mathbf{x},\mathbf{w})=j)]$ where the distribution $Q\pi(d\mathbf{w})=Q(\mathbf{w})\pi(d\mathbf{w})/\int Q(\mathbf{v})\pi(d\mathbf{v})$. If this conjecture is true, then Adaboost is equivalent to a random forest where the weights on the training set are selected at random from the distribution $Q\pi$.

Its truth would also explains why Adaboost does not overfit--an experimental fact that has been puzzling. (There is some experimental evidence that Adaboost may overfit if run thousands of times. But I suspect that this may be due to numerical problems. See the discussion in Bauer and Kohavi[1999])

The truth of this conjecture does not solve the problem of how Adaboost selects the favorable distributions on the weight space that it does. I have experimented with other distributions on weight space and none gives generalization accuracy comparable to Adaboost. Note that the distribution of the weights will depend on the training set. In the usual random forest, the distribution of the random vectors does not depend on the training set.

5. <u>The Effects of Output Noise</u>

Dietterich [1998] showed that when a fraction of the output labels in the training set are randomly altered, the accuracy of Adaboost degenerates, while bagging and random split selection are relatively immune to the noise. Since some noise in the outputs is often present, robustness with respect to noise is a desirable property. Following Dietterich [1998] the following experiment was done which changed about one in twenty class labels.

For each data set in the experiment 10% at random is split off as a test set. Two runs are made on the remaining training set. The first run is on the training set as is. The second run is on a noisy version of the training set. The noisy version is gotten by changing a random 5% of the class labels into another class label. The alternative class label is chosen uniformly from the other labels.

This is repeated 50 times using Adaboost (deterministic version), Forest-RI and Forest-RC. The test set results are averaged over the 50 repetitions and the percent increase due to the noise computed. In both random forests, we used the number of features giving the lowest test set error in the Section 5 and 6 experiments. Because of the lengths of the runs, only the 9 smallest data sets are used. Table 3 gives the increases in error rates due to the noise.

Table 3 Increases in Error Rates Due to Noise (%)

Data Set	Adaboost	Forest-RI	Forest-RC
glass	1.6	.4	-.4
breast cancer	43.2	1.8	11.1
diabetes	6.8	1.7	2.8
sonar	15.1	-6.6	4.2
ionosphere	27.7	3.8	5.7
soybean	26.9	3.2	8.5
ecoli	7.5	7.9	7.8
votes	48.9	6.3	4.6
liver	10.3	-.2	4.8

Adaboost deteriorates markedly with 5% noise, while the Forest procedures generally show small changes. The effect on Adaboost is curiously data set dependent , with the two multiclass data sets, glass and ecoli, along with diabetes, least effected by the noise. The Adaboost algorithm iteratively weights up the points most frequently misclassified. Instances having incorrect class labels will persist in being misclassified. Then, Adaboost will mistakenly concentrate increasing weight on these noisy instances and become warped. The Forest procedures do not concentrate weight on any subset of the instances and the noise effect is smaller.

8. **Data With Many Weak Inputs**

Data sets with many weak inputs are becoming more common, i.e. in medical diagnosis, document retrieval, etc. The common characteristics is no no single input or small group of inputs can distinguish between the classes. This type of data is difficult for the usual classifiers--neural nets and trees.

To see if there is a possibility that Forest-RI methods can work, the following 10 class, 1000 binary input, was generated: (rnd is a uniform random number, selected anew each time it appears)

```
do j=1,10
do k=1,1000
p(j,k)=.2*rnd+.01
end do
end do

do j=1,10
do k=1, nint(400*rnd)      !nint=nearest integer
k=nint(1000*rnd)
p(j,k)=p(j,k)+.4*rnd
end do
```

```
end do

do n=1,N
j=nint(10*rnd)
do m=1,1000
if (rnd<p(j,m) then
x(m,n)=1
else
x(m,n)=0
end if
y(n)=j        ! y(n) is the class label of the nth example
end do
end do
```

This code generates a set of probabilities {p(j,m)} where j is the class label and m is the input number. Then the inputs for a class j example are a string of M binary variables with the mth variable having probability p(j,m) of being one.

For the training set, N=1000. A 4000 example test set was also constructed using the same {p(j,k)}. Examination of the code shows that each class has higher underlying probability at certain locations. But the total over all classes of these locations is about 2000, so there is significant overlap. Assuming one knows all of the {p(j,k)} the Bayes error rate for the particular {p(j,m)} computed in our run is 1.0%.

Since the inputs are independent of each other, the Naive Bayes classifier, which estimates the {p(j,k)} from the training data is supposedly optimal and has an error rate of 6.2%. This is not an endorsement of Naive Bayes, since it would be easy to create a dependence between the inputs which would increase the Naive Bayes error rate. We stayed with this example because the Bayes rate and the Naive Bayes error rates are easy to compute.

We started with a run of Forest-RI with F=1. It converged very slowly and by 2500 iterations, when it was stopped, had still not converged. The test set error was 10.7%. The strength was .069 and the correlation .012 with a c/s2 ratio of 2.5. Even though the strength was low, the almost zero correlation meant that we were adding small increments of accuracy as the iterations proceeded.

Clearly, what was desired was an increase in strength while keeping the correlation low. Forest-RI as run again using F=int($\log_2$M+1)=10. The results were encouraging. It converged after after 2000 iterations. The test set error is 3.0%. The strength is .22, the correlation .045 and c/s2=.91. Going with the trend, Forest-RI was run with F=25 and stopped after 2000 iterations. The test set error is 2.8%. Strength is .28, correlation .065 and c/s2 = .83.

Its interesting that forest-RI could produce error rates not far above the Bayes error rate. The individual classifiers are weak. For F=1, the average tree error rate is 80%, for F=10, it is 65% , and for F=25, it is 60%. Forests seem to have the ability to work with very weak classifiers as long as their correlation is low. A comparison using Adaboost was tried, but I can't get Adaboost to run on this data.

9. **Random Forests for Regression.**

Random forests for regression are formed by growing trees depending on a random vector Θ such that the tree predictor $h(\mathbf{x},\Theta)$ takes on numerical values. The output values are numerical and we assume that the training set is independently drawn from the distribution of the random vector $Y,\mathbf{X}$. The mean squared generalization error for any numerical predictor $h(\mathbf{x})$ is

$$E_{\mathbf{X},Y}(Y-h(\mathbf{X}))^2$$

The random forest predictor is formed by taking the average over k of the trees $\{h(\mathbf{x},\Theta_k)\}$. Similarly to the classification case, the following holds:

Theorem 9.1 *As the number of trees in the forest goes to infinity , almost surely,*

$$E_{\mathbf{X},Y}(Y-av_k h(\mathbf{X},\Theta_k))^2 \to E_{\mathbf{X},Y}(Y-E_\Theta h(\mathbf{X},\Theta))^2 \qquad (8)$$

Proof. :see Appendix I.

Denote the right hand side of (8) as PE*(forest)--the generalization error of the forest. Define the average generalization error of a tree as:

$$\text{PE*(tree)}=E_\Theta\, E_{\mathbf{X},Y}(Y-h(\mathbf{X},\Theta))^2$$

Theorem 9.2 *Assume that for all* Θ, $EY=E_{\mathbf{X}}h(\mathbf{X},\Theta)$. *Then*

$$PE^*(forest) \le \bar{\rho} PE^*(tree)$$

where $\bar{\rho}$ *is the weighted correlation between the residuals* $Y-h(\mathbf{X},\Theta)$ *and* $Y-h(\mathbf{X},\Theta')$ *where* Θ,Θ' *are independent.*

Proof.

$$\begin{aligned} PE^*(forest) &= E_{\mathbf{X},Y}[E_\Theta(Y-h(\mathbf{X},\Theta)]^2 \\ &= E_\Theta E_{\Theta'} E_{\mathbf{X},Y}(Y-h(\mathbf{X},\Theta))(Y-h(\mathbf{X},\Theta')) \qquad (9) \end{aligned}$$

The term on the right in (9) is a covariance and can be written as

$$E_\Theta E_{\Theta'}(\rho(\Theta,\Theta')sd(\Theta)sd(\Theta'))$$

where $sd(\Theta)=\sqrt{E_{\mathbf{X},Y}(Y-h(\mathbf{X},\Theta))^2}$. Define the correlation as:

$$\bar{\rho}=E_\Theta E_{\Theta'}(\rho(\Theta,\Theta')sd(\Theta)sd(\Theta'))/(E_\Theta sd(\Theta))^2.$$

Then

$$PE^*(forest)=\bar{\rho}\,(E_\Theta sd(\Theta))^2\le\bar{\rho}PE^*(tree).$$

Theorem (9.1) pinpoints the requirements for accurate regression forests--low correlation between residuals and low error trees. The forest decreases the average error of the trees employed by the factor $\bar{\rho}$. The randomization employed needs to aim at low correlation.

10 Empirical results in Regression

In regression forests we use random feature selection on top of bagging. Therefore we can use the monitoring provided by out-of-bag estimation to give estimates of PE*(forest), PE*(tree) and $\bar{\rho}$. These are derived similarly to the estimates in classification. Throughout, features formed by a random linear sum of two inputs are used. The variable is how many of these features are generated to determine the split at each node. The more features used, the lower PE*(tree) but the higher $\bar{\rho}$. In our empirical study the following data sets are used:

Table 4 Data Set Summary

Data Set	Nr. Inputs	#Training	#Test
Boston Housing	12	506	10%
Ozone	8	330	10%
Servo -2	4	167	10%
Abalone	8	4177	25%
Robot Arm -2	12	15,000	5000
Friedman#1	10	200	2000
Friedman#2 +3	4	200	2000
Friedman#3 -3	4	200	2000

Of these data sets, the Boston Housing, Abalone and Servo are available at the UCI repository. The Robot Arm data was provided by Michael Jordan. The last three data sets are synthetic. They originated in Friedman [1991] and are also described in Breiman[1998]. These are the same data sets used to compare adaptive bagging to bagging (see Breiman[1999}). except that one synthetic data set (Peak20) which was found anomalous, both by other researchers and myself, is eliminated.

The first three data sets listed are moderate in size and test set error was estimated by leaving out a random 10% of the instances, running on the remaining 90% and using the left out 10% as a test set. This was repeated 100 times and the test set errors averaged. The abalone data set is larger with 4177 instances and 8 input variables. It originally came with 25% of the instances set aside as a test set. We ran this data set leaving out a randomly selected 25% of the instances to use as a test set, repeated this 10 times and averaged

Table 4 gives the test set mean-squared error for bagging, adaptive bagging, and the random forest. These were all run using 25 features to split each node, each feature a random combination of two inputs. All runs with all data sets, combined 100 trees. In all data sets, the rule "don't split if the node size is < 5" was enforced.

Table 5 Mean-Squared Test Set Error

Data Set	Bagging	Adapt. Bag	Forest
Boston Housing	11.4	9.7	10.2
Ozone	17.8	17.8	16.3
Servo -2	24.5	25.1	24.6
Abalone	4.9	4.9	4.6
Robot Arm -2	4.7	2.8	4.2
Friedman #1	6.3	4.1	5.7
Friedman #2+3	21.5	21.5	19.6
Friedman #3 -3	24.8	24.8	21.6

An interesting difference between regression and classification is that the correlation increases quite slowly as the number of features used increases. The major effect is the decrease in PE*(tree). Therefore, a relatively large number of features are required to reduce PE*(tree) and get near optimal test set error.

The results shown in Table 5 are mixed. Random forest-random features are always better than bagging. In data sets for which adaptive bagging gives sharp decreases in error, the decreases produced by forests are not as pronounced. In data sets in which adaptive bagging gives no improvements over bagging, forests produce improvements. I have gotten better results for some of the data sets using random combinations of 3 inputs, by shutting off the bagging or by using a large number of features. For instance, for the robot arm data set, using 100 features drops the error to 3.9.

For the same number of inputs combined, over a wide range, the error does not change much with the number of features. In the number used is too small, PE*(tree) becomes too large and the error goes up. If the number used is too large, the correlation goes up and the error decreases. The in-between range is usually large. In this range, as the number of features goes up, the correlation decreases, but PE*(tree) compensates by decreasing.

Table 6 gives the test set errors, the out-of-bag error estimates, and the OB estimates for PE*(tree) and the correlation.

Table 6. Error and OB Estimates

Data Set	Test Error	OB Error	PE*(tree)	Cor.
Boston Housing	10.2	11.6	26.3	.45
Ozone	16.3	17.6	32.5	.55
Servo -2 ,	24.6	27.9	56.4	.56
Abalone	4.6	4.6	8.3	.56
Robot Arm -2	4.2	3.7	9.1	.41
Friedman #1	5.7	6.3	15.3	.41
Friedman #2+3	19.6	20.4	40.7	.51
Friedman #3 -3	21.6	22.9	48.3	.49

As expected, the OB error estimates are consistently high . It is low in robot arm but I believe that this is an artifact of its separation into a single training set and single test set.

11 **Remarks and Conclusions**

Random forests are an effective tool in prediction. Because of the Law of Large Numbers they do not overfit. Injecting the right kind of randomness makes them accurate classifiers and regressors. Furthermore, the framework in terms of strength of the individual predictors and their correlations gives insight into the ability of the forest to predict. Using out-of-bag estimation makes concrete the otherwise theoretical values of strength and correlation.

For a while, the conventional thinking was that forests could not compete with arcing type algorithms in terms of accuracy. Our results dispel this belief, but lead to interesting questions. Boosting and arcing algorithms have the ability to reduce bias as well as variance (Schapire et.al [1998]). The adaptive bagging algorithm in regression (Breiman [1999]) was designed to reduce bias and operates effectively in classification as well as in regression. But , like arcing, it also changes the training set as it progresses.

Forests give results competitive with boosting and adaptive bagging, yet do not progressively change the training set. Their accuracy indicates that they act to reduce bias. The mechanism for this is not obvious. Random forests may also be viewed as a Bayesian procedure. Although I doubt that this is a fruitful line of exploration, if it could explain the bias reduction I might become more of a Bayesian.

Random inputs and random features produce good results in classification--less so in regression. The only types of randomness used in this study is bagging and random features. It may well be that other types of injected randomness give better results. The idea of injecting independent randomness into each base predictor is not limited to trees. Many others could be used.

An almost obvious question is whether gains in accuracy can be gotten by combining random features with boosting. For the larger data sets, it seems that significantly lower error rates are possible. On some runs, we got errors as low as 5.1% on the zip-code data, 2.2% on the letters data and 7.9% on the satellite data. The improvement was less on the smaller data sets. More work is needed on this--but it does suggest that different injections of randomness can produce better results.

References

Amit, Y. and Geman, D. [1997] Shape quantization and recognition with randomized trees, Neural Computation 9,1545-1588

Bauer, E. and Kohavi, R. [1999] An Empirical Comparison of Voting Classification Algorithms, Machine Learning, 36, No. 1/2, 105-139

Breiman, L. [1999] Using adaptive bagging to debias regressions, Technical Report 547, Statistics Dept. UCB

Breiman, L. [1998a], Arcing Classifiers, (discussion paper) Annals of Statistics, 26, 801-824

Breiman. L.[1998b] Randomizing Outputs To Increase Prediction Accuracy. Technical Report 518, May 1, 1998, Statistics Department, UCB (in press Machine Learning)

Breiman, L. [1996a], Bagging Predictors, Machine Learning, 26, No. 2, 123-140

Breiman, L. (1996b) Out-of-bag estimation, ftp.stat.berkeley.edu/pub/users/breiman/OOBestimation.ps

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

Schapire, R. ,Freund Y. Bartlett P. and Lee W. [1998] Boosting the margin: A new explanation for the effectiveness of voting methods.

The Annals of Statistics, 26(5):1651-1686, 1998.
Tibshirani, R. [1996] Bias, Variance, and Prediction Error for Classification Rules, Technical Report, Statistics Department, University of Toronto
Wolpert, D.H. and Macready, W.G. [1997] An Efficient Method to Estimate Bagging's Generalization Error (in press, Machine Learning)

Appendix I Almost Sure Convergence

Proof of Theorem 1.2

It suffices to show that there is a set of probability zero C on the sequence space $\Theta_1, \Theta_{2, \ldots}$ such that outside of C, for all $\mathbf{x}$,

$$\frac{1}{N}\sum_{n=1}^{N} I(h(\Theta_n,\mathbf{x})=j) \rightarrow P_\Theta(h(\Theta,\mathbf{x})=j).$$

For a fixed training set and fixed Θ the set of all $\mathbf{x}$ such that $h(\Theta,\mathbf{x})=j$ is a union of hyper-rectangles. For all $h(\Theta,\mathbf{x})$ there is only a finite number K of such unions of hyper-rectangles, denoted by $S_1, \ldots S_K$. Define $\varphi(\Theta)=k$ if $\{\mathbf{x}: h(\Theta,\mathbf{x})=j\}=S_k$. Let N_k be the number of times that $\varphi(\Theta_n)=k$ in the first N trials. Then

$$\frac{1}{N}\sum_{n=1}^{N} I(h(\Theta_n,\mathbf{x})=j) = \frac{1}{N}\sum_{k} N_k I(\mathbf{x}\in S_k)$$

By the Law of Large Numbers,

$$N_k = \frac{1}{N}\sum_{n=1}^{N} I(\varphi(\Theta_n)=k)$$

converges a.s. to $P_\Theta(\varphi(\Theta)=k)$. Taking unions of all the sets on which convergence does not occur for some value of k gives a set C of zero probability such that outside of C,

$$\frac{1}{N}\sum_{n=1}^{N} I(h(\Theta_n,\mathbf{x})=j) \rightarrow \sum_{k} P_\Theta(\varphi(\Theta)=k) I(\mathbf{x}\in S_k).$$

The right hand side is $P_\Theta(h(\Theta,\mathbf{x})=j)$.

Proof of Theorem 9.1 There are a finite set of hyper-rectangles $R_1, \dots R_K$ such that if $\bar{y}_k$ is the average of the training sets y-values for all training input vectors in R_k then $h(\Theta,\mathbf{x})$ has one of the values $I(\mathbf{x}\in S_k)\bar{y}_k$. The rest of the proof parallels that of Theorem 1.2

Appendix II Out-of Bag Estimates for Strength and Correlation

At the end of a combination run, let

$$Q(\mathbf{x},j)=\sum_k I(h(\mathbf{x},\Theta_k)=j;(y,\mathbf{x})\notin T_{k,B})/\sum_k I((y,\mathbf{x})\notin T_{k,B}).$$

Thus, $Q(\mathbf{x},j)$ is the out-of-bag proportion of votes cast at $\mathbf{x}$ for class j, and is an estimate for $P_\Theta(h(\mathbf{x},\Theta)=j)$. From Definition 2.1 the strength is the expectation of

$$P_\Theta(h(\mathbf{x},\Theta)=y)-\max_{j\neq\mathbf{y}} P_\Theta(h(\mathbf{x},\Theta)=j).$$

Substituting $Q(\mathbf{x},j)$, $Q(\mathbf{x},y)$ for $P_\Theta(h(\mathbf{x},\Theta)=j)$, $P_\Theta(h(\mathbf{x},\Theta)=y)$ in this latter expression and taking the average over the training set gives the strength estimate.

From equation (6),

$$\bar{\rho}=\mathrm{var}(mr)/(E_\Theta sd(\Theta))^2.$$

The variance of mr is

$$E_{\mathbf{X},Y}[P_\Theta(h(\mathbf{x},\Theta)=y)-\max_{j\neq\mathbf{y}} P_\Theta(h(\mathbf{x},\Theta)=j)]^2-s^2 \qquad \text{(A1)}$$

where s is the strength. Replacing the first term in (A1) by the average over the training set of

$$(Q(\mathbf{x},y)-\max_{j\neq\mathbf{y}} Q(\mathbf{x},j))^2$$

and s by the out-of-bag estimate of s gives the estimate of var(mr). The standard deviation is given by

$$sd(\theta) = [p_1 + p_2 + (p_1 - p_2)^2]^{1/2} \qquad \text{(A2)}$$

where

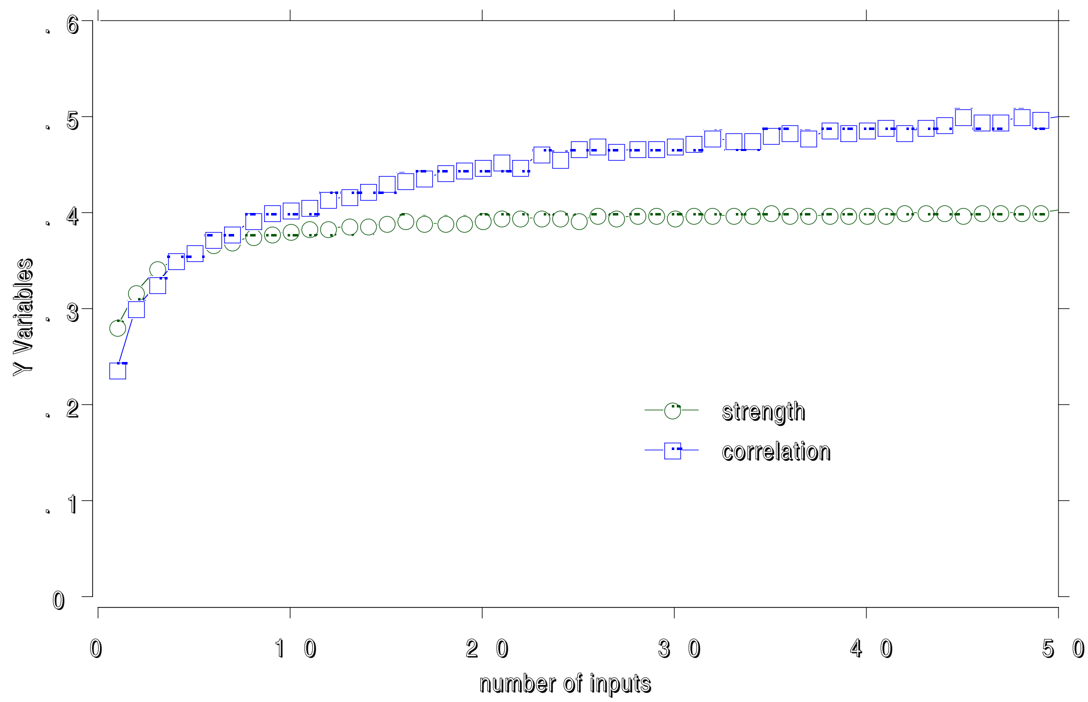
$$p_1 = E_{\mathbf{X},Y}(h(\mathbf{X},\Theta) = Y)$$
$$p_2 = E_{\mathbf{X},Y}(h(\mathbf{X},\Theta) = \hat{j}(\mathbf{X},Y)).$$

After the kth classifier is constructed, $Q(\mathbf{x},j)$ is computed, and used to compute $\hat{j}(\mathbf{x},y)$ for every example in the training set. Then, let p_1 be the average over all (y,**x**) in the training set but not in the kth bagged training set of $I(h(\mathbf{x},\Theta_k)=y)$. Then p_2 is the similar average of $I(h(\mathbf{x},\Theta_k)=\hat{j}(\mathbf{x},y))$. Substitute these estimates into (A2) to get an estimate of $sd(\Theta_k)$. Average the $sd(\Theta_k)$ over all k to get the final estimate of $sd(\Theta)$.

FIGURE 1

EFFECT OF NUMBER OF INPUTS ON SONAR DATA

CORRELATION AND STRENGTH



TEST SET AND OB ERROR

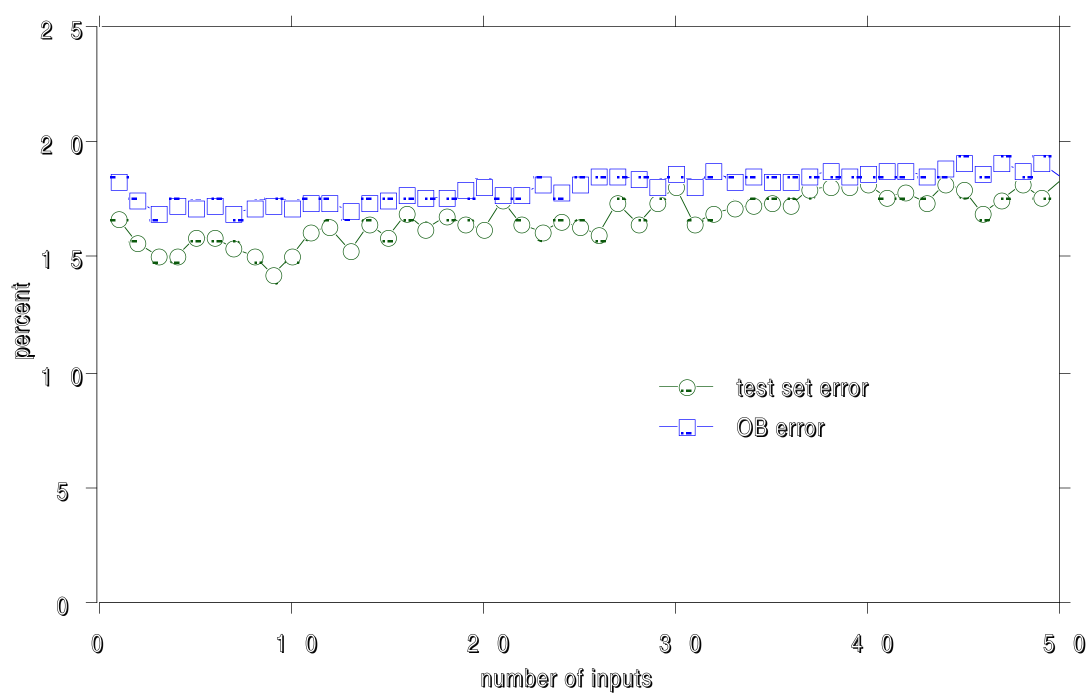
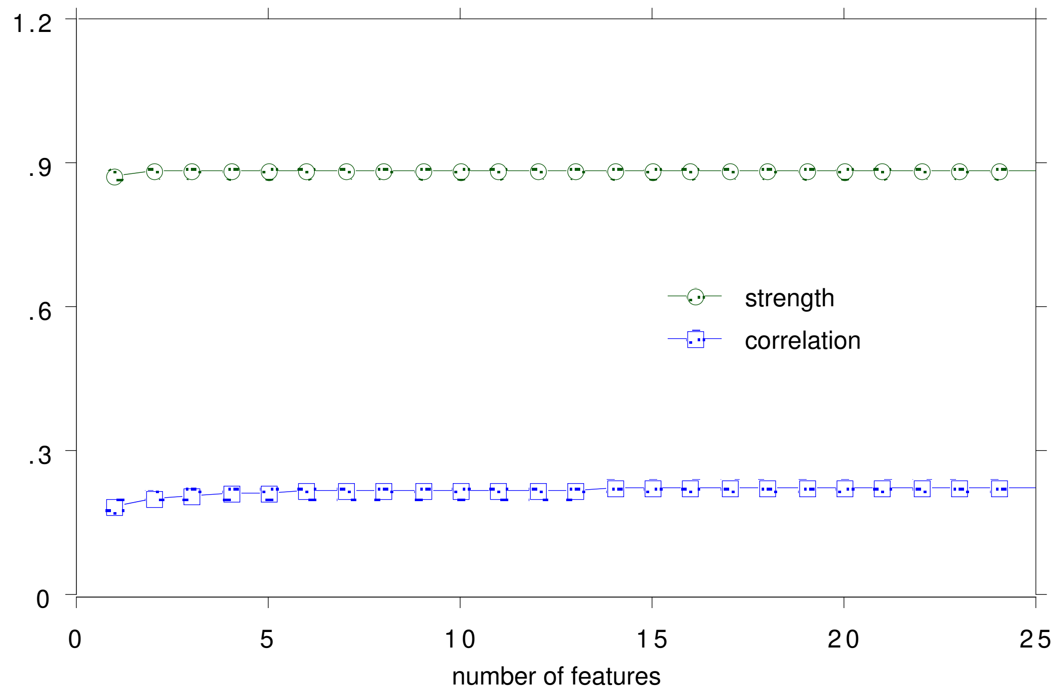


FIGURE 2

EFFECT OF THE NUMBER OF FEATURES ON THE BREAST DATA SET

CORRELATION AND STRENGTH



TEST SET AND OB ERROR

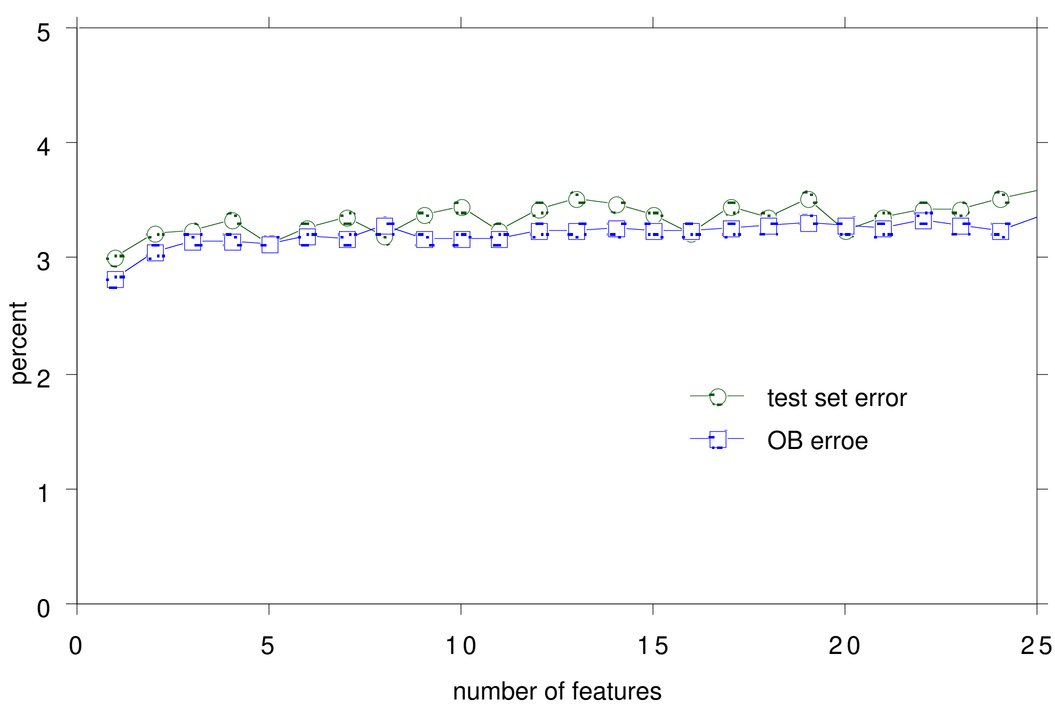
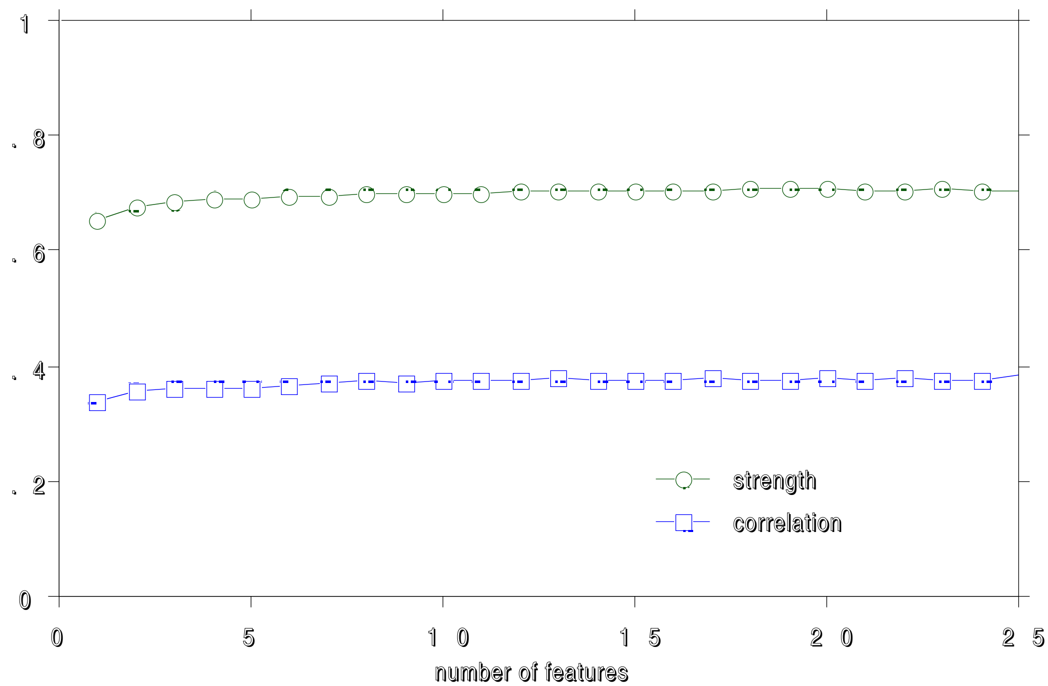


FIGURE 3

EFFECT OF NUMBER OF FEATURES ON SATELLITE DATA

CORRELATION AND STRENGTH



TEST SET AND OB ERROR