Chapter 4
Bagging

4.1 Introduction

In this chapter, we make a major transition. We have thus far focused on statistical procedures that produce a single set of results: regression coefficients, measures of fit, residuals, classifications, and others. There is but one regression equation, one set of smoothed values, or one classification tree. Most statistical procedures operate in a similar fashion.

The discussion now shifts to statistical learning that builds on many sets of outputs aggregated to produce results. Such algorithms make a number of passes over the data. On each pass, inputs are linked to outputs just as before. But the ultimate results of interest are the collection of all the results from all passes over the data.

Bayesian model averaging may be a familiar illustration from another statistical tradition (Madigan et al. 1996; Hoeting et al. 1999). In Bayesian model averaging, there is an assumed $f(X)$; there is a "true model." A number of potentially true models, differing in the predictors selected, are evaluated. The model output is then averaged with weights determined by model uncertainty. Output from models with greater uncertainty are given less weight. From a statistical learning perspective, Bayesian model averaging has a number of complications, including the dependence that is necessarily built in across model results (Xu and Golay 2006). Also, it is not clear why a model with less uncertainty is necessarily closer to the true model. We address shortly how statistical learning procedures relying on multiple results proceed rather differently.

Aggregate results from many passes over the data can have several important benefits. For example, under the right circumstances, averaging over sets of fitted values can increase their stability. The averaging tends to cancel out results shaped by idiosyncratic features of the data. In turn, generalization error is reduced. An increase

The original version of this chapter was revised: See the "Chapter Note" section at the end of this chapter for details. The erratum to this chapter is available at https://doi.org/10.1007/978-3-319-44048-4_10.

in stability permits the use of more complex functions of the predictors when they are needed.

In this chapter, we focus on bagging, which capitalizes on a particular kind of averaging process that can address complexity and stability. In more traditional terms, bagging can have beneficial consequences for the bias-variance tradeoff. Sometimes you can have your cake and eat it too.

Although bagging can be applied to a wide variety of statistical procedures, we will again concentrate on classifiers. The rationale is largely the same: the exposition is more effective and the step to quantitative responses is easy to make. We begin with a return to the problem of overfitting. Although overfitting has been discussed several times in earlier chapters, it needs to be linked more directly to CART to help set the stage for a full exposition of bagging and subsequent procedures.

4.2 The Bagging Algorithm

The notion of combining fitted values from a number of fitting attempts has been suggested by several authors (LeBlanc and Tibshirani 1996; Mojirsheibani 1997, 1999). In an important sense, the whole becomes more than the sum of its parts. It is a bit like crowd sourcing.

"Bagging," which stands for "Bootstrap Aggregation," is perhaps the earliest procedure to exploit sets of fitted values over random samples of the data. Unlike model averaging, bagging is not a way to arrive at a model. Bagging is an algorithm that can help improve the performance of fitted values from a given statistical procedure. Breiman's remarkable 1996 paper on bagging is well worth a careful read.

For training data having N observations and a binary response variable, bagging takes the following form.

1. Take a random sample of size N *with replacement* from the data. These are sometimes called "bootstrap samples."
2. Construct a classification tree as usual.
3. Assign a class to each terminal node as usual, and store the class attached to each case and the predictor values that define the neighborhood in which in each terminal node resides (e.g., males under 30 years of age with a high school diploma).
4. Repeat Steps 1–3 a large number of times.
5. For each observation in the dataset, 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.
6. Assign each observation to a final class by a majority vote over the set of trees. If the outcome has two classes, and more than 50 % of the time over a large number of trees a given observation is classified as a 1, that becomes its classification. The same reasoning applies to the 0 class. The winning class is determined by a majority vote.

Although there remain important variations and details to consider, these are the key steps to produce “bagged” classification trees. Averaging occurs in the votes over classification trees. The voting results for each case are proportions that can be seen as means for response variables coded as 1 or 0.

The assigned class for each case is used much as it was for CART. Confusion tables are good place to start, especially if imputation or forecasting is in the offing. But there is no longer a single tree to interpret because there are many trees and no such thing as an average tree. Predictor values are linked to fitted classes, but not in a manner that can be substantively interpreted. We have a true blackbox statistical learning procedure. There will be more of them.

The idea of classifying by averaging over the results from a large number of bootstrap samples generalizes easily to a wide variety of classifiers beyond CART. Later we show that bagging can be usefully applied to quantitative responses as well. Nothing fundamentally changes.

4.3 Some Bagging Details

The bagging algorithm may seem straightforward. Bagging is just as way to average out unwanted noise. But there are a number of subtleties that we will need to carry forward in this chapter and later chapters.

4.3.1 Revisiting the CART Instability Problem

One good way to motivate bagging is to consider again the instability of classification trees. That instability can be readily apparent even across bootstrap samples that necessarily share large fractions of the data.

Figure 4.1 shows two such classification trees from the Titanic data. Although, as before, the first split for both is on gender, the two trees subsequently part company. The next two splits are the same for both trees, but the thresholds differ. Then, the splits that follow on the right branch differ in some of the predictors selected as well as thresholds. And the counts in all of the terminal nodes vary as well across the two trees. All of these differences lead to somewhat different classifications. For example, there are 10 boys between 9.5 and 13 years of age who are excluded from the far left terminal node on the top tree but who are not excluded from the far left terminal node on the bottom tree. Overall, about 10 % of the cases common to both trees were classified differently. The bottom tree was far more likely to assign the class of “survived” than the top tree. Again, this is a best case scenario in the sense that about 68 % of the observations in the two analyses are shared, and the overall sample size is relatively large.

A classification tree can be used for level I analyses when all one cares about is characterizing the data on hand. Instability becomes relevant for level II analyses.

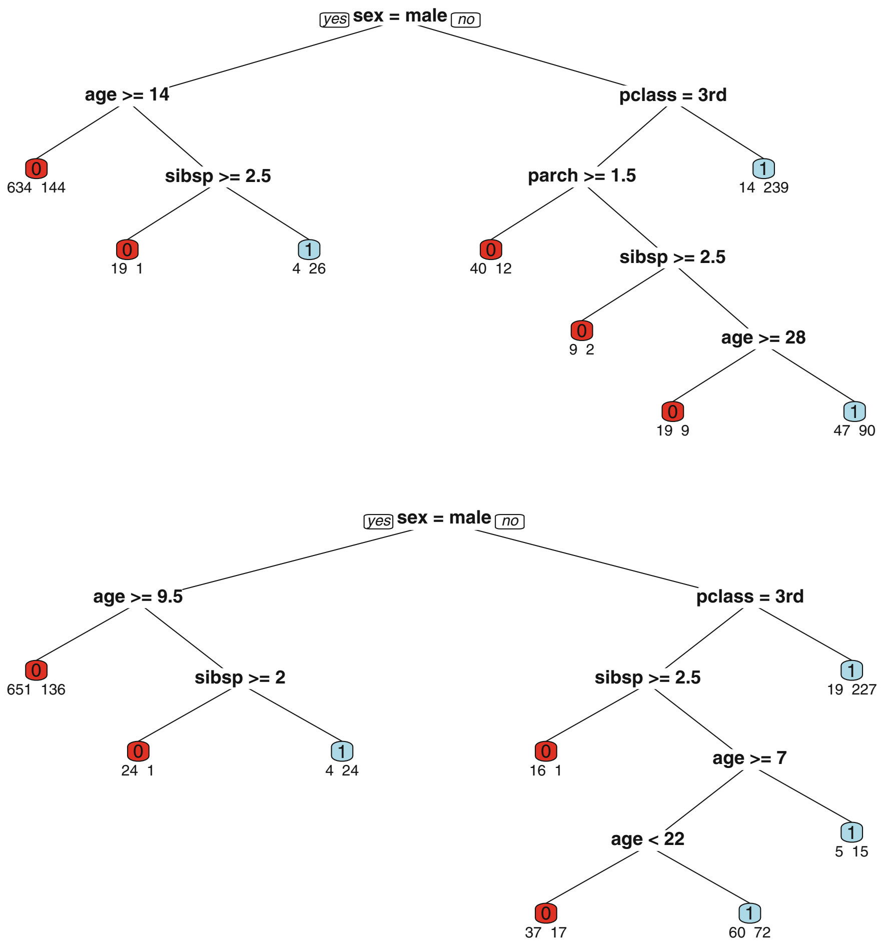

Fig. 4.1 Classification tree analysis of Titanic survival for two bootstrap samples of 1309 observations from the same training data (The *red nodes* are assigned the class of "perished," and the *blue nodes* are assigned the class of "survived")

For example, instability is a concern for generalization error, which speaks to performance in test data, not the data on hand. The same reasoning applies to bagging. Because bagging addresses instability, a level II perspective is required.

4.3.2 Some Background on Resampling

The top bagging priority is to reduce instability, and it all begins with resampling. Over the past two decades, many resampling procedures have been developed that can provide information about the sampling distributions of data-derived estimates. The bootstrap is perhaps the most well known. Other techniques include the jack-

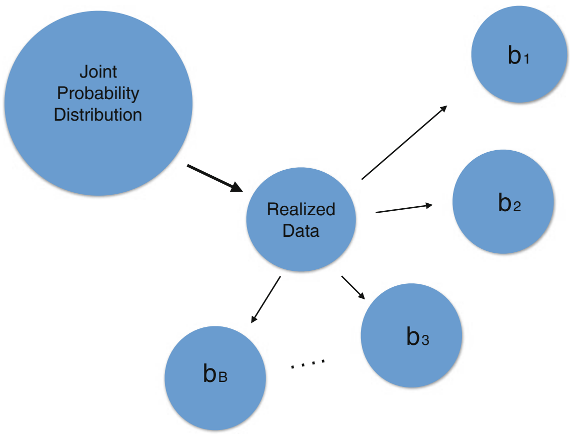

Fig. 4.2 A schematic for bootstrap sampling

knife (Efron and Tibshirani 1993: Sect. 10.5) and permutation tests (Edgington and Onghena 2007). Resampling procedures can provide asymptotically valid tests and confidence intervals making fewer assumptions than conventional methods and can sometimes be applied when there are no conventional methods at all.[1] For a discussion of bagging, we only need a few ideas from the bootstrap, and in particular, how one can generate a very large number of random sampling from training data. There are very good and extensive treatments of the bootstrap provided by Efron and Tibshirani (1993) and Hall (1997). Code in R to do a CART bootstrap is provided as part of an exercise at the end of the chapter.

The bootstrap is essentially a simulation of the frequentist thought experiment and as such, is automatically a level II formulation. For the frequentist, the data on hand are realized independently from a joint probability distribution or a finite population. In Fig. 4.2, the process is shown by the thick arrow toward the left side. The data on hand are seen as a single set of independent realizations from a limitless number of realized datasets that could be produced. Sample statistics in principle can be computed from each these realized datasets leading to one or more sampling distributions. From these, confidence intervals and statistical tests can follow. For example, if the mean were computed from each of the realized datasets, the result would be a sampling distribution for the mean. In practice, however, the data analyst usually gets to see only one realized dataset, and that is what the figure shows. Depending on the setting, the realized data could be training data, evaluation data, or test data.

[1]Recall that the jackknife can be seen as N-fold cross validation. Permutation tests are essentially the same as randomization tests and boil down to randomly shuffling some feature of the data on hand. For example, consider a conventional regression with a single predictor. Under the null hypothesis that the regression coefficient equals 0.0, one requires the sampling distribution of the regression coefficient if the null hypothesis is true. It can be effective to simulate that null distribution by randomly shuffling the response variable over and over, each time computing the value of the regression coefficient. With this approximation of the null distribution in hand, it is easy to calculate whether a regression coefficient as big or bigger than the one computed from the training data appears less the 5 % of the time. A statistical test has been performed with a critical value of .05. Good (2004) provides a very accessible treatment.

To approximate a limitless number of independently realized datasets, a large number of probability samples are drawn with replacement from the single realized dataset; hence the term "resampling." These probability samples are denoted by $b_1, b_2 \ldots, b_B$, where B is the total number of samples. If there are N observations in the realized data, each sample has N observations.[2]

Should the sampling be done without replacement, each sample of the realized data and the realized data itself will be identical and nothing has been gained. But should the N observations in each sample be drawn *with* replacement, the samples will almost certainly differ by chance from one another and from the realized data. This follows because a given observation in the realized data may be selected more than once. The set of samples drawn in this manner is meant to approximate the canonical frequentist thought experiment.[3]

Bagging exploits this resampling strategy in the algorithm's first step, but with an interpretative twist. A total of N observations with replacement is drawn. Sampling with replacement on the average causes about 37 % of cases to be excluded from a given sample. It follows that a substantial number of cases are selected more than once. From a frequentist perspective, one has the formal sampling properties required, but there is less information in such samples than had all of the observations appeared only once. Some statistical power can be lost, and procedures that are sample-size dependent can suffer.[4]

But, the resampling is not being used to construct an empirical sampling distribution. The resampling is an algorithmic device that allows one to draw a large number of random samples from the training data. Each sample is used to grow a tree whose fitted values are then averaged over trees. There is no estimation. There are also no confidence intervals or statistical tests.

At the same time, the sampling with replacement means that for each tree, a random sample of about 37 % of the observations are excluded from the tree-growing calculations. Each tree, therefore, automatically has "hold-out" observations, often called "out-of-bag" (OOB) observations. OOB observations are an immediate source of test data. Having valid test data as a byproduct of sampling with replacement is huge. Up to this point, test data had to be obtained as part of the data collection process or constructed from split samples.

[2]In more complete treatments, there can be more than or less than N observations in each bootstrap sample. Sampling without replacement is also an option as long as the sample size is less than N. However, some statistical procedures such as CART, are sample-size dependent. With more observation one can grow larger trees. It can make good sense, therefore to start with bootstrap samples having the same number of observations as the training data.

[3]There are many different kinds of bootstrap procedures, and it remains an important research area. The resampling just described is sometimes called a "pairs" bootstrap because both Y and X are sampled, or a "nonparametric" bootstrap because there is no model specified through which the realized data were generated.

[4]If one draws random samples *without* replacement with $.50 \times N$ observations, one has on the average a dataset with about the same information content as N observations drawn at random with replacement (Buja and Stuetzle 2006). Still, sampling with replacement is the usual approach.

4.3.3 *Votes and Probabilities*

For each case in the bootstrap sample, there is a vote over trees. For a binary outcome, the class with the majority vote is the class assigned to that case. For outcomes with more than two classes, the class with a plurality is the class assigned to that case. Because each classifier is grown with a random sample of the training data, the proportion of votes each class musters has the look and feel of probabilities. However, the samples drawn in bagging are not fully independent, which undermines the usual assumption of independent trials, and compromises treating vote proportions are probabilities. But it's worse.

One must be clear about what such probabilities could represent (Breiman 1996: Sect. 4.2). Suppose for a particular set of predictor values the true outcome probability of a 1 is .80. Suppose also that each classification tree votes for 1, given those x-values. Although the true probability of a 1 is .80, the vote "probability" over trees is 1.0. Clearly, two different kinds of outcomes are in play: how a tree votes and the outcome class for a given case. They must not be confused. More will be said about this in the next chapter.

4.3.4 *Imputation and Forecasting*

Obtaining fitted classes for imputation or forecasting follows directly from the way fitted values from the training data are computed. Recall that for a single classification tree, a new set of predictor values with an unknown outcome are "dropped down" the tree. The assigned class of the terminal node in which the case lands is the imputed or forecasted class. Also recall that this process can be represented in a conventional regression structure where the task is prediction. There is nothing mysterious going on.

When there are K classification trees, the set of predictor values is dropped down each of the K trees. Then as before, a vote is taken. The winning class is the forecast for those x-values. But as just noted, the winning proportion is not an estimate of the probability that the imputation or forecast is correct. This seems to be an all too common error.

4.3.5 *Margins*

The meaning of "margin" in bagging is somewhat different from the meaning of "margin" in margin maximizing forms of statistical learning (e.g., adaboost, support vector machines). But the statistical goals are the same: to arrive at stable classifications. From a bagging perspective, Bremen (2001a: 7) defines the margin as

$$mg(\mathbf{X}, Y) = av_k I(h_k(\mathbf{X}) = Y) - \max_{j \neq Y} av_k I(h_k(\mathbf{X}) = j), \tag{4.1}$$

where for randomly realized data and a given case, there is an ensemble of K classifiers denoted by $h_k(\mathbf{X})$, Y is the correct class, j is some other class, and $I(.)$ is the indicator function as before. The K classifiers might be K classification trees. In words, over the K classifiers, there is a proportion of times the case is classified correctly, and a maximum proportion of times the case is classified incorrectly. The difference in the two proportions is the margin for that case.[5] "The larger the margin, the more confidence in the classification" (Breimen 2001: 7).

Suppose that over all of the bagged trees, an observation is correctly classified 75 % of the time and incorrectly classified 25 % of the time. The margin is $.75 - .25 = .50$. A negative margin implies misclassification. If an observation is correctly classified 30 % of the time and incorrectly classified 70 % of the time, the margin is $.30 - .70 = -.40$.

A lopsided vote in favor of the correct class conveys that despite noise introduced by the resampling, most of the time the case is classified correctly. One can say that the correct classification for that case is highly reliable. A lopsided vote in favor of the incorrect class is also highly reliable; reliable and wrong. If the vote is very close, the case is just about as likely to be misclassified as classified correctly. Systematic relationships between the response and the predictors cannot meaningfully overcome the algorithm-generated noise. One can say that the classification, whether correct or incorrect, is unreliable.

One can only know when a classification is correct, if the actual class is known. That will be true in training data, evaluation data, and test data. In that context, margins can be used as diagnostic tools. How reliable are the correct classifications? How reliable are the incorrect classifications? Ideally, the former are very reliable and the latter are not. In new realizations of the data, there is then a good chance that many of the incorrect classifications will be overturned. One might also be able to identify particular types of cases for which misclassifications are likely and/or unreliable.

Forecasting applications differ because the actual outcome is not known. The votes over classifiers still represent reliability, but whether the classification is correct or incorrect is not known. Nevertheless, reliability is important. Suppose for a given case, the vote is close. Because vote is close, bagging the very same data again could easily result in a close vote the other way. The initial forecasted class is not reliable; the forecast could have easily been different. It is usually important for stakeholders who will use the forecast to know the forecast's reliability, even if they do not know whether the forecast is correct. Should the vote be lopsided, the forecasted class is reliable, and even though the true class is not known, the forecast may be given more credibility.

Whatever the level of confidence, it is with respect to the performance of the bagged classifier itself. Was it able to classify particular cases with sufficient reliability? It is not confidence about any larger issues such as whether the fitted values

[5]The average of an indicator variable is a proportion. The "max" allows for more than two outcome classes with the proper comparison a worse case scenario.

are good approximations of the true response surface. It is nothing like a model diagnostic or model misspecification test. There is often confusion on this point.

In summary, large margins can be a major asset. For each case, the margin can be a measure of reliability for the class assigned. Larger margins imply greater bagging reliability. Forecasting is different. In the earlier prison inmate instance, housing decisions at intake were based on an inmate's forecasted misconduct class. If the vote for a given inmate is equivocal, prison staff properly might decide to base the housing decision on other information. If the vote is decisive, prison staff properly might base the housing decision primarily on the class assigned.

4.3.6 Using Out-Of-Bag Observations as Test Data

In conventional CART software, a tree is grown with training data, and the training data used to grow the tree are used again to compute the number of classification errors. The training data are dropped down the tree to determine how well the tree performs. The training data are said to be "resubstituted" when tree performance is evaluated.

In some implementations of bagging, out-of-bag observations from each tree can be treated as a test dataset and dropped down the tree. There need be no resubstitution. A record is kept of the class with which each out-of-bag observation is labeled, as well as its values on all of the predictors. Then in the averaging process, only those class labels are used. In other words, the averaging for a given case over trees is done only using the trees for which that case was not used to grow the tree. This leads to still more honest fitted values and more honest confusion tables than with conventional bagging.

4.3.7 Bagging and Bias

Although the major target of bagging is the variance of fitted values, there can be in certain situations a reduction in the bias as well. Figure 4.3 illustrates how bagging can affect the bias. To keep the graph simple, there is a smooth nonlinear $f(X)$ linking a single predictor to a quantitative response Y. The true response function is shown.

Imagine that a regression tree is applied one time to each of three different bootstrap samples of the data. Each time, only one break in the predictor is allowed. (Such trees are sometimes called "stumps.") Three step functions that could result are overlaid as open rectangles. One has a very rough approximation of the $f(X)$. If that function is the estimation target, there is substantial bias.

Suppose now that there are eleven bootstrap samples and eleven stumps. Eleven step functions that could result are shown with the light blue rectangles. Clearly, the approximation is better and bias is reduced. Because in bagging there are often hundreds of trees, there is the possibility of approximating complex functions rather

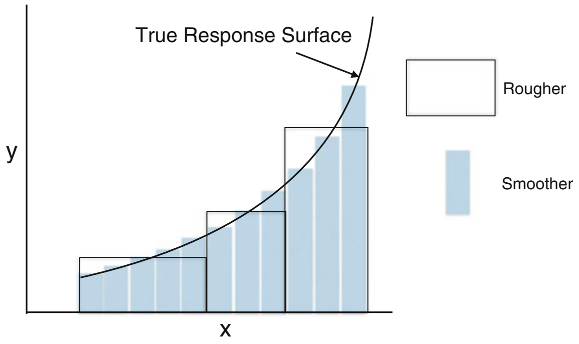

Fig. 4.3 How bagging smooths using a set of step functions to approximate the true response surface

well. The same reasoning applies to categorical outcomes. In short, bagging can reduce bias by what is, in effect, smoothing (Bühlmann and Yu 2002).

In addition, noted earlier was an indirect impact that bagging can have on bias. Because of the averaging in bagging, one can employ more complex functions of the data with less worry about the impact of overfitting on generalization error. For example, bagging gives more license to grow very large trees with few observations in terminal nodes. The larger trees can lead to less bias while bagging increases the stability of fitted values.

4.3.8 Level I and Level II Analyses with Bagging

As always, a level I analysis is justified. For bagging that may mean little more than studying a histogram of the fitted values or examining a confusion table derived from resubstituted data. There might also be interest in the margins. But such analyses go primarily to how well the bagging procedure performs. Because there is no longer a tree to interpret, there is little that can be easily done to describe how the predictors are related to response. In the next chapter, some tools will be introduced that can help.

However, the usual motivation for bagging and the usual interest in margins imply concerns about generalization error. Generalization error is a level II matter. Moreover, bagged output can then be seen as estimates. Assuming that the training data plausibly can be treated as independent random realizations from a relevant joint probability distribution, the level II issues are much the same as discussed for CART. There is an estimation target, which is the population approximation of the true response surface. The details of that approximation depend on the classifier being used. If the classifier is adaptive (i.e., inductive), model selection can be a serious estimation complication. As before, the best hope is to have test data. The training data and bagging results are taken to be fixed. If test data are used to construct fitted values, the observed class and the fitted classes can be tabulated in a confusion table from which various kind of generalization error can be estimated. For example, there can be an overall misclassification proportion weighted by the asymmetric costs of classification errors. There can also be misclassification proportions for either of the two (or more) outcome classes. One can also use the fitted classes from the test data as asymptotically unbiased estimates of the bagging approximation response surface. Imputation and forecasting directly follow.

4.4 Some Limitations of Bagging

In general, bagging is a reasonably safe procedure. But it is hardly a panacea. Sometimes it does not help and on occasion it can make things worse.

4.4.1 Sometimes Bagging Cannot Help

Bagging only returns an average of fitted values that is different from those that could be obtained from one pass over the original data if the fitting procedure is a nonlinear or an adaptive function of the data (Hastie et al. 2009: 282). For example, there is no reason to apply bagging to conventional linear regression. The average of the fitted values over a large number of bootstrap samples would be effectively the same as the fitted values obtained from conventional linear regression applied once to the training data. In contrast, there can be useful differences for smoothing splines when the value of λ is determined empirically. This point helps to underscore an earlier discussion about the bootstrap samples used in bagging: in bagging, the goal is not to approximate a sampling distribution but to allow for many passes over the data.

4.4.2 Sometimes Bagging Can Make the Bias Worse

Look again at Fig. 4.3. Suppose $f(X)$ is really very jagged, much like a step function. Then, the smoothing that bagging accomplishes can increase bias because the smoothing on the average moves the fitted values away from the true response surface. One does not want the sharp corners of the CART estimates "sanded off". Classification can also be adversely affected.

Weak classifiers can also create problems, especially when the distribution of the response is highly unbalanced. Weak classifiers are sometimes defined as those that do not do materially better than the marginal distribution. Suppose the marginal distribution of the response is unbalanced so that it is very difficult for a fitting procedure using the predictors to perform better than the marginal distribution. Under

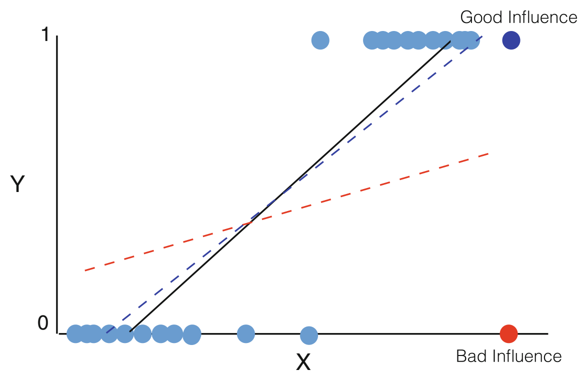

Fig. 4.4 Good and bad influence in bagging

those circumstances the rare class will likely be misclassified most of the time because votes will be typically won by the class that is far more common.[6]

To illustrate this point, suppose there is a binary response variable, and for the moment, we are interested in a single observation that actually happens to be a "success." Over K classification trees, that observation is classified as a success about two times out of ten. So, the classification for that observation will be wrong about 80 % of the time. But if one classifies by majority vote, the class assigned would be a failure and that would be wrong 100 % of the time. Because the K classifiers do a poor job, the majority vote makes things worse. Stronger classifiers typically would place that observation in terminal nodes where the major of the cases were successes. Then the vote over trees would help.

In practice, such problems will be rare if the data analyst pays attention to how the classifier performs before bagging is applied. If it performs very poorly, bagging risks making things worse. We show in later chapters that if one has a set of weak classifiers, alternative procedures may be called for that can help.

4.4.3 Sometimes Bagging Can Make the Variance Worse

Bagging sometimes can also perform poorly with respect to the variance (Grandvalet 2004). Figure 4.4 shows a scatterplot with a binary outcome. The observations are represented by filled circles. The light blue circles represent the mass of the data. The dark blue circle and the red circle are high leverage observations because they are outliers in the x-direction. Consider now their role for the fitted values.

To keep the exposition simple, suppose that within the range of X, true response surface is a linear function of the X. The solid black line shows the fitted values with both the blue circle and the red circle excluded from the dataset. Suppose that the blue

[6]Moreover, sometimes the procedure will fail because none of the rare cases are included in a given bootstrap sample.

outlier is not included in the data. The broken red line shows the fitted values with the lower-right outlier included. The lines are rather different, implying that whether that red outlier is included in the analysis alters the response function substantially. Therefore, the outlier is influential. In a bagging application, the fitted values will vary widely depending on whether the red observation happens to be included. Then, averaging over classifiers reduces the variance of the fitted values. Bagging works as it should.

In contrast, suppose that the red outlier is excluded from the data. Whether the blue outlier is included makes a small difference in the fitted values. It happens to fall near the line generated by the other fitted values. The broken blue line shows the result. Deleting the blue outlier does not change the fit a great deal. Therefore, it is not influential. Rather, it helps to stabilize the fitted values. When it is excluded because of bootstrap sampling that added stability is lost. Bagging may increase the variance. In practice, however, such situations are very rare, and the increase in variance is likely to be small.

The problems with bagging just described have their analogues for quantitative responses. Bagging is at its best when the problem to overcome is instability. Bagging when the fitted values are already very stable can make things worse. In practice, it can be useful to inspect several of the K classifiers derived from different bootstrap samples to see how serious the instability may be.

4.5 A Bagging Illustration

In practice, bagging is not used much as a stand-alone procedure. There are far better statistical learning tools. But like classification and regression trees, it can be a key component of more effective approaches and many of the details need to be understood.

Consider now bagging applied to the Titanic data largely to show some R-code. The library in R is *ipred* and bagging procedure itself is *bagging()*.[7] We use the same classification tree specification as before, which assumes symmetric costs for classification errors. Table 4.1 is the confusion table constructed solely from the training data and Fig. 4.5 shows the code responsible. By all of the performance measures shown in the table, the fit is quite good, but the table is constructed from in-sample data. Also, there is no tree to interpret.

[7]The package *ipred()* is written by A. Peters, T. Hothorn, B.D. Ripley, T. Therneau, and B. Atkinson. There are number of bagging-related procedures in *ipred()*.

Table 4.1 Bagged classification tree confusion table for survival on the Titanic (N = 1309)

	Classify perished	Classify survived	Model error
Perished	759	50	.05
Survived	100	400	.21
Use error	.12	.10	Overall error = .12

```
## Bagging
library(PASWR) # Where the data are
data("titanic3") # Load data
library(ipred) # Load library

# Bag Classification trees
out1<-bagging(as.factor(survived)~sex+age+pclass+sibsp+parch,
               data=titanic3,coob=T, keepX=T, nbagg=50,
               minsplit=10, cp=.05, xval=0)

fitted<-predict(out1, newdata=titanic3,
                type="class") # fitted class
tab<-table(titanic3$survived,fitted) # confusion table
prop.table(tab,1) # use error
prop.table(tab,2) # model error
```

Fig. 4.5 R code for bagging Titanic data

4.6 Bagging a Quantitative Response Variable

Bagging works by the same principles when the response variable is quantitative. Recall that CART constructs a regression tree by maximizing the reduction in the error sum of squares at each split. Each case is placed in a terminal node with a conditional mean. That mean is the fitted value for all cases of that terminal node.

All of the concerns about CART instability apply, especially given the potential impact that outliers can have on the fitting process when the response variable is quantitative. Because of the sum of squares loss function, a few cases that fall a substantial distance from the mass of the data can produce results that can vary substantially over samples, do not characterize well the mass of the data, and do not generalize well either.

With a numerical response variable, bagging averages over trees in much the same way it averages over trees when the response variable is categorical. For each tree, each observation is placed in a terminal node and assigned the mean of that terminal node. Then, the average of these assigned means over trees is computed for each observation. This average value for each case is the bagged fitted value used. The averaging process will tend to moderate instability. If for each tree, the OOB data that are placed in terminal nodes is used in the averaging, stability can be improved more effectively.

4.7 Summary and Conclusions

Bagging is an important conceptual advance and a useful tool in practice. The conceptual advance is to aggregate fitted values from a large number of bootstrap samples. Ideally, many sets of fitted values, each with low bias but high variance, may be averaged in a manner than can effectively reduce the bite in the bias–variance tradeoff. Thanks to bagging, there can be a way to usefully address this long-standing dilemma in statistics. Moreover, the ways in which bagging aggregates the fitted values is the basis for other statistical learning developments.

In practice, bagging can generate fitted values that often reproduce the data well and forecast with considerable accuracy. Both masters are served without making unrealistic demands on available computing power. Bagging can also be usefully applied to a wide variety of fitting procedures. However, bagging is not much used as a stand-alone procedure because there are statistical learning procedures readily available that import the best features of bagging, add some new wrinkles, and then perform better.

In addition, bagging also suffers from several problems. Perhaps most important, there is no way within the procedure itself to depict how the predictors are related to the response. With test data or OOB data, one can obtain a more honest set of fitted values and a more honest evaluation of how good the fitted values really are. But as an explanatory device, bagging is pretty much a bust. Other tools are needed, which are considered in the next chapter.

A second problem is that because so much of the data are shared from tree to tree, the fitted values are not independent. The common set of available predictors can build in additional dependence. Consequently, the averaging is not as effective as it could be. This too is addressed shortly.

Third, bagging may not help much if the fitting function is consistently and substantially inappropriate. Large and systematic errors in the fitted values are just reproduced a large number of times and do not, therefore, cancel out in the averaging process. For categorical response variables, bagging a very weak classifier can sometimes make things worse.

Fourth, the bootstrap sampling can lead to problems when categorical predictors or outcomes are highly unbalanced. For any given bootstrap sample, the unbalanced variable can become a constant. Depending on the fitting function being bagged, the entire procedure may abort.

Finally, bagging can actually increase instability if there are outliers that help to anchor the fit. Such outliers will be lost to some of the bootstrap samples. It is difficult in practice to know whether this is a problem or not.

Bagging can be extended so that many of these problems are usefully addressed, even if full solutions are not available. We turn to some of these potential solutions in the next chapter. They are found in another form of statistical learning, still farther away from conventional regression analysis.

Exercises

Problem Set 1

The sampling done in bagging must be with replacement. Run the following code and compare the tables. How many duplicate observations are there in s1 compared to s2? Write code to find out. Run the code a second time. Again, how many duplicate observations are there in s1 compared to s2? Write code to find out. What have you learned about the differences between the samples drawn by the two methods?

```
x<-1:100
s1<-sample(x,replace=T)
table(s1)
s2<-sample(x,replace=F)
table(s2)
```

Problem Set 2

The goal of this exercise is to compare the performance of linear regression, CART, and bagging applied to CART. Construct the following data set in which the response is a quadratic function of a single predictor.

```
x1=rnorm(500)
x12=x1^2
y=1+(2*(x12))+(2*rnorm(500))
```

1. Plot the 1 + (2 × x12) against x1. This is the "true" relationship between the response and the predictor without the complication of the disturbances. This is the $f(X)$ you hope to recover from the data.
2. Proceed as if you know that the $f(X)$ is quadratic. Fit a linear model with x12 as the predictor. Then plot the fitted values against x1. You can see how well linear regression does when the functional form is known.
3. Now suppose that you do not know that the $f(X)$ is quadratic. Apply linear regression to the same response variable using x1 (not x12) as the sole predictor. Construct the predicted values and plot the fitted values against x1. How do the fitted values compare to what you know to be the correct $f(X)$? (It is common to assume the functional form is linear when the functional form is unknown.)
4. Apply CART to the same response variable using *rpart()* and x1 (not x12) as the sole predictor. Use the default settings. Construct the predicted values, using *predict()*. Then plot the fitted values against x1. How do the CART fitted values compare to what you know to be the correct $f(X)$? How do the CART fitted values compare to the fitted values from the linear regression with x1 as the sole predictor?

5. Apply bagging to the same response variable using *bagging()* from the *ipred()* library, and x1 as the sole predictor. Use the default settings. Construct the predicted values using *predict()*. Then plot the fitted values against x1. How do the bagged fitted values compare to the linear regression fitted values?
6. You know that the relationship between the response and x1 should be a smooth parabola. How do the fitted values from CART compare to the fitted values from bagging? What feature of bagging is highlighted?

Problem Set 3

Load the dataset "Freedman" from the *car* library. For 100 American cities, there are four variables: the crime rate, the population, population density, and proportion nonwhite. As before, the crime rate is the response and the other variables are predictors.

1. Use *rpart()* and its default values to fit a CART model. Compute the root mean square error for the model. One way to do this is to use *predict.rpart()* to obtain the fitted values and with the observed values for the variable "crime," compute the root mean square error in R. Then use *bagging()* from the library *ipred* and the out-of-bag observations to obtain a bagged value for the root mean square error for the same CART model. Compare the two estimates of fit and explain what you see. Keep in mind that at least two things are going on: (1) in-sample v. out-of-sample comparisons, and (2) the averaging that bagging provides.
2. Using *sd()*, compute the standard deviation for the CART fitted values and the bagged fitted values. Compare the two standard deviations and explain what you see.

Problem Set 4

Load the dataset "frogs" from the library *DAAG*. Using "pres.abs" as the response, build a CART model under the default settings.

1. Construct a confusion table with "pres.abs" and the predicted classes from the model. Now, using *bagging()* from the library *ipred*, bag the CART model using the out-of-bag observations. Construct a confusion table with "pres.abs" and the bagged predicted classes from the model. Compare the two confusion tables and explain why they differ. Keep in mind that at least two things are going on: (1) in-sample v. out-of-sample comparisons, and (2) the averaging that bagging provides.