An Introduction to Ensemble Methods for Data Analysis

Richard A. Berk
Department of Statistics
UCLA

July 25, 2004

Abstract

This paper provides an introduction to ensemble statistical procedures as a special case of algorithmic methods. The discussion beings with classification and regression trees (CART) as a didactic device to introduce many of the key issues. Following the material on CART is a consideration of cross-validation, bagging, random forests and boosting. Major points are illustrated with analyses of real data.

1 Introduction

There are a growing number of new statistical procedures Leo Breiman (2001b) has called "algorithmic." Coming from work primarily in statistics, applied mathematics, and computer science, these techniques are sometimes linked to "data mining," "machine learning," and "statistical learning."

With algorithmic methods, there is no statistical model in the usual sense; no effort to made to represent how the data were generated. And no apologies are offered for the absence of a model. There is a practical data analysis problem to solve that is attacked directly with procedures designed specifically for that purpose. If, for example, the goal is to determine which prison inmates are likely to engage in some form of serious misconduct while in prison (Berk and Baek, 2003), there is a classification problem. Should the goal be to minimize some function of classification errors, procedures are applied with that minimization talk paramount. There is no need to represent

how the data were generated if it is possible to accurately classify inmates by other means.

Algorithmic methods have been applied to a wide variety of data analysis problems, particularly in the physical and biomedical sciences (Sutton and Barto, 1999; Witten and Frank, 2000; Christianini and Shawne-Taylor, 2000; Breiman, 2001b, Hastie et al., 2001): predicting next-day ozone levels, finding fraudulent credit card telephone calls among many millions of transactions, and determining the predictors of survival for hepatitus patients. There is growing evidence that a number of algorithmic methods perform better than conventional statistical procedures for the tasks algorithmic methods are designed to undertake (Breiman, 2001a; Hastie et al., 2001). In addition, they raise important issues about the statistical modeling and the nature of randomness more generally.

Among the great variety of algorithmic approaches, there is a group that depends on combining the fitted values from a number of fitting attempts; fitted values are said to be "combined" or "bundled" (Hothorn, 2003). For example, one might combine the fitted values from several regression analyses that differ in how nuisance parameters are handled.

The term "ensemble methods" is commonly reserved for bundled fits produced by a stochastic algorithm, the output of which is some combination of a large number of passes through the data. Such methods are loosely related to iterative procedures on the one hand and to bootstrap procedures on the other. An example is the average of a large number of kernel smoothes of a given variable, each based on a bootstrap sample from the same data set. The idea is that a "weak" procedure can be strengthened if given a chance to operate "by committee." Ensemble methods often perform extremely well and in many cases, can be shown to have desirable statistical properties (Breiman, 2001a; 2001c).

This paper provides a review of ensemble methods likely to be especially useful for the analysis of social science data. Illustrations and software are considered. The purpose of the discussion is to introduce ensemble methods, although some deeper conceptual issues will be briefly considered. For didactic purposes, techniques for categorical response variables will be emphasized. The step to equal interval response variables is then easily made.

2 Classification and Regression Trees as an Algorithmic Method

Classification and Regression Trees (CART) has been around for about 20 years (Breiman et al., 1984) and is implemented in any a number of popular statistical packages. It may, therefore, be familiar to many social scientists and provide a good transition from traditional modeling to algorithmic approaches. CART is also a building block for most of the ensemble procedures we will consider.

The goal is to use a set of predictors to subset the data such that within each subset, the values of the response variable are as similar as possible. This is accomplished by subsetting the space defined by the data one partitioning at a time in recursive fashion; once a partition is constructed, it is not reconsidered when later partitions are defined. CART constructs partitions with a series of straight-line boundaries, as in Figure 1, perpendicular to axis of the predictor being used.

Figure 1 represents a simple classification problem. There is a binary outcome coded "A" or "B," and predictors x and z. The red vertical line produces the first partition. The green horizontal line produces the second partition. The yellow horizontal line produces the third partition.

The data are first divided vertically and then for each of these two partitions, the data are further divided horizontally in an upper and lower part. In this simple illustration, the upper left partition and the lower right partition are fully homogeneous. There remains considerable heterogeneity in the other two partitions and in principle, their partitioning could continue.

Figure 1 reveals that cases high on z and low on x are always "B." Cases low on z and high on x are always "A." In a real analysis, the terms "high" and "low" would be defined by where the boundaries cross the x and z axes.

CART output is often shown as an inverted tree. Figure 2 is a simple illustration. The full data set is contained in the root node. The final partitions are subsets of the data contained in the terminal nodes. The internal nodes contain subsets of data for intermediate steps.

CART results depend on solving in order six technical problems.

1. A criterion for the splitting is required.

2. A criterion for the variable selection is required.

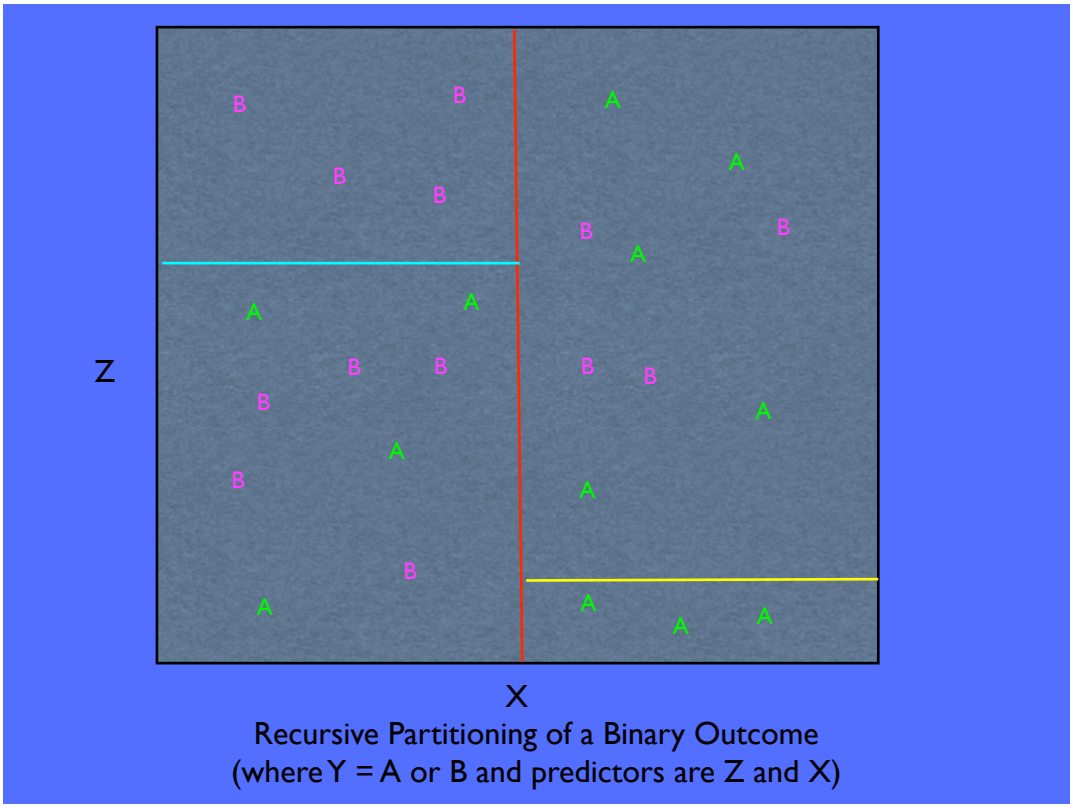

Figure 1: Recursive Partitioning Logic in CART

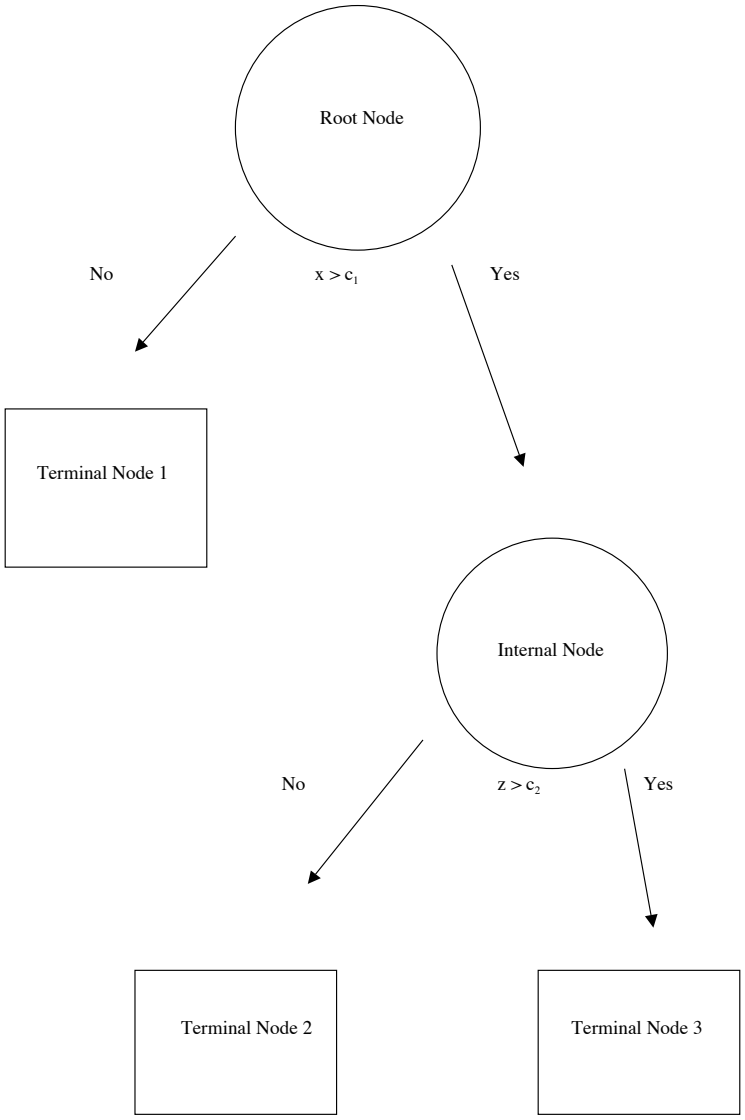

Figure 2: CART Tree Structure

3. A way is needed to influence the size of the tree so that only useful terminal nodes are constructed.

4. A way is needed to consider how "good" the tree is.

5. A way is needed to protect against overfitting.

6. A way is needed to interpret and communicate the results.

The specific solutions to these problems depend on whether the response is categorical or quantitative: whether a classification tree or a regression tree respectively is desired. In this paper, we continue with the the emphasis on classification problems.

2.1 Splitting a Node

For a quantitative predictor, suppose there are m distinct values. Then there are $m - 1$ splits that maintain the existing ordering of values. The same is true for ordinal predictors. For example, if there are 40 distinct high school GPA scores possible, there are 39 possible splits that maintain the existing order.

Because order does not matter for categorical predictors, a categorical variable with k categories has $(2^{k-1} - 1)$ possible splits. For example, if there are 5 ethnic categories, there are 15 allowable splits. Hence, the computational burdens are often much heavier for categorical variables.

Beginning the root node, CART considers all possible splits on all predictor variables and picks the best split over all of these variables. That is, the single variable is chosen such that the "best split" is better than the best split of any other predictor. The same basic exercise is undertaken for all subsequent nodes until a set of terminal nodes is reached.

There are three different criteria for determining "best" that are commonly considered. For ease of exposition, assume the response is a binary variable coded 1 or 0 (and there can be additional complications when there are more than 2 categories).

The "impurity" of node τ is defined as a non-negative function of the probability $p(y = 1|\tau)$. Ideally, if τ is a terminal node, it should be composed of cases that are all equal to 1 or all equal to 0. Then, its impurity would be as small as possible. In contrast, if half the cases are equal to 1 and half the

cases are equal to 0, τ is the most impure it can be; a given case is as likely to be a 1 as it is a 0.

Suppose we formally define impurity as

$$i(\tau) = \phi[p(y = 1|\tau)], \tag{1}$$

where $\phi \geq 0$, $\phi(p) = \phi(1 - p)$, and $\phi(0) = \phi(1) < \phi(p)$; impurity is non-negative, and symmetrical with a minimum when τ contains all 0's or all 1's.

There are three popular options for ϕ: in order below, Bayes error, the entropy function, and the Gini Index.

$$\phi(p) = min(p, 1 - p). \tag{2}$$

$$\phi(p) = -plog(p) - (1 - p)log(1 - p). \tag{3}$$

$$\phi(p) = p(1 - p). \tag{4}$$

All three equations are concave with minimums at $p = 0$ and $p = 1$ and a maximum at $p = .5$. Entropy and the Gini index are the most commonly used. They generally give very similar results except when there are more than two response categories. Then, Breiman et al. (1984: 111) favor the Gini index because the Gini index favors splits with one relatively homogeneous node having relatively few cases. The other nodes are then relatively heterogeneous and have relatively more cases. For most data analyses, this result is desirable. Entropy tends to create splits that are about equal in size and homogeneity.

To help fix these ideas, consider a simple illustration. Assume for the moment that the data are a random sample from some well defined population. For any internal node, we can produce a left daughter node, τ_L and a right daughter node τ_R. Imagine now a table (Table 1) in which that split is represented.

Let $y = 1$ if there is a success and 0 otherwise.[1] One can then estimate $p(y = 1|\tau_L)$ by $n_{12}/n_{1.}$. Similarly, $p(y = 1|\tau_R)$ can be estimated by $n_{22}/n_{2.}$.

[1]What follows is drawn heavily from H. Zhang and B. Singer, *Recursive Partitioning in the Health Sciences*, Springer-Verlag, 1999, chapters 2 and 4.

	Failure	Success	Total
Left Node: $x \leq c$	n_{11}	n_{12}	$n_{1.}$
Right Node: $x > c$	n_{21}	n_{22}	$n_{2.}$
	$n_{.1}$	$n_{.2}$	$n_{..}$

Table 1: Input to Split

"Entropy impurity" for the left daughter is

$$i(\tau_L) = -\frac{n_{11}}{n_{1.}} log(\frac{n_{11}}{n_{1.}}) - \frac{n_{12}}{n_{1.}} log(\frac{n_{12}}{n_{1.}}). \tag{5}$$

"Entropy impurity" for the right daughter is

$$i(\tau_R) = -\frac{n_{21}}{n_{2.}} log(\frac{n_{21}}{n_{2.}}) - \frac{n_{22}}{n_{2.}} log(\frac{n_{22}}{n_{2.}}). \tag{6}$$

Suppose for the left daughter there are 300 observations with 100 successes and 200 failures. Then, the impurity is $-.67(-.40)-.33(-1.11) = .27+.37 = .64$. For the right daughter there are 100 observations with 45 successes and 55 failures. Then, the impurity is $-.55(-.60) - .45(-.80) = .33 + .36 = .69$. Note that the largest impurity one could get is for .5 for successes and .5 for failures. That value is .693. For proportions equal to 1 or 0, the value of entropy impurity is 0. So, the minimum value is 0, and the maximum is a little more than .69. The closer one gets to 50-50, where the impurity is the greatest, the closer one gets to .693.

After all possible splits across all possible variables are evaluated, the question becomes which split to use. The improvement resulting from a split is the entropy impurity of the parent node minus the left and right daughter entropy impurities. If this is a big number, entropy impurity is reduced substantially.

More formally, the goodness of the split, s, is measured by

$$\Delta I(s, \tau) = i(\tau) - p(\tau_L) i(\tau_L) - p(\tau_R) i(\tau_R), \tag{7}$$

where $i(\tau)$ is the the value of the parent entropy impurity, $p(\tau_R)$ is the probability of being in the right daughter, $p(\tau_L)$ is the probability of being in the left daughter, and the rest is defined as before. The two probabilities can be estimated from a table such as Table 1; they are simply the marginal

proportions $n_{1.}/n_{..}$ and $n_{2.}/n_{..}$.[2] CART computes $\Delta I(s,\tau)$ for all splits on each variable and chooses the variable and split with the largest value. The same operation is undertaken for all subsequent nodes.

In principle, the CART algorithm can keep partitioning until there is only one case in each node. Such a tree is called "saturated." However, well before a tree is saturated, there will usually be far too many terminal nodes to interpret, and the number of cases in each will be quite small. One option, is to instruct CART not to construct terminal nodes with samples smaller than some specified value. A second option will be considered shortly.

Figure 3 shows a classification tree for an analysis of which prison inmates engage in some form of reportable misconduct while in prison. A minimum node sample size of 100 was imposed for reasons to be explained shortly. The variables in Figure 3, selected by CART from a larger set of 12 predictors, are defined as follows.

1. gang.f: involved in gang activity with a = no and b = yes.
2. term: sentence length in years.
3. agerec: age at arrival at the prison Reception Center in years with a = 16-20, b = 21-26, c = 27-35, and d = 36 or older.

At each split, the subset identified by the splitting criterion is moved to the left daughter node. Terminal notes are labeled "N" if the majority do not engage in misconduct and "Y" if the majority do. The numbers below each terminal node show the distribution of no misconduct to misconduct. Thus, for the far right terminal node, there are 226 cases with no misconduct and 270 cases with misconduct. All of the cases in this terminal node are gang members serving terms of 2.5 years or more and who arrived at the prison reception center under the age of 27. A examination of each of the other terminal nodes will reveal that the risks of misconduct are the highest by far for this group.

These findings would be no surprise to criminologists. However, a conventional analysis using logistic regression might well miss them. The far right terminal node would be a triple interaction effect in logistic regression, and the node just to its left would as well. The far left terminal node is the

[2] $\Delta I(s,\tau)$ is effectively the reduction in the deviance and thus, there is a clear link to the generalized linear model.

only main effect, while the terminal node immediately to its right is a double interaction effect. It is extremely unlikely that a researcher would have specified such a model in logistic regression and when the more likely all-main-effect model was applied to these data, the fit was dramatically worse and led to somewhat different conclusions.[3]

2.2 Terminal Nodes and Pruning

A second way (besides restricting node size) to constrain the size of the tree is called pruning. The pruning process collapses terminal nodes that do not reduce heterogeneity sufficiently for the extra complexity added. Pruning does not figure significantly in ensemble methods, but is an important feature of CART and raises some key conceptual issues that ensemble methods address in other ways.

For tree T define the overall "quality" of a tree as

$$R(T) = \sum_{\tau \in \tilde{T}} p(\tau) r(\tau), \tag{8}$$

where $\tilde{T}$ is the set of terminal nodes of T, $p(\tau)$ is the probability that a case will fall in terminal node τ, and $r(\tau)$ is a certain quality of that node, which has some parallels to the error sum of squares in linear regression. More details will be provided shortly. The purpose of pruning is to select the best subtree T^* starting with the saturated tree T_0 so that $R(T)$ is minimized. To get that job done one needs $r(\tau)$. That, in turn will depend on two factors: classification error and the complexity of the tree. And there will be a tradeoff between the two.

2.3 Classification Errors and Costs

One might think that $r(\tau)$ would be a measure of impurity for node τ. But it is usually far more instructive in practice to use a measure that is grounded in the data analysis task we want to accomplish. For this, some function of

[3]One can obtain the same fit statistics as provided in CART by specifying a logistic regression as described immediately above (one main effect, one double interaction effect, and two triple interaction effects). One can then directly compare the CART results with other model specifications

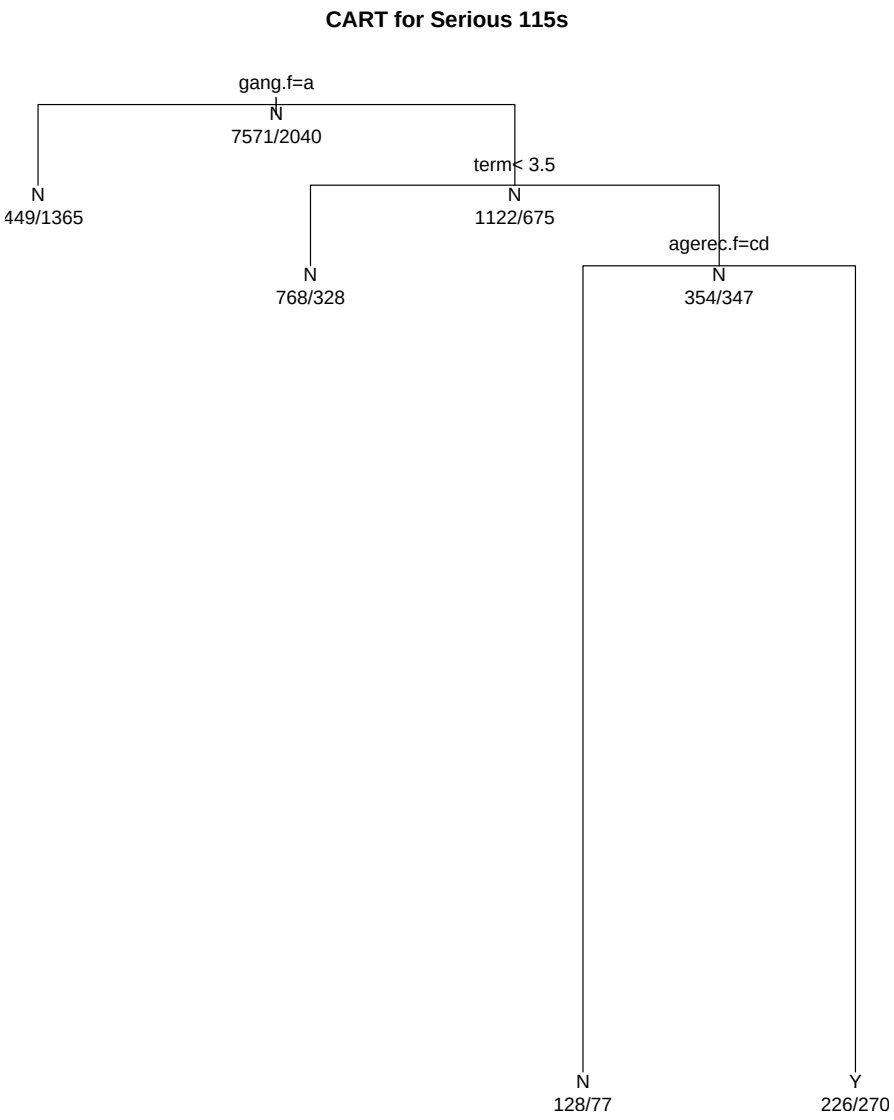

Figure 3: Recursive Partitioning of CDC Data

classification errors is often exploited.[4]

Suppose the problem is whether a beach is classified as closed because of pollution. Closed is coded 1 and not closed is coded 0. We then examine each terminal node and "predict" all of the cases to be 1's if the majority of cases in that node are 1. This is an application of a "majority vote" rule. If the majority of cases in that node are 0, we "predict" that all of the cases will be 0's. This too is an application of a majority vote rule. For example, if there are 17 cases in a given terminal node and 10 are 1's, we predict 1 (closed) for all of the cases in that node. We then necessarily have 7 misclassifications. If 5 cases in that node are 1's, we "predict" that all of the cases are 0, and then there are 5 misclassifications.

There are two kind of classification errors: false positives and false negatives. Here, a false positive here would occur if the model classifies a given beach as closed when in fact it is not. A false negative would occur if the model classifies a given beach as open when it is not. Clearly, one would like both false positives and false negatives to be as small as possible.

However, the consequences of the two kinds of errors may not be the same. Are the social and health costs the same for telling the public that a beach is closed when it is not as for telling the public that a beach is open when it is not? Some might argue that the costs of falsely claiming a beach is open are higher because people might travel to that beach and then be turned away and/or not know the beach is polluted and swim there. The general point is that one needs to consider building in the costs of both kinds of errors. (For an example, see Berk et al., 2004a)

In practice, all one needs is the relative costs of the two kinds of errors, such as 5 to 1. But with that in hand, one can for each node determine the costs of predicting either $y = 1$ or $y = 0$.

Suppose that there are 7 cases in a terminal node, 3 coded as 1 (the beach is closed) and 4 are coded as 0 (the beach is open). For now, the costs of both false positives and false negatives are 1. Then, the "expected cost" of a false positive is $1 \times 4/7 = .57$ while the expected cost of a false negative is $1 \times 3/7 = .43$. (4/7 is the probability — assuming random sampling — of a false positive and 3/7 is the probability of a false negative.) For these costs (and any costs that are the same for false positives and false negatives), we

[4]We will see later that boosting goes directly for a minimization of classification error. There is no path through a measure of impurity. One cost is that links to more conventional statistical modeling are lost and apparently, a number of desirable properties of estimators.

are better off predicting for all of the cases in this node that the beach is open; the expected cost (of false negatives) is lower: .43 compared to .57. The value of .43 is often called the within-node misclassification cost or the conditional (on τ) misclassification cost.

But suppose now that the costs of a false negative are 3 times the cost of a false positive. So, the costs would be 1 unit for a false positive and 3 units for a false negative. Then, the expected cost of a false positive is $1 \times 4/7 = .57$, while the expected cost of a false negative is $3 \times 3/7 = 1.29$. Clearly, it is better for this node to predict that the beach will be closed: .57 compared to 1.29.

But there is more. One does not want just the within-node costs, but the costs aggregated over all of the terminal nodes. So one needs to multiply the conditional (within node) misclassification cost for each terminal node by the probability of a case falling in that node. For example, if there are 500 cases in the root node and 50 cases in terminal node τ, that probability is .10. If the expected cost of that node is, say, 1.29, the unconditional cost is $.10 \times 1.29 = .129$. To arrive at the costs over all terminal nodes, and hence for that tree, one simply adds all of the unconditional costs for each terminal node.

Notice that all of the data have been used to build the tree, and then these same data are "dropped " down that tree to get the overall costs of that tree. The same data used to build the model are used to evaluate it. And given all of data manipulations, overfitting is a real problem. As a result, expected costs are usually too low. The notation $R^s(\tau)$ is to make clear that the costs are probably too optimistic. The "s" stands for "resubstitution." Measures of model quality are derived when the data used to build the model are "resubstituted" back into the model. Overfitting will be addressed at length shortly.

2.4 Prior Probabilities

In some implementations of recursive partitioning, one can specify the prior probability of a success and a failure.[5] This is especially handy as another

[5]The prior probability is one's believe about what the marginal distribution of the response is. For example, based on other data or research, one might believe that the probability of some form of reportable misconduct in prison is .25, and the probability of no reportable misconduct in prison is .75.

way to specify costs when the procedure being used, or the software, does not allow for specification of the costs directly.

The basic logic is this: if the cost of a false positive is 5 times greater than the cost of a false negative, it is much the same as saying that for every false negative there are 5 false positives. Costs translate into numbers of cases. The same kind of logic applies when using the prior probabilities. While the data may indicate that there are 3 successes for each failure, if a false success is twice as important as a false failure, it is as if there were 6 successes for each failure.[6] In short, one can get to the same place by using prior probabilities or costs. For the formal details see Breiman et al., (1984: Section 4.4).

2.5 Cost Complexity

Just as in any modeling, simplicity is desirable for ease of interpretation and the robustness of the results. Moreover, with added complexity comes the risk of more serious overfitting.

To take complexity into account, one can define an objective function for recursive partitioning that includes a penalty for complexity based not on the number of parameters (as in conventional regression), but the number of terminal nodes.

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|, \tag{9}$$

where $\alpha \geq 0$ is the complexity parameter, and $|\tilde{T}|$ is the number of terminal nodes in tree T. Thus, the second term on the right hand side provides the penalty. The larger the value of α, the harsher the penalty for complexity. When $\alpha = 0$, there is no penalty and a saturated tree results. But how does one pick α?

Breiman et al. (1984, Section 3.3) prove that for any value of the complexity parameter α, there is a unique smallest subtree of T the minimizes cost-complexity. Thus, there cannot be two subtrees of the same size with the same cost complexity. Given α, there is a unique solution.

In many CART procedures, there are ways the software can select a reasonable value for α or for parameters that play the same role (Zhang and Singer, 1999, section 4.2.3). Still, in some cases, one or more of the

[6]Keep in mind that there are no costs for true positives and true negatives. They play no role in the calculations for costs.

sample sizes for terminal nodes may be too small. The results, then, can be unstable. One option is to specify through trial and error a value of α for pruning that leads to terminal nodes each with a sufficient number of cases. Alternatively, it is often possible to set the minimum sample size that is acceptable as another tuning parameter.

2.6 Overfitting and Cross-Validation

Any attempt to summarize patterns in a data set risks overfitting. Broadly stated, all fitting procedures adapt to the data on hand so that if the fit is applied to a new sample from the same population, it will not fit as well.[7] Hence, generalization can be somewhat risky. And to the degree that a fitting procedure is highly "flexible," overfitting is exacerbated. Hastie et al., (2001: 200-203) show in a slightly different context that the unjustified "optimism increases linearly with the number of inputs or basis functions ..., but decreases as the training sample size increases."[8]

Hastie and his colleagues define a basis function as follows (Hastie et al., 2001: 115-116). If there are p predictor variables contained in X, let $h_m(X) : R^p \rightarrow R$ be the mth transformation of X, $m = 1, \ldots, M$. Then the linear basis expansion of X is

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X), \tag{10}$$

where β_m is the weight given to the mth term. Note that once the basis functions h_m are defined, the model is linear in these transformations of X. For example, a fourth order polynomial would have four basis function of X (here a single variable): X, X^2, X^3, X^4. These basis function in X could be entered into a conventional linear regression model.[9]

The basis function formulation can be applied to CART at three points in the fitting process. First, for any given predictor being examined for its optimal split, overfitting will increase with the number of splits possible. In effect, a greater number of basis functions are being screened (where a given split is a basis function).

[7] If the new data are from a different population, the fitting is likely to be even worse.

[8] A "training sample" is the data used to build the model."

[9] See Hastie et al. (2001, section 5.4.1) for a related discussion of "effective degrees of freedom."

Second, for each split, CART includes as X all predictors as inputs. An optimal spilt is chosen over all possible splits of all possible predictors. This defines the optimal basis function for that stage. Hence within each stage, overfitting increases as the number of candidate predictors increases.

Then for each new stage, a new optimal basis function is chosen and applied. Consequently, overfitting increases with the number of stages, which for CART means the number of optimal basis functions, typically represented by the number of nodes in the tree.

Once a node is defined, it is unchanged by splits farther down the tree. New basis functions are just introduced along with the old ones. Therefore, CART is forward stagewise additive model that can produce overfitting within predictors, within stages and over stages.

The overfitting can be be misleading in at least two ways. First, measures meant to reflect how well the model fits the data are likely to be too optimistic. Thus, for example, the number of classification errors may be too small. Second, the model itself may have a structure that will not generalize well. For example, one or more predictors may be included in a tree that really do not belong.

Ideally, there would be two random samples from the same population. One would be a training data set and one would be a testing data set. A tree would be built using the training data set and some measure of fit obtained. A simple measure might be the fraction of cases that is classified correctly. A more complicated measure might takes costs into account. Then with the tree fixed, cases from the testing data set would be "dropped down" the tree and the fit computed again. It is almost certain that the fit would degrade, and how much is a measure of overfitting. The fit measure from the test data set is a better indicator of how accurate the classification process really is.

Often there there is only be a single data set. An alternative strategy is to split the data up into several randomly chosen, non-overlapping parts. Ten such subsets are common. The tree is built on nine of the splits and evaluated with the remaining one. So, if there are 1000 observations, one would build the tree on 900 randomly selected observations and evaluate the tree using the other 100 observations. This would be done ten times, one for each non-overlapping split, and for each a measure of fit computed. The proper measure of fit is the average over the ten splits. The procedure is called 10-fold cross-validation and is routinely available in many CART programs. Extensions on this basic idea using bootstrap samples are also available (Efron and Tibsharini, 1993, Chapter 17).

Unfortunately, cross-validiation neglects that the model itself is potentially misleading. A method is needed to correct the model for overfitting, not just its goodness-of-fit measure. For this we turn to "ensemble methods."

3 Ensemble methods

The idea of averaging over ten measure of fit constructed from random, non-overlapping subsets of the data can be generalized in a very powerful way. If averaging over measures of fit can help correct for overly optimistic fit assessments, then "averaging" over a set of CART trees might help correct for overly optimistic trees.

3.1 Bagging

The idea of combining fitted values from a number of fitting attempts has been suggested by several authors (LeBlanc and Tibsharini, 1996; Mojirshebeibani, 1997; 1999; Mertz, 1999). In an important sense, the whole becomes more than the sum of its parts. Perhaps the earliest procedure to exploit a combination of "random trees" is bagging (Breiman, 1996). Bagging stands for "bootstrap aggregation" and may be best understood initially as nothing more than an algorithm.

Consider the following steps in a fitting algorithm with a data set having n observations and a binary response variable.

1. Take a random sample of size n with replacement from the data.
2. Construct a classification tree as usual but do not prune.
3. Repeat steps 1-2 a large number of times.
4. For each case in the data set, count the number of times over trees that it is classified in one category and the number of times over trees it is classified in the other category
5. Assign each case to a category by a majority vote over the set of trees.[10]

[10]So if 51% of the time over a large number of trees a given case is classified as a "1," that becomes it estimated classification. The difference between the proportion of times a cases is *correctly* classified and the proportion of times it is *incorrectly* classified is called the "margin" for that case. If a case is correctly classified 75% of the time and incorrectly

An additional operation is often introduced. At each step, observations not included in the bootstrap sample (called "out-of-bag" data) are "dropped" down the tree. A record is kept of the class to which each out-of-bag observation is assigned. This information can be used to compute a more appropriate measure of the classification error, because it derives from the data not used to build the tree.[11]

The key point, however, is that assigning cases to categories by majority vote over a set of bootstrapped classification trees compensates for overfitting. One has the results from an "average" tree. But the price is high. There is no longer a single tree structure to interpret and no direct way, therefore, to learn which predictors are driving the outcome. For now, we will ignore this problem to further generalize the idea of combining fitted values.[12]

Making a clean conceptual break with CART can be instructive. Bagging may be seen less as a means to patch-up CART and more as an example of what Breiman (2001b) calls "algorithmic modeling." The basic idea is that one develops computer algorithms to solve particular data analysis problems. These problems often involve classification or forecasting in which a set of inputs is linked to one or more outputs. There may also be an interest in describing how the inputs are linked to outputs. But, there is no effort to represent in the algorithm the mechanisms by which the linkage occurs. In that sense, the associations revealed by algorithmic methods can be seen as descriptive. Algorithmic models are not causal models.

At the same time, there remains the very real numerical dependence on CART as a critical part of the algorithm. Consequently, certain features of the CART procedure need to be specified for the bagged trees to be properly constructed. Within algorithmic models, these arguments can be seen a "tuning parameters" affecting how the algorithm functions. For example, specification of a prior distribution for a binary response can be seen as a way to compensate for highly skewed response distributions, rather than as a belief about what the marginal distribution of the response really is.

The basic output from bagging is simply the predicted classes for each

classified 25% of the time, the margin is $.75 - .25 = .50$. Larger margins are desirable because a more stable classification is implied. Overfitting in CART tends to be more serious when the margins that would be constructed by bagging are small. The CART results (which depends on a single pass through the data) can be dominated by "noise."

[11] This is sometimes called an estimate "generalization error."

[12] Bagging has been used recently in a number of interesting ways. See for example, Hothorn et al. (2002).

case, often organized into a table. Commonly there is also an estimate of the classification error and a cross-tabulation of the classes predicted by the classes observed. The cross-tabulation can be useful for comparing the number of false positives to the number of false negatives. Sometime the software stores each of the trees as well, although these are rarely of any interest because the amount of information is typically overwhelming. One effective implementation of bagging is the package "ipred" available in R.

Once one considers bagging to be an example of an algorithmic method, the door is opened to a wide variety of other approaches. An algorithmic procedure called "random forests" is the next step through that door.

3.2 Random Forests

Just as in bagging, imagine constructing a large number of trees from bootstrap samples of data set. But as each tree is constructed, draw a random sample of predictors before each node is split. For example, if there are ten predictors, choose a random five as candidates for defining the split. Then construct the best split, as usual, but selecting only from the five chosen. Repeat this process for each node. And as in bagging, do not prune. Thus, each tree is produced from of a random sample of cases, and at each split a random sample of predictors. Then just as in bagging, classify by a majority vote of the full set of trees. Breiman calls the set of such trees a "random forest" (Breiman, 2001a).

By appropriate measures of fit, bagging is in principle an improvement over CART. And by these same criteria, random forests is in principle an improvement over bagging. Indeed, random forests is among the very best classifiers invented to date (Breiman, 2001a). The improvement goes well beyond compensation for overfitting and capitalizes on some new principles that are not yet fully understood.

Consider Figure 4. It was constructed from data drawn as 4 random samples of 15 from a bivariate Poisson distribution in which the two variables were moderately related. For each sample, the 15 points are simply connected by straight lines, which is nothing more than a set of linear interpolations. Because there is no smoothing, each fit is an unbiased estimate of the relationship between x and y at each value of x. Yet, one can see that the fitted lines vary substantially from sample to sample; the variance over fits is large. However, were one to take the average of y for each value of x and connect those points, the fit would still be unbiased and much smoother.

Moreover, if one repeated the same procedures with 4 new random samples, the averaged fits from the 2 sets of 5 samples would likely be far more similar than the fits that were constructed from a single sample. The variance estimated from a set of 4 samples will be smaller than the variance estimated from a single sample.

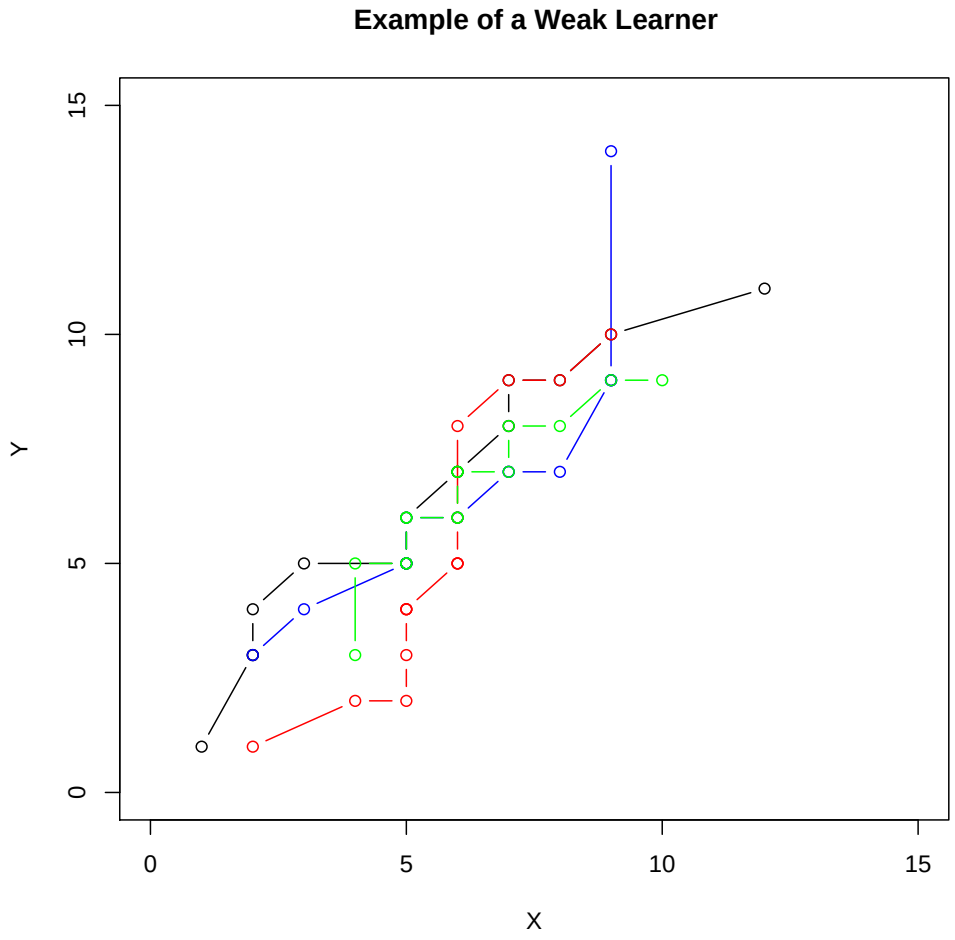

Figure 4: Four Realizations of a Weak Learner

Note that the same argument would apply had the response been categorical and the fitted values were interpreted as probabilities. The logic works

for classification problems as well. It is just harder to see what is going on in a graph such as Figure 4.

One can formalize these ideas a bit (Breiman, 2001c) by recalling that for conventional regression analysis and given set of predictor values x_0, the variance over repeated independent samples of an observed value of the response variable y around the fitted value $\hat{f}(x_0)$ is the sum of three parts: 1) an irreducible error, 2) the bias in the fitted value, and 3) the variance of the fitted value. More explicitly (Hastie et al., 2001: 197),

$$E[(y - \hat{f}(x_0))^2 | x = x_0] = \sigma_\epsilon^2 + [E\hat{f}(x_0) - f(x_0)]^2 + E[\hat{f}(x_0) - E\hat{f}(x_0)]^2. \quad (11)$$

There is nothing that can be done about the σ_ϵ^2, which results from the model's error term ϵ. In general, the more complex the model $\hat{f}$, the smaller the squared bias, but the larger the variance; there is a tradeoff between the two that constrains conventional modeling.

However, ensemble predictors can in principle sever the link between the bias and the variance. In an ideal world, the bias would be zero, and the variance would go to zero as the number of independent fits in the ensemble is increased without limit. In practice, one can make the $\hat{f}$ as complex as needed in an effort to achieve unbiasedness while shrinking the variance by averaging over a large number of fitted values. But in real applications, matters are more complex.

First, it is rare to know exactly how the response and the predictors are related; biased estimates can be expected so that the the bias will not be zero. This implies the need to work with very flexible fitting procedures that can respond well to patterns in the data (Breiman, 2001a). Still, some bias will almost certainly remain.

Second, one should use procedures that produce fits as uncorrelated as possible with one another (Breiman, 2001a). This is where random forests can improve over bagging. By sampling predictors at random as well as subsets of the data to analyze, greater independence between fits is typically achieved.

Third, it often essential to learn which predictors are driving the results. Yet, random forests, like bagging, leaves no tree behind to interpret. One approach to predictor importance is to record the decrease in the fitting measure (e.g., Gini Index) each time a given variable is used to define a split. The sum of these reductions for a given tree is measure of importance for that variable. For random forests, one can average this measure of importance over the set of trees.

Like variance partitions, however, reductions in the fitting criterion ignore the *predictive* ability of a model, perhaps the most important asset of ensemble methods. Breiman (2001a) has suggested another form of randomization to assess the role of each predictor, which is implemented in the latest version of random forests (randomForest V 4.1) in R.

1. Construct one or more measures of prediction error for each tree as usual by dropping the out-of-bag (OOB) data down the tree. Note that this is a real forecasting enterprise.

2. If there are K predictors, repeat step 1 K times, but each time with the values of a given predictor randomly shuffled. For each predictor, compute new measures of prediction error.

3. For each of the K predictors, average over trees the difference between the prediction error for the original model and the prediction error with the kth predictor shuffled.

The average increase in the prediction error when a given predictor is shuffled represents the importance of that predictor. There is currently a lively discussion on precisely how best to characterize the prediction error. But, that discussion is beyond the scope of this paper.

Unfortunately, none of these approaches reveal much about precisely how each predictor is related to the response. One useful solution, based on an earlier suggestion by Breiman and colleagues (Breiman, et al, 1984) is "'partial dependence plots" (Friedman, 2001; Hastie et al., 2001: section 10.13.2). For tree-based approaches, one proceeds as follows.

1. Grow a forest.

2. Suppose x is the predictor of interest, and it has V distinct values in the training data. Construct V data sets as follows.

   (a) For each of the V values of x, make up a new data set where x only takes on that value, leaving the rest of the data the same.

   (b) For each of the V data sets, predict the response using the random forests.

   (c) Average these predictions over the trees.

   (d) Plot the average prediction for each x against the values of x.

3. Go back to #2 and repeat for each predictor.

Partial dependence plots show the relationship between a given predictor and the response averaged within the joint values of the other predictors as they are represented in a tree structure. As such, the other predictors are being "held constant." And partial plots generalize to quantitative responses and to responses with more than two categories.

A final complication is that all fitting procedures do not perform well when the response is highly skewed. For classification procedures of the sort we have discussed, the fitting process is governed by reductions in heterogeneity (impurity). But if, for example, there is .98/.02 split in the response, one can classify cases correctly 98% of the time by simply assuming the more common response category. It will typically be difficult to find predictors that will reduce overall heterogeneity a meaningful amount. And if the goal is overall accuracy, there may well be nothing more to do. But when failure to classify rare cases correctly has a heavy price, alternative procedures are necessary.

Recall that within CART, there are two equivalent approaches to unbalanced data (Breiman et al., 1984). One can apply a prior marginal distribution that makes the rare events less rare or specify the relative costs of false positives and false negatives. Both options remain when trees are bagged, and specification of a compensating prior exists in current software for random forests.

There are other approaches as well currently being explored. One is differentially weighting the classification votes over trees. For example, one vote for classification in the rare category might count the same as two votes for classification in the common category. A second is to use a different threshold when classifications are determined. For instance, rather that classifying as "1" all cases in which the vote is larger than 50%, one might classify all cases as "1" when the vote is larger than 33%. Thirdly, when the bootstrap sample for each tree is drawn, one can oversample rare cases (e.g., two rare cases for every common case) in much the same spirit as using a compensating prior. The three rely at least implicitly on differential costs, but will not necessarily lead to exactly the same results.

Consider the following illustration. Recall the earlier CART classification example using data from the California Department of Corrections. Now we make the problem more difficult by trying to forecast the very small number of inmates (about 2.5%) who are likely to commit offenses that would likely

be felons if committed outside of prison: assault, rape, drug trafficking and the like. The predictors are the following.

1. Age at arrival at the Reception Center in years (AgeRec) — 16-20 (12%); 21-26 (23%); 27-35 (30%); 36 or older (35%)
2. Age at the time of the earliest arrest in years (AgeArr) — 0-17 (30%); 18-21 (41%); 22-29 (19%) 30-35 (6%); 36 or older (4%)
3. Associated with gang activity (Gang) — (19%)
4. Mental Illness (Psych) — (8%)
5. Previously served time in a county jail (Jail)— (66%)
6. Previously served time under the California Youth Authority (CYA) — (8%)
7. Previously served time under the California Department of Corrections (CDC) — (30%)
8. Sentence length in years (Term) — (median = 2; IQR = 2)[13]

The prior probability of very serious misconduct was used as a tuning parameter so that the ratio of false postives to false negatives approximated the relative costs of these errors to the Department of corrections. Table 2 shows how successful the forecasting exercise was. The results come from a the out-of-bag data not used to construct the random forest. Because of the highly unbalance nature of the response, forecasting accuracy overall will be very high even with no predictors. If "no misconduct" is forecast for every inmate, that prediction will be correct well over 95% of the time. Nevertheless, one can get some additional leverage on the problem of finding the few very high risk inmates by treating the prior distribution as a tuning parameter (For details, see Berk and Baek, 2003). Of the 18 true positives in the test data set, 5 are correctly identified, and the number of false negatives is from a policy point of view acceptable.

[13]Sentence length is recorded on the CDC intake form 839 so that it is capped at 25 years. Thus, for example, a life sentence without the possibility of parole is recorded as 25 years. The median and IQR are here unaffected by the truncation at 25.

All 18 of these inmates were missed by logistic regression model using the same predictors (Berk and Baek, 2003). CART did no better until a prior reflecting relative costs was introduced and even then did not perform as well as random forests.

	No Misconduct Predicted	Misconduct Predicted	Error
No Misconduct	453	29	0.06
Misconduct	13	5	0.72

Table 2: Random Forests "Confusion Table" for Forecasting Very Serious Inmate Misconduct

Figure 5 shows one measure of importance for each of the predictors used. The height of each bar represents the mean reduction over trees in the random forest fitting criterion (the Gini index) when a specified predictor becomes a new splitting criterion. The pattern is somewhat similar for other measures of importance and has the didactic advantage of close links to deviance partitions familiar to social scientists. Clearly, term length, age at arrest and age at reception into CDC are the substantially driving the fit. Partial dependence plots (not shown) indicate that the signs of the relationships are just what one would expect. In short, random forests provides useful results when other methods are likely to fail.

The results reported above were produced by the package "random forests" available in R. However, like so much of the software for ensemble methods, the software for random forests is undergoing rapid development. It is important to check for a new version every few months.

4 Boosting

Like CART, boosting is a forward stagewise additive model (Hastie et al., 2001: 305-306). But where CART works with smaller and smaller partitions of the data at each stage, boosting uses the entire data set as each stage.

Boosting gets its name from its ability to take a "weak learning algorithm" (which by definition performs just a bit better than random guessing) and "boosting" it into an arbitrarily "strong" learning algorithm (Schapire, 1999: 1). It "combines the outputs from many 'weak' classifiers to produce a powerful 'committee' "(Hastie et a., 2001: 299). But boosting formally has no

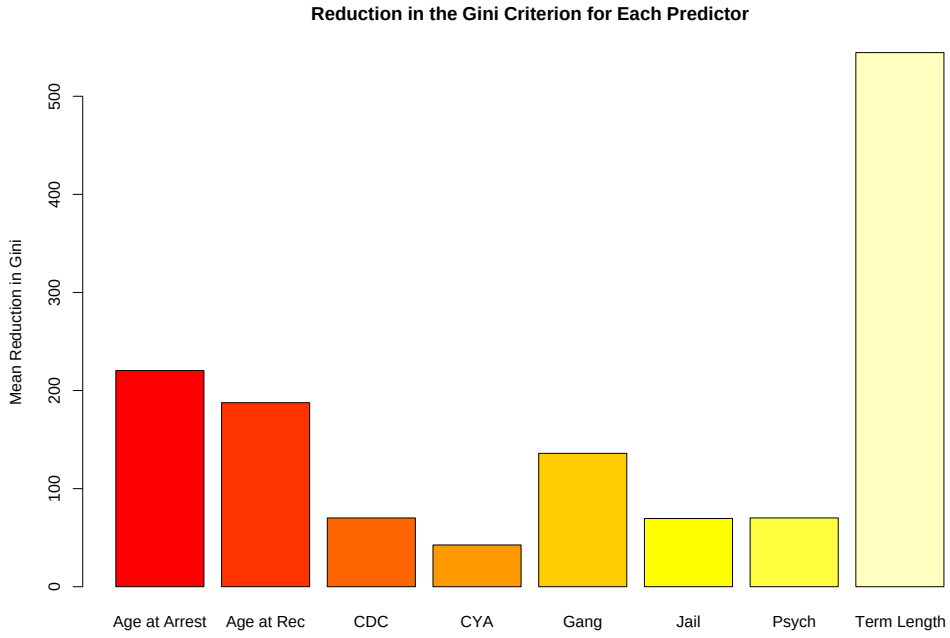

Figure 5: Average Reduction in Gini Criterion for Each Predictor for Very Serious Misconduct

stochastic components and so may not seem to be an ensemble method (at least as we have defined it).

Consider, for example, the ADAboost.M1 algorithm (Hastie et al., 2001: 301; Freund and Schapire, 1996; Schapire, 1999).[14]

1. Initialize the observations weights $w_i = 1/N, i = 1, 2, \ldots, N$.
2. For $m = 1$ *to* M:
   (a) Fit a classifier $G_m(x)$ to the training data using the weights w_i.
   (b) Compute: $err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{n} w_i}$.
   (c) Compute $\alpha_m = \log((1 - err_m)/err_m$.
   (d) Set $w_i \leftarrow w_i \cdot exp[\alpha_m \cdot I(y_i \neq G_m(x_i))], i = 1, 2, \ldots, N$
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

There are N cases and M iterations. $G_m(x)$ is a classifier for pass m over the data. Any number of procedures might be used to build a classifier, but highly truncated trees (called "stumps") are common. The operator I is an indicator variable equal to 1 if the logical relationship is true, and -1 otherwise. The binary response is coded 1 and -1 so that the sign defines the outcome. Classification error for pass m over the data is denoted by err_m.

The value of err_m is normalized as a proportion and transformed as α_m. The new weights, one for each case, are then computed as w_i. All cases incorrectly classified are "up-weighted" relative to the previous pass over the data by e^{α_m}. All cases correctly classified are "down-weighted" relative to the previous pass over the data by $e^{-\alpha_m}$. Consequently, ADAboost.M1 will pay relatively more attention in the next iteration to the cases that were misclassified. In the end, classification is determined by a vote over the M classifiers G_m, with each vote weighted by α_m.

Hastie and his colleagues (2001: 305-306) show that the loss function for each pass over the data is $e^{-yf(x)}$, where y is the response coded 1 or -1, and $f(x)$ here is the fitted values. Friedman et al., (2000) show that the overall

[14]The ADA stands for "adaptive." See Schapire (1999: 2) for an explanation. It is probably fair to say that to date ADAboost.M1 is the poster child for boosting and provides, therefore, a useful sense to the method.

additive expansion constructed by ADAboost.M1 is the same as estimating one half the log odds that $(y = 1|x)$.

While the exponential loss function can be computationally attractive, it may not be ideal. Cases that are atypical can lead ADAboost.M1 in a misleading direction. For example, work by Friedman and his colleagues (2000) suggests that logistic loss is more robust and less vulnerable to overfitting.

There is no stopping rule for ADAboost.M1. The number of fits is a tuning parameter that in practice depends on trial and error. Consequently, unlike CART, bagging and random forests, boosting does not meet statistical criteria commonly employed to justify particular procedures. Moreover, "boosting forever" is not consistent (Mannor, et al., 2002). Often the number of classification errors will decline up to a particular number of passes over the data and then begin to increase. The point of inflection can sometimes be treated as a useful stopping point. But, there is nothing in boosting implying convergence.

Still, there is a broad consensus that ADAboost.M1 performs well. Part of the reason may be found in a conjecture from Breiman (2001a: 20-21) that ADAboost.M1 is really a random forest. Recall that random number generators are deterministic computer algorithms. Breiman's conjecture is that ADAboost behaves as if the weights constructed at each pass over the data were stochastic. If this is correct, formal theory explaining the success of random forests may apply to ADAboost.M1. Adaboost.M1, and boosting more generally, may actually be ensemble methods.

The key output from boosting is much the same as the key output from bagging: predicted classifications, error rates and "confusion tables." And like for bagging software, information on the importance of predictors is not readily available, but lots of good ideas are being implemented. For example, "Generalized Boosting Regression Models" (GBM) in R has the measure of importance that lies behind partial dependence plots (Friedman, 2001) and a measure of importance much like the one Breiman implements in random forests.

5 Equal Interval Response Variables

For all of the procedures discussed, the transition to an equal interval response variable is relatively straightforward. To keep the exposition brief, consider CART as an illustration.

The key feature that changes is the splitting criterion. For the normal regression case, the impurity of a node is represented by the within-node sum of squares of the response:

$$i(\tau) = \sum (y_i - \bar{y}(\tau))^2, \tag{12}$$

where the summation is over all cases in the node, and $\bar{y}(\tau)$ is the mean of those cases. As before, the split s is chosen that it maximizes

$$\Delta(s, \tau) = i(\tau) - i(\tau_L) - i(\tau_R). \tag{13}$$

No cost weights are used because there are no false positives and false negatives. Then, to get the impurity for the entire tree, one sums over all terminal nodes to arrive at $R(T)$. There are pruning procedures akin to the categorical case. Bagging, out-of-bag estimates, random forests and boosting follow pretty much as before. For count data, one can replace the sum of squares criterion with a deviance criterion.

With CART, prediction is based on some feature of each terminal node, such as the mean or median. That value is assigned to each case in the node. For ensemble methods, prediction for each case is the average over replications of the summary statistic.

Without classification, the current options for representing predictor importance are fewer. They boil down to the contribution of each predictor to the fitting criterion, much like partitions of the total sum of squares in linear regression, or partitions of the deviance under the generalized linear model. It is also possible to work with partitions of the total prediction error, which has much the same flavor, but is based on test data or out-of-bag (OOB) observations.

6 Uses of Ensemble Methods

It should already be clear that ensemble methods can be used in social science analyses for classification and forecasting. Many can also be useful for describing the relationships between inputs and an outputs. But there are several other applications as well.

1. Ensemble methods can be used as overall diagnostic procedures for more conventional model building Berk, et al., 2004b). The larger the

difference in fit quality between one of the stronger ensemble methods and a conventional statistical model, the more information that the conventional model is probably missing. This may indicate which models need to be handled in a highly circumspect manner and/or to justify a search for better model specifications. However, some care must be taken in how the fit for a conventional model is computed. In particular, cross-validation should be used to compensate for overfitting (Efron and Tibshirani, 1993, Chapter 17). The usual methods based on "resubstitution" will generally be too optimistic.

2. Ensemble methods can be used to evaluate the relationships between explanatory variables and the response in conventional statistical models. Predictors or basis functions overlooked in a conventional model, may surface with an ensemble approach. Conversely, predictors thought be important in a conventional model, may prove to be worthless in output from an ensemble analysis. It does not necessarily follow that the ensemble results are superior. But the process of trying to understand why the two results differ will likely be instructive.

3. Analyses from observational data of the causal impact of an intervention can sometimes be usefully undertaken with propensity score adjustments (Rosenbaum, 2002, Chapter 10).[15] Propensity scores are usually the product of a logistic regression. Ensemble methods could perhaps do that job better (Berk et a., 2004). That is, the selection process could be better captured and the probability of membership in each treatment group estimated with less bias. More credible estimates of intervention effects would follow.

4. More generally, one could use ensemble methods to implement the covariance adjustments inherent in multiple regression and related procedures. One would "residualize" the response and the predictors of interest with ensemble methods. The desired regression coefficients would then be estimated from the sets of residuals. For example, if there is a single predictor on which interest centers (e.g., participation in a social program), one would simply regress the residualized response

[15]Propensity scores are the predicted probability of membership in each of the treatment groups. They are estimated in a separate analysis in which treatment group membership is the response variable and the predictors are variables thought to affect selection into treatment groups.

on the single residualized predictor. Both would have been constructed as the difference between the observed values and the estimated values from the ensemble output. It is likely that the adjustments for confounding would be more complete than with conventional covariance adjustments.

7 Conclusions

Ensemble methods and related procedures are not simply an elaboration on conventional statistical and causal modeling. They can represent a fundamental break with current traditions in applied social research dominated by causal modeling. There is no doubt that for many kind of applications, ensemble methods are the strongest procedures known. And with software now under development, they will soon be widely available.

8 References

Berk, R.A. (2003) *Regression Analysis: A Constructive Critique*. Sage Publications, Newbury Park, CA.

Berk.R.A. (2004) "Data Mining within a Regression Framework," in *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*, Oded Maimon and Lior Rokach (eds.), Kluwer Academic Publishers, Forthcoming, 2004.

Berk, R.A., and J. Baek. (2003) "Ensemble Procedures for Finding High Risk Prison Inmates." Department of Statistics, UCLA (under review).

Berk, R.A., Sorenson, S.B., and Y. He (2004a) "Developing a Practical Forecasting Screener for Domestic Violence Incidents," Department of Statistics, UCLA (under review).

Berk, R.A., Li, A., and L.J. Hickman (2004b) "Statistical Difficulties in Determining the Role of Race in Capital Cases: A Re-analysis of Data from the State of Maryland," Department of Statistics, UCLA (under review).

Breiman, L., Friedman, J.H., Olshen, R.A., and C.J. Stone (1984) *Classification and Regression Trees*. Monterey, Ca: Wadsworth

Breiman, L. (1996) "Bagging Predictors." *Machine Learning* 26:123-140.

Breiman, L. (2000) "Some Infinity Theory for Predictor Ensembles." *Technical Report 522*, Department of Statistics, University of California, Berkeley, California. and Brooks/Cole.

Breiman, L. (2001a) "Random Forests." *Machine Learning* 45: 5-32.

Breiman, L. (2001b) "Statistical Modeling: Two Cultures," (with discussion) *Statistical Science* 16: 199-231.

Breiman, L. (2001c) "Wald Lecture I: Machine Learning," at ftp://ftp.stat.berkeley.edu/pub/users/breiman/

Breiman, L. (2001d) "Wald Lecture II: Looking Inside the Black Box," at ftp://ftp.stat.berkeley. edu/pub/users/breiman/

Breiman, L. (2003) "Manual – Setting Up, Using, and Understanding Random Forests V4.0", at
ftp://ftp.stat.berkeley.edu/pub/users/breiman/

Christianini, N., and J. Shawne-Taylor. (2002) *An Introduction to Support Vector Machines*. Cambridge University Press.

Freund, Y., and R. Schapire. (1996) "Experiments with a New Boosting Algorithm. *Macine Learning: Proceedings for the Thirteenth International Conference* 148-156. Morgan Kaufmann, San Francisco.

Friedman, J.H. (2001) "Greedy Function Approximation: A Gradient Boosting Machine," *Annals of Statistics* 29:1189-1232.

Friedman, J.H. (2002) "Stochastic Gradient Boosting," *Computational Statistics and Data Analysis* 38(4):367-378.

Friedman, J., Hastie, T., and R. Tibsharini (2000). "Additive Logistic Regression: A Statistical View of Boosting" (with discussion). *Annals of Statistics* 28: 337-407.

Hastie, T., Tibshirani, R., and J. Friedman, *The Elements of Statistical Learning*. Springer-Verlag, 2001.

Hothorn, T. (2003) "Bundling Predictors in R." DSC 2003 Working Papers (draft version). http://www.ci.tuwien.ac.at/Conferences/DSC-2003/.

Hothorn, T., Berthold, L., Brenner, A., and M. Radespiel-Tröger. (2002) "Bagging Survival Trees." Department of Medical Informatics, Biometry and Epidemiology, Freidrich-alexander-University Erlangen-Nuremberg, preprint.

LeBlanc, M., and R. Tibshirani (1996) "Combining Estimates on Regression and Classification." *Journal of the American Statistical Association* 91: 1641-1650.

Mannor, S., Meir, R. and T. Zhang (2002) "The Consistency of Greedy Algorithms for Classification." In J. Kivensen and R.H. Sloan (eds.), COLT 2002, LNAI 2375: 319-333.

Mertz, C.J. (1999) "Using Correspondence Analysis to Combine Classifiers." *Machine Learning* 36: 33-58.

Mojirsheibani, M. (1997) "A Consistent Combine Classification Rule." *Statistics & probability Letters* 36:43-47

Mojirsheibani, M. (1999) "Combining Classifiers vis Discretization." *Journal of the American Statistical Association* 94: 600-609.

Rosenbaum, P.R. (2002) *Observational Studies*, second edition. Springer-Verlag.

Shapire, R.E. (1999) "A Brief Introduction to Boosting." *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence.*

Sutton, R.S. and A.G. Barto (1999) *Reinforcement Learning.* MIT Press.

Venables, W.N. and B.D. Ripley, *Modern Applied Statistics with S*, fourth edition, Springer-Verlag, 2002, Chapter 9.

Witten, I.H., and E. Frank, *Data Mining*, Morgan Kaufman Publishers, 2000.

Zhang, H., and B. Singer, *Recursive Partitioning in the Health Sciences*, Springer-Verlag, 1999.