of the process. In the present example, this means that the best predictor and corresponding split point are found from 4/5ths of the data. The effect of predictor choice is seen in the top right panel. Since the class labels are independent of the predictors, the performance of a stump on the 4/5ths training data contains no information about its performance in the remaining 1/5th. The effect of the choice of split point is shown in the bottom left panel. Here we see the data for predictor 436, corresponding to the blue dot in the top left plot. The colored points indicate the 1/5th data, while the remaining points belong to the 4/5ths. The optimal split points for this predictor based on both the full training set and 4/5ths data are indicated. The split based on the full data makes no errors on the 1/5ths data. But cross-validation must base its split on the 4/5ths data, and this incurs two errors out of four samples.

The results of applying five-fold cross-validation to each of 50 simulated datasets is shown in the bottom right panel. As we would hope, the average cross-validation error is around 50%, which is the true expected prediction error for this classifier. Hence cross-validation has behaved as it should. On the other hand, there is considerable variability in the error, underscoring the importance of reporting the estimated standard error of the CV estimate. See Exercise 7.10 for another variation of this problem.

7.11 Bootstrap Methods

The bootstrap is a general tool for assessing statistical accuracy. First we describe the bootstrap in general, and then show how it can be used to estimate extra-sample prediction error. As with cross-validation, the bootstrap seeks to estimate the conditional error $\mathrm{Err}_{\mathcal{T}}$, but typically estimates well only the expected prediction error Err.

Suppose we have a model fit to a set of training data. We denote the training set by $\mathbf{Z} = (z_1, z_2, \ldots, z_N)$ where $z_i = (x_i, y_i)$. The basic idea is to randomly draw datasets with replacement from the training data, each sample the same size as the original training set. This is done B times ($B = 100$ say), producing B bootstrap datasets, as shown in Figure 7.12. Then we refit the model to each of the bootstrap datasets, and examine the behavior of the fits over the B replications.

In the figure, $S(\mathbf{Z})$ is any quantity computed from the data $\mathbf{Z}$, for example, the prediction at some input point. From the bootstrap sampling we can estimate any aspect of the distribution of $S(\mathbf{Z})$, for example, its variance,

$$\widehat{\mathrm{Var}}[S(\mathbf{Z})] = \frac{1}{B-1}\sum_{b=1}^{B}(S(\mathbf{Z}^{*b}) - \bar{S}^{*})^2, \tag{7.53}$$

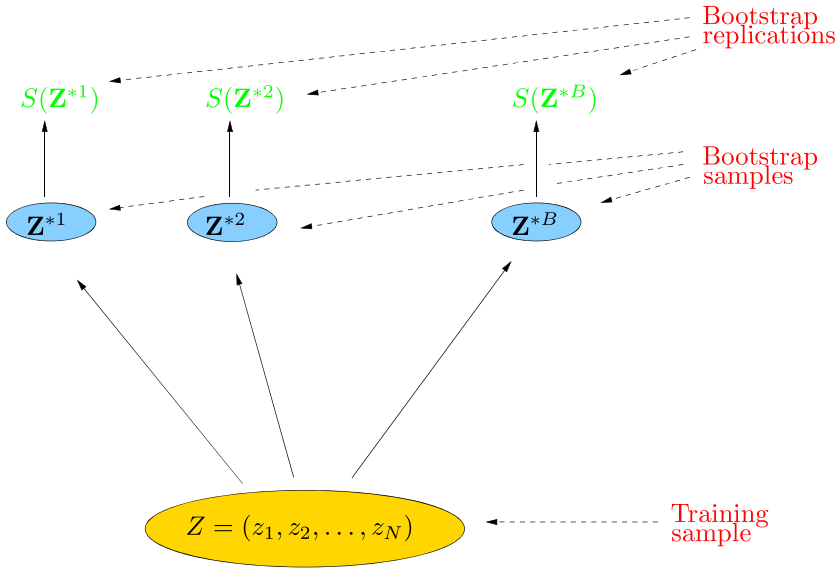

FIGURE 7.12. *Schematic of the bootstrap process. We wish to assess the statistical accuracy of a quantity $S(\mathbf{Z})$ computed from our dataset. B training sets $\mathbf{Z}^{*b}$, $b = 1, \ldots, B$ each of size N are drawn with replacement from the original dataset. The quantity of interest $S(\mathbf{Z})$ is computed from each bootstrap training set, and the values $S(\mathbf{Z}^{*1}), \ldots, S(\mathbf{Z}^{*B})$ are used to assess the statistical accuracy of $S(\mathbf{Z})$.*

where $\bar{S}^* = \sum_b S(\mathbf{Z}^{*b})/B$. Note that $\widehat{\mathrm{Var}}[S(\mathbf{Z})]$ can be thought of as a Monte-Carlo estimate of the variance of $S(\mathbf{Z})$ under sampling from the empirical distribution function $\hat{F}$ for the data $(z_1, z_2, \ldots, z_N)$.

How can we apply the bootstrap to estimate prediction error? One approach would be to fit the model in question on a set of bootstrap samples, and then keep track of how well it predicts the original training set. If $\hat{f}^{*b}(x_i)$ is the predicted value at x_i, from the model fitted to the bth bootstrap dataset, our estimate is

$$\widehat{\mathrm{Err}}_{\mathrm{boot}} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{i=1}^{N} L(y_i, \hat{f}^{*b}(x_i)). \tag{7.54}$$

However, it is easy to see that $\widehat{\mathrm{Err}}_{\mathrm{boot}}$ does not provide a good estimate in general. The reason is that the bootstrap datasets are acting as the training samples, while the original training set is acting as the test sample, and these two samples have observations in common. This overlap can make overfit predictions look unrealistically good, and is the reason that cross-validation explicitly uses non-overlapping data for the training and test samples. Consider for example a 1-nearest neighbor classifier applied to a two-class classification problem with the same number of observations in

each class, in which the predictors and class labels are in fact independent. Then the true error rate is 0.5. But the contributions to the bootstrap estimate $\widehat{\text{Err}}_{\text{boot}}$ will be zero unless the observation i does not appear in the bootstrap sample b. In this latter case it will have the correct expectation 0.5. Now

$$\begin{aligned}\Pr\{\text{observation } i \in \text{bootstrap sample } b\} &= 1 - \left(1 - \frac{1}{N}\right)^N \\ &\approx 1 - e^{-1} \\ &= 0.632. \end{aligned} \tag{7.55}$$

Hence the expectation of $\widehat{\text{Err}}_{\text{boot}}$ is about $0.5 \times 0.368 = 0.184$, far below the correct error rate 0.5.

By mimicking cross-validation, a better bootstrap estimate can be obtained. For each observation, we only keep track of predictions from bootstrap samples not containing that observation. The leave-one-out bootstrap estimate of prediction error is defined by

$$\widehat{\text{Err}}^{(1)} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L(y_i, \hat{f}^{*b}(x_i)). \tag{7.56}$$

Here C^{-i} is the set of indices of the bootstrap samples b that do *not* contain observation i, and $|C^{-i}|$ is the number of such samples. In computing $\widehat{\text{Err}}^{(1)}$, we either have to choose B large enough to ensure that all of the $|C^{-i}|$ are greater than zero, or we can just leave out the terms in (7.56) corresponding to $|C^{-i}|$'s that are zero.

The leave-one out bootstrap solves the overfitting problem suffered by $\widehat{\text{Err}}_{\text{boot}}$, but has the training-set-size bias mentioned in the discussion of cross-validation. The average number of distinct observations in each bootstrap sample is about $0.632 \cdot N$, so its bias will roughly behave like that of twofold cross-validation. Thus if the learning curve has considerable slope at sample size $N/2$, the leave-one out bootstrap will be biased upward as an estimate of the true error.

The ".632 estimator" is designed to alleviate this bias. It is defined by

$$\widehat{\text{Err}}^{(.632)} = .368 \cdot \overline{\text{err}} + .632 \cdot \widehat{\text{Err}}^{(1)}. \tag{7.57}$$

The derivation of the .632 estimator is complex; intuitively it pulls the leave-one out bootstrap estimate down toward the training error rate, and hence reduces its upward bias. The use of the constant .632 relates to (7.55).

The .632 estimator works well in "light fitting" situations, but can break down in overfit ones. Here is an example due to Breiman et al. (1984). Suppose we have two equal-size classes, with the targets independent of the class labels, and we apply a one-nearest neighbor rule. Then $\overline{\text{err}} = 0$,

$\widehat{\text{Err}}^{(1)} = 0.5$ and so $\widehat{\text{Err}}^{(.632)} = .632 \times 0.5 = .316$. However, the true error rate is 0.5.

One can improve the .632 estimator by taking into account the amount of overfitting. First we define γ to be the *no-information error rate*: this is the error rate of our prediction rule if the inputs and class labels were independent. An estimate of γ is obtained by evaluating the prediction rule on all possible combinations of targets y_i and predictors $x_{i'}$

$$\hat{\gamma} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} L(y_i, \hat{f}(x_{i'})). \tag{7.58}$$

For example, consider the dichotomous classification problem: let $\hat{p}_1$ be the observed proportion of responses y_i equaling 1, and let $\hat{q}_1$ be the observed proportion of predictions $\hat{f}(x_{i'})$ equaling 1. Then

$$\hat{\gamma} = \hat{p}_1(1 - \hat{q}_1) + (1 - \hat{p}_1)\hat{q}_1. \tag{7.59}$$

With a rule like 1-nearest neighbors for which $\hat{q}_1 = \hat{p}_1$ the value of $\hat{\gamma}$ is $2\hat{p}_1(1-\hat{p}_1)$. The multi-category generalization of (7.59) is $\hat{\gamma} = \sum_\ell \hat{p}_\ell(1-\hat{q}_\ell)$.

Using this, the *relative overfitting rate* is defined to be

$$\hat{R} = \frac{\widehat{\text{Err}}^{(1)} - \overline{\text{err}}}{\hat{\gamma} - \overline{\text{err}}}, \tag{7.60}$$

a quantity that ranges from 0 if there is no overfitting ($\widehat{\text{Err}}^{(1)} = \overline{\text{err}}$) to 1 if the overfitting equals the no-information value $\hat{\gamma} - \overline{\text{err}}$. Finally, we define the ".632+" estimator by

$$\begin{aligned} \widehat{\text{Err}}^{(.632+)} &= (1 - \hat{w}) \cdot \overline{\text{err}} + \hat{w} \cdot \widehat{\text{Err}}^{(1)} \\ \text{with } \hat{w} &= \frac{.632}{1 - .368\hat{R}}. \end{aligned} \tag{7.61}$$

The weight w ranges from .632 if $\hat{R} = 0$ to 1 if $\hat{R} = 1$, so $\widehat{\text{Err}}^{(.632+)}$ ranges from $\widehat{\text{Err}}^{(.632)}$ to $\widehat{\text{Err}}^{(1)}$. Again, the derivation of (7.61) is complicated: roughly speaking, it produces a compromise between the leave-one-out bootstrap and the training error rate that depends on the amount of overfitting. For the 1-nearest-neighbor problem with class labels independent of the inputs, $\hat{w} = \hat{R} = 1$, so $\widehat{\text{Err}}^{(.632+)} = \widehat{\text{Err}}^{(1)}$, which has the correct expectation of 0.5. In other problems with less overfitting, $\widehat{\text{Err}}^{(.632+)}$ will lie somewhere between $\overline{\text{err}}$ and $\widehat{\text{Err}}^{(1)}$.

7.11.1 Example (Continued)

Figure 7.13 shows the results of tenfold cross-validation and the .632+ bootstrap estimate in the same four problems of Figures 7.7. As in that figure,