and worst possible model choices: $\min_\alpha \mathrm{Err}_{\mathcal{T}}(\alpha)$ and $\max_\alpha \mathrm{Err}_{\mathcal{T}}(\alpha)$. The boxplots show the distribution of the quantity

$$100 \times \frac{\mathrm{Err}_{\mathcal{T}}(\hat{\alpha}) - \min_\alpha \mathrm{Err}_{\mathcal{T}}(\alpha)}{\max_\alpha \mathrm{Err}_{\mathcal{T}}(\alpha) - \min_\alpha \mathrm{Err}_{\mathcal{T}}(\alpha)},$$

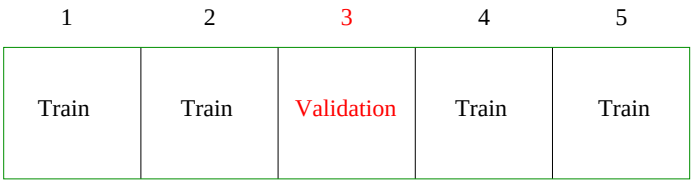
which represents the error in using the chosen model relative to the best model. For linear regression the model complexity was measured by the number of features; as mentioned in Section 7.5, this underestimates the df, since it does not *charge* for the search for the best model of that size. This was also used for the VC dimension of the linear classifier. For k-nearest neighbors, we used the quantity N/k. Under an additive-error regression model, this can be justified as the exact effective degrees of freedom (Exercise 7.6); we do not know if it corresponds to the VC dimension. We used $a_1 = a_2 = 1$ for the constants in (7.46); the results for SRM changed with different constants, and this choice gave the most favorable results. We repeated the SRM selection using the alternative practical bound (7.47), and got almost identical results. For misclassification error we used $\hat{\sigma_\varepsilon}^2 = [N/(N-d)] \cdot \overline{\mathrm{err}}(\alpha)$ for the least restrictive model ($k = 5$ for KNN, since $k = 1$ results in zero training error). The AIC criterion seems to work well in all four scenarios, despite the lack of theoretical support with 0–1 loss. BIC does nearly as well, while the performance of SRM is mixed.

7.10 Cross-Validation

Probably the simplest and most widely used method for estimating prediction error is cross-validation. This method directly estimates the expected extra-sample error $\mathrm{Err} = \mathrm{E}[L(Y, \hat{f}(X))]$, the average generalization error when the method $\hat{f}(X)$ is applied to an independent test sample from the joint distribution of X and Y. As mentioned earlier, we might hope that cross-validation estimates the conditional error, with the training set $\mathcal{T}$ held fixed. But as we will see in Section 7.12, cross-validation typically estimates well only the expected prediction error.

7.10.1 K-Fold Cross-Validation

Ideally, if we had enough data, we would set aside a validation set and use it to assess the performance of our prediction model. Since data are often scarce, this is usually not possible. To finesse the problem, K-fold cross-validation uses part of the available data to fit the model, and a different part to test it. We split the data into K roughly equal-sized parts; for example, when $K = 5$, the scenario looks like this:



For the kth part (third above), we fit the model to the other $K-1$ parts of the data, and calculate the prediction error of the fitted model when predicting the kth part of the data. We do this for $k = 1, 2, \ldots, K$ and combine the K estimates of prediction error.

Here are more details. Let $\kappa : \{1, \ldots, N\} \mapsto \{1, \ldots, K\}$ be an indexing function that indicates the partition to which observation i is allocated by the randomization. Denote by $\hat{f}^{-k}(x)$ the fitted function, computed with the kth part of the data removed. Then the cross-validation estimate of prediction error is

$$\mathrm{CV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i)). \tag{7.48}$$

Typical choices of K are 5 or 10 (see below). The case $K = N$ is known as *leave-one-out* cross-validation. In this case $\kappa(i) = i$, and for the ith observation the fit is computed using all the data except the ith.

Given a set of models $f(x, \alpha)$ indexed by a tuning parameter α, denote by $\hat{f}^{-k}(x, \alpha)$ the αth model fit with the kth part of the data removed. Then for this set of models we define

$$\mathrm{CV}(\hat{f}, \alpha) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i, \alpha)). \tag{7.49}$$

The function $\mathrm{CV}(\hat{f}, \alpha)$ provides an estimate of the test error curve, and we find the tuning parameter $\hat{\alpha}$ that minimizes it. Our final chosen model is $f(x, \hat{\alpha})$, which we then fit to all the data.

It is interesting to wonder about what quantity K-fold cross-validation estimates. With $K = 5$ or 10, we might guess that it estimates the expected error Err, since the training sets in each fold are quite different from the original training set. On the other hand, if $K = N$ we might guess that cross-validation estimates the conditional error $\mathrm{Err}_{\mathcal{T}}$. It turns out that cross-validation only estimates effectively the average error Err, as discussed in Section 7.12.

What value should we choose for K? With $K = N$, the cross-validation estimator is approximately unbiased for the true (expected) prediction error, but can have high variance because the N "training sets" are so similar to one another. The computational burden is also considerable, requiring N applications of the learning method. In certain special problems, this computation can be done quickly—see Exercises 7.3 and 5.13.

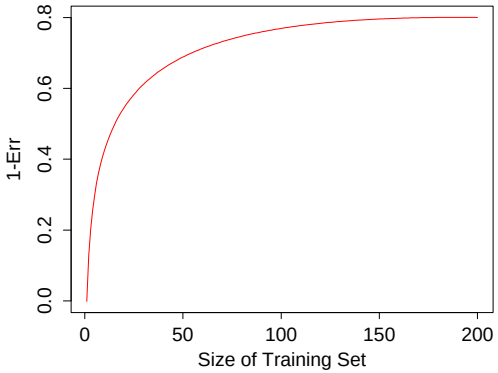


FIGURE 7.8. *Hypothetical learning curve for a classifier on a given task: a plot of* $1 - \text{Err}$ *versus the size of the training set* N*. With a dataset of* 200 *observations,* 5*-fold cross-validation would use training sets of size* 160*, which would behave much like the full set. However, with a dataset of* 50 *observations fivefold cross-validation would use training sets of size* 40*, and this would result in a considerable overestimate of prediction error.*

On the other hand, with $K = 5$ say, cross-validation has lower variance. But bias could be a problem, depending on how the performance of the learning method varies with the size of the training set. Figure 7.8 shows a hypothetical "learning curve" for a classifier on a given task, a plot of $1 - \text{Err}$ versus the size of the training set N. The performance of the classifier improves as the training set size increases to 100 observations; increasing the number further to 200 brings only a small benefit. If our training set had 200 observations, fivefold cross-validation would estimate the performance of our classifier over training sets of size 160, which from Figure 7.8 is virtually the same as the performance for training set size 200. Thus cross-validation would not suffer from much bias. However if the training set had 50 observations, fivefold cross-validation would estimate the performance of our classifier over training sets of size 40, and from the figure that would be an underestimate of $1 - \text{Err}$. Hence as an estimate of Err, cross-validation would be biased upward.

To summarize, if the learning curve has a considerable slope at the given training set size, five- or tenfold cross-validation will overestimate the true prediction error. Whether this bias is a drawback in practice depends on the objective. On the other hand, leave-one-out cross-validation has low bias but can have high variance. Overall, five- or tenfold cross-validation are recommended as a good compromise: see Breiman and Spector (1992) and Kohavi (1995).

Figure 7.9 shows the prediction error and tenfold cross-validation curve estimated from a single training set, from the scenario in the bottom right panel of Figure 7.3. This is a two-class classification problem, using a lin-

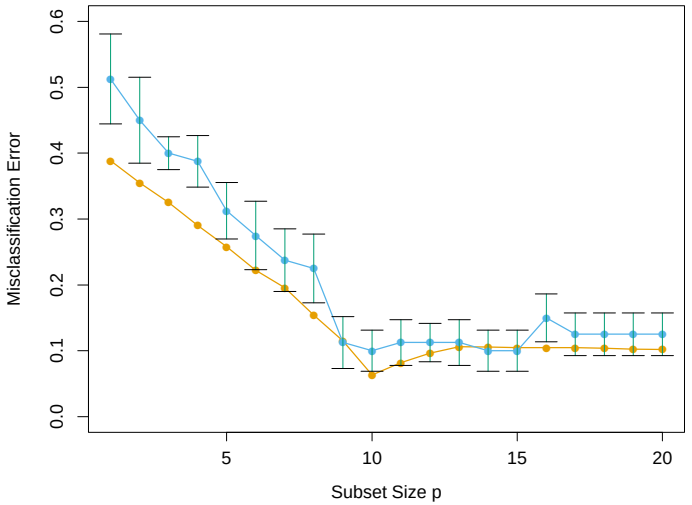


FIGURE 7.9. *Prediction error (orange) and tenfold cross-validation curve (blue) estimated from a single training set, from the scenario in the bottom right panel of Figure 7.3.*

ear model with best subsets regression of subset size p. Standard error bars are shown, which are the standard errors of the individual misclassification error rates for each of the ten parts. Both curves have minima at $p = 10$, although the CV curve is rather flat beyond 10. Often a "one-standard error" rule is used with cross-validation, in which we choose the most parsimonious model whose error is no more than one standard error above the error of the best model. Here it looks like a model with about $p = 9$ predictors would be chosen, while the true model uses $p = 10$.

Generalized cross-validation provides a convenient approximation to leave-one out cross-validation, for linear fitting under squared-error loss. As defined in Section 7.6, a linear fitting method is one for which we can write

$$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}. \tag{7.50}$$

Now for many linear fitting methods,

$$\frac{1}{N}\sum_{i=1}^{N}[y_i - \hat{f}^{-i}(x_i)]^2 \quad = \quad \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}\right]^2, \tag{7.51}$$

where S_{ii} is the ith diagonal element of $\mathbf{S}$ (see Exercise 7.3). The GCV approximation is

$$\mathrm{GCV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - \mathrm{trace}(\mathbf{S})/N}\right]^2. \tag{7.52}$$

The quantity trace($\mathbf{S}$) is the effective number of parameters, as defined in Section 7.6.

GCV can have a computational advantage in some settings, where the trace of $\mathbf{S}$ can be computed more easily than the individual elements S_{ii}. In smoothing problems, GCV can also alleviate the tendency of cross-validation to undersmooth. The similarity between GCV and AIC can be seen from the approximation $1/(1-x)^2 \approx 1+2x$ (Exercise 7.7).

7.10.2 The Wrong and Right Way to Do Cross-validation

Consider a classification problem with a large number of predictors, as may arise, for example, in genomic or proteomic applications. A typical strategy for analysis might be as follows:

1. Screen the predictors: find a subset of "good" predictors that show fairly strong (univariate) correlation with the class labels
2. Using just this subset of predictors, build a multivariate classifier.
3. Use cross-validation to estimate the unknown tuning parameters and to estimate the prediction error of the final model.

Is this a correct application of cross-validation? Consider a scenario with $N = 50$ samples in two equal-sized classes, and $p = 5000$ quantitative predictors (standard Gaussian) that are independent of the class labels. The true (test) error rate of any classifier is 50%. We carried out the above recipe, choosing in step (1) the 100 predictors having highest correlation with the class labels, and then using a 1-nearest neighbor classifier, based on just these 100 predictors, in step (2). Over 50 simulations from this setting, the average CV error rate was 3%. This is far lower than the true error rate of 50%.

What has happened? The problem is that the predictors have an unfair advantage, as they were chosen in step (1) on the basis of *all of the samples*. Leaving samples out *after* the variables have been selected does not correctly mimic the application of the classifier to a completely independent test set, since these predictors "have already seen" the left out samples.

Figure 7.10 (top panel) illustrates the problem. We selected the 100 predictors having largest correlation with the class labels over all 50 samples. Then we chose a random set of 10 samples, as we would do in five-fold cross-validation, and computed the correlations of the pre-selected 100 predictors with the class labels over just these 10 samples (top panel). We see that the correlations average about 0.28, rather than 0, as one might expect.

Here is the correct way to carry out cross-validation in this example:

1. Divide the samples into K cross-validation folds (groups) at random.
2. For each fold $k = 1, 2, \ldots, K$

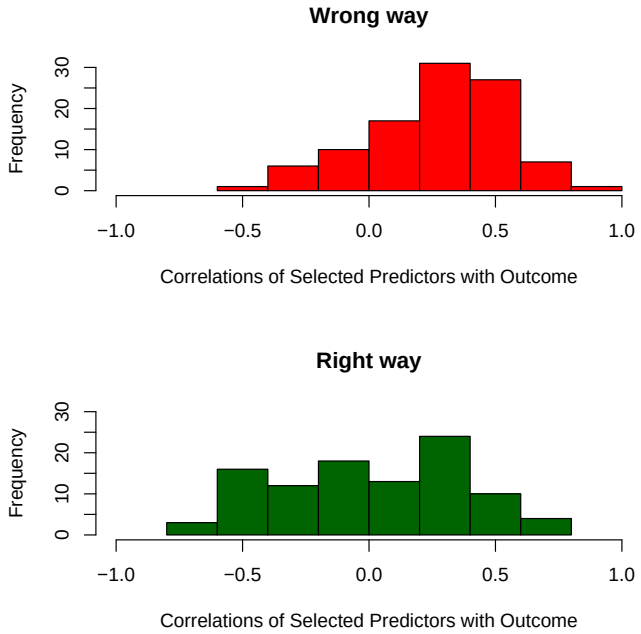


FIGURE 7.10. *Cross-validation the wrong and right way: histograms shows the correlation of class labels, in* 10 *randomly chosen samples, with the* 100 *predictors chosen using the incorrect (upper red) and correct (lower green) versions of cross-validation.*

(a) Find a subset of "good" predictors that show fairly strong (univariate) correlation with the class labels, using all of the samples except those in fold k.

(b) Using just this subset of predictors, build a multivariate classifier, using all of the samples except those in fold k.

(c) Use the classifier to predict the class labels for the samples in fold k.

The error estimates from step 2(c) are then accumulated over all K folds, to produce the cross-validation estimate of prediction error. The lower panel of Figure 7.10 shows the correlations of class labels with the 100 predictors chosen in step 2(a) of the correct procedure, over the samples in a typical fold k. We see that they average about zero, as they should.

In general, with a multistep modeling procedure, cross-validation must be applied to the entire sequence of modeling steps. In particular, samples must be "left out" before any selection or filtering steps are applied. There is one qualification: initial *unsupervised* screening steps can be done before samples are left out. For example, we could select the 1000 predictors

with highest variance across all 50 samples, before starting cross-validation. Since this filtering does not involve the class labels, it does not give the predictors an unfair advantage.

While this point may seem obvious to the reader, we have seen this blunder committed many times in published papers in top rank journals. With the large numbers of predictors that are so common in genomic and other areas, the potential consequences of this error have also increased dramatically; see Ambroise and McLachlan (2002) for a detailed discussion of this issue.

7.10.3 Does Cross-Validation Really Work?

We once again examine the behavior of cross-validation in a high-dimensional classification problem. Consider a scenario with $N = 20$ samples in two equal-sized classes, and $p = 500$ quantitative predictors that are independent of the class labels. Once again, the true error rate of any classifier is 50%. Consider a simple univariate classifier: a single split that minimizes the misclassification error (a "stump"). Stumps are trees with a single split, and are used in boosting methods (Chapter 10). A simple argument suggests that cross-validation will not work properly in this setting[2]:

> *Fitting to the entire training set, we will find a predictor that splits the data very well. If we do 5-fold cross-validation, this same predictor should split any* 4/5*ths and* 1/5*th of the data well too, and hence its cross-validation error will be small (much less than 50%.) Thus CV does not give an accurate estimate of error.*

To investigate whether this argument is correct, Figure 7.11 shows the result of a simulation from this setting. There are 500 predictors and 20 samples, in each of two equal-sized classes, with all predictors having a standard Gaussian distribution. The panel in the top left shows the number of training errors for each of the 500 stumps fit to the training data. We have marked in color the six predictors yielding the fewest errors. In the top right panel, the training errors are shown for stumps fit to a random 4/5ths partition of the data (16 samples), and tested on the remaining 1/5th (four samples). The colored points indicate the same predictors marked in the top left panel. We see that the stump for the blue predictor (whose stump was the best in the top left panel), makes two out of four test errors (50%), and is no better than random.

What has happened? The preceding argument has ignored the fact that in cross-validation, the model must be completely retrained for each fold

[2]This argument was made to us by a scientist at a proteomics lab meeting, and led to material in this section.

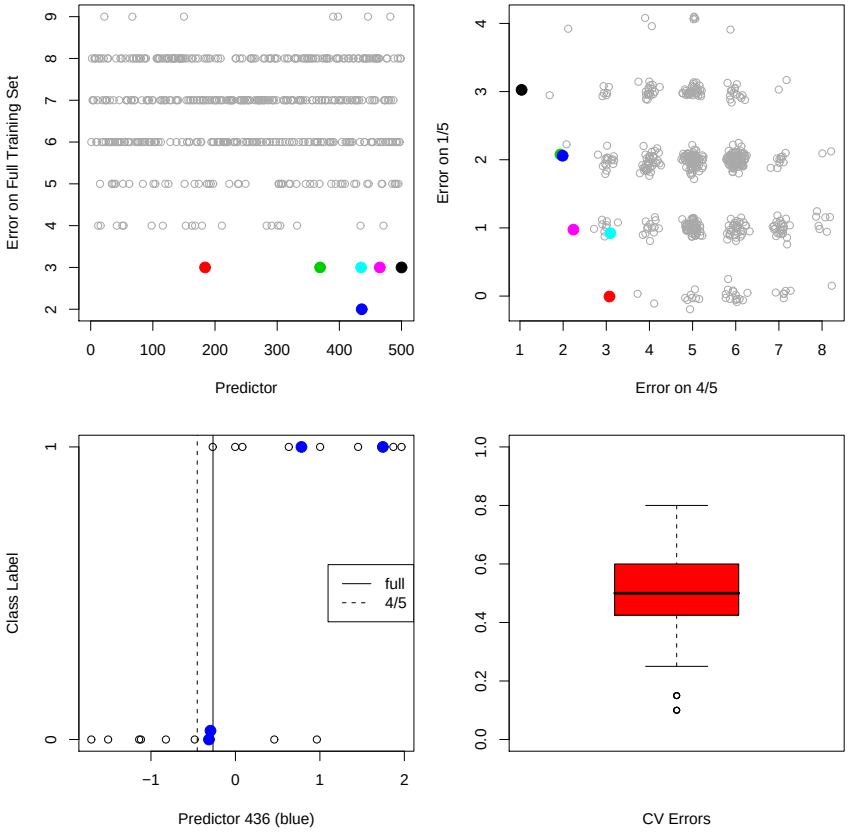


FIGURE 7.11. *Simulation study to investigate the performance of cross validation in a high-dimensional problem where the predictors are independent of the class labels. The top-left panel shows the number of errors made by individual stump classifiers on the full training set (*20 *observations). The top right panel shows the errors made by individual stumps trained on a random split of the dataset into* 4/5*ths (*16 *observations) and tested on the remaining* 1/5*th (*4 *observations). The best performers are depicted by colored dots in each panel. The bottom left panel shows the effect of re-estimating the split point in each fold: the colored points correspond to the four samples in the* 1/5*th validation set. The split point derived from the full dataset classifies all four samples correctly, but when the split point is re-estimated on the* 4/5*ths data (as it should be), it commits two errors on the four validation samples. In the bottom right we see the overall result of five-fold cross-validation applied to* 50 *simulated datasets. The average error rate is about* 50%*, as it should be.*

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,

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