7

Model Assessment and Selection

7.1 Introduction

The *generalization* performance of a learning method relates to its prediction capability on independent test data. Assessment of this performance is extremely important in practice, since it guides the choice of learning method or model, and gives us a measure of the quality of the ultimately chosen model.

In this chapter we describe and illustrate the key methods for performance assessment, and show how they are used to select models. We begin the chapter with a discussion of the interplay between bias, variance and model complexity.

7.2 Bias, Variance and Model Complexity

Figure 7.1 illustrates the important issue in assessing the ability of a learning method to generalize. This is the same as Figure 2.11; because it is so important, we display it here again. Consider first the case of a quantitative or interval scale response. We have a target variable Y, a vector of inputs X, and a prediction model $\hat{f}(X)$ that has been estimated from a training sample. The loss function for measuring errors between Y and

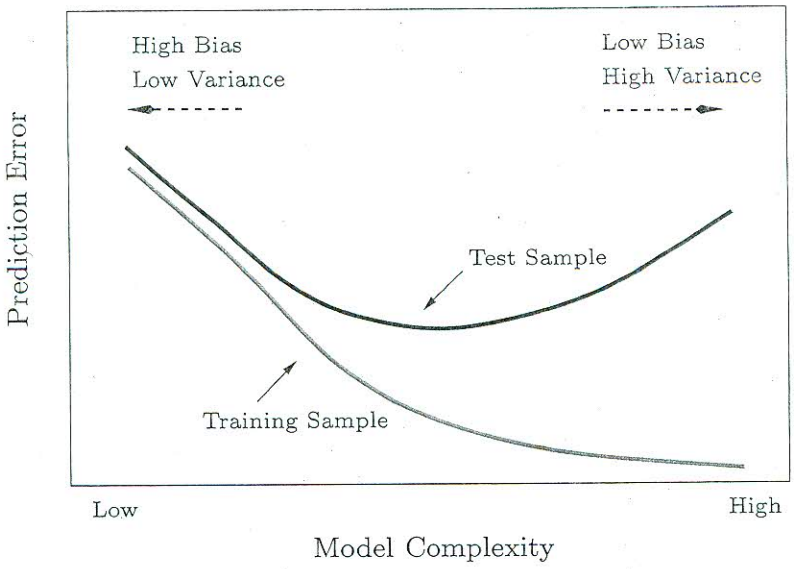

FIGURE 7.1. *Behavior of test sample and training sample error as the model complexity is varied.*

$\hat{f}(X)$ is denoted by $L(Y, \hat{f}(X))$. Typical choices are

$$L(Y, \hat{f}(X)) = \begin{cases} (Y - \hat{f}(X))^2 & \text{squared error} \\ |Y - \hat{f}(X)| & \text{absolute error.} \end{cases} \tag{7.1}$$

Test error, also referred to as *generalization error*, is the expected prediction error over an independent test sample

$$\text{Err} = \text{E}[L(Y, \hat{f}(X))], \tag{7.2}$$

where both X and Y are drawn randomly from their joint distribution (population). Note that this expectation averages anything that is random, including the randomness in the training sample that produced $\hat{f}$. *Training error* is the average loss over the training sample

$$\overline{\text{err}} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i)). \tag{7.3}$$

We would like to know the test error of our estimated model $\hat{f}$. As the model becomes more and more complex, it is able to adapt to more complicated underlying structures (a decrease in bias), but the estimation error increases (an increase in variance). In between there is an optimal model complexity that gives minimum test error.

Unfortunately training error is not a good estimate of the test error, as seen in Figure 7.1. Training error consistently decreases with model complexity, typically dropping to zero if we increase the model complexity enough. However, a model with zero training error is overfit to the training data and will typically generalize poorly.

The story is similar for a qualitative or categorical response G taking one of K values in a set $\mathcal{G}$, labelled for convenience as $1, 2, \ldots, K$. Typically we model the probabilities $p_k(X) = \Pr(G = k|X)$ (or some monotone transformations $f_k(X)$), and then $\hat{G}(X) = \arg\max_k \hat{p}_k(X)$. In some cases, such as 1-nearest neighbor classification (Chapters 2 and 13) we produce $\hat{G}(X)$ directly. Typical loss functions are

$$\begin{aligned} L(G, \hat{G}(X)) &= I(G \neq \hat{G}(X)) \quad \text{0–1 loss,} && (7.4) \\ L(G, \hat{p}(X)) &= -2 \sum_{k=1}^{K} I(G = k) \log \hat{p}_k(X) \\ &= -2 \log \hat{p}_G(X) \quad \text{log-likelihood.} && (7.5) \end{aligned}$$

The log-likelihood is sometimes referred to as *cross-entropy* loss or *deviance*.

Again, test error is given by $\text{Err} = \text{E}[L(G, \hat{G}(X))]$, the expected misclassification rate, or $\text{Err} = \text{E}[L(G, \hat{p}(X))]$. Training error is the sample analogue, for example,

$$\overline{\text{err}} = \frac{-2}{N} \sum_{i=1}^{N} \log \hat{p}_{g_i}(x_i), \tag{7.6}$$

the sample log-likelihood for the model.

The log-likelihood can be used as a loss-function for general response densities, such as the Poisson, gamma, exponential, log-normal and others. If $\Pr_{\theta(X)}(Y)$ is the density of Y, indexed by a parameter $\theta(X)$ that depends on the predictor X, then

$$L(Y, \theta(X)) = -2 \cdot \log \Pr_{\theta(X)}(Y). \tag{7.7}$$

The "2" in the definition makes the log-likelihood loss for the Gaussian distribution match squared-error loss.

For ease of exposition, for the remainder of this chapter we will use Y and $f(X)$ to represent all of the above situations, since we focus mainly on the quantitative response (squared-error loss) setting. For the other situations, the appropriate translations are obvious.

In this chapter we describe a number of methods for estimating the test error curve for a model. Typically our model will have a tuning parameter or parameters α and so we can write our predictions as $\hat{f}_\alpha(x)$. The tuning parameter varies the complexity of our model, and we wish to find the value of α that minimizes error, that is, produces the minimum of the test error curve in Figure 7.1. Having said this, for brevity we will often suppress the dependence of $\hat{f}(x)$ on α.

It is important to note that there are in fact two separate goals that we might have in mind:

Model selection: estimating the performance of different models in order to choose the (approximate) best one.

Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.

If we are in a data-rich situation, the best approach for both problems is to randomly divide the dataset into three parts: a training set, a validation set, and a test set. The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model. Ideally, the test set should be kept in a "vault," and be brought out only at the end of the data analysis. Suppose instead that we use the test set repeatedly, choosing the model with smallest test set error. Then the test set error of the final chosen model will underestimate the true test error, sometimes substantially.

It is difficult to give a general rule on how to choose the number of observations in each of the three parts, as this depends on the signal-to-noise ratio in the data and the training sample size. A typical split might be 50% for training, and 25% each for validation and testing:

The methods in this chapter are designed for situations where there is insufficient data to split it into three parts. Again it is too difficult to give a general rule on how much training data is enough; among other things, this depends on the signal-to-noise ratio of the underlying function, and the complexity of the models being fit to the data.

The methods of this chapter approximate the validation step either analytically (AIC, BIC, MDL, SRM) or by efficient sample re-use (cross-validation and the bootstrap). Besides their use in model selection, we also examine to what extent each method provides a reliable estimate of test error of the final chosen model.

Before jumping into these topics, we first explore in more detail the nature of test error and the bias–variance tradeoff.

7.3 The Bias–Variance Decomposition

As in Chapter 2, if we assume that $Y = f(X) + \varepsilon$ where $\mathrm{E}(\varepsilon) = 0$ and $\mathrm{Var}(\varepsilon) = \sigma_\varepsilon^2$, we can derive an expression for the expected prediction error

of a regression fit $\hat{f}(X)$ at an input point $X = x_0$, using squared-error loss:

$$\begin{aligned}\text{Err}(x_0) &= E[(Y - \hat{f}(x_0))^2 | X = x_0] \\ &= \sigma_\varepsilon^2 + [E\hat{f}(x_0) - f(x_0)]^2 + E[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\ &= \sigma_\varepsilon^2 + \text{Bias}^2(\hat{f}(x_0)) + \text{Var}(\hat{f}(x_0)) \\ &= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}. \end{aligned} \tag{7.8}$$

The first term is the variance of the target around its true mean $f(x_0)$, and cannot be avoided no matter how well we estimate $f(x_0)$, unless $\sigma_\varepsilon^2 = 0$. The second term is the squared bias, the amount by which the average of our estimate differs from the true mean; the last term is the variance; the expected squared deviation of $\hat{f}(x_0)$ around its mean. Typically the more complex we make the model $\hat{f}$, the lower the (squared) bias but the higher the variance.

For the k-nearest-neighbor regression fit, these expressions have the simple form

$$\begin{aligned}\text{Err}(x_0) &= E[(Y - \hat{f}_k(x_0))^2 | X = x_0] \\ &= \sigma_\varepsilon^2 + \left[f(x_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(x_{(\ell)})\right]^2 + \sigma_\varepsilon^2/k. \end{aligned} \tag{7.9}$$

Here we assume for simplicity that training inputs x_i are fixed, and the randomness arises from the y_i. The number of neighbors k is inversely related to the model complexity. For small k, the estimate $\hat{f}_k(x)$ can potentially adapt itself better to the underlying $f(x)$. As we increase k, the bias—the squared difference between $f(x_0)$ and the average of $f(x)$ at the k-nearest neighbors—will typically increase, while the variance decreases.

For a linear model fit $\hat{f}_p(x) = \hat{\beta}^T x$, where the parameter vector β with p components is fit by least squares, we have

$$\begin{aligned}\text{Err}(x_0) &= E[(Y - \hat{f}_p(x_0))^2 | X = x_0] \\ &= \sigma_\varepsilon^2 + [f(x_0) - E\hat{f}_p(x_0)]^2 + ||\mathbf{h}(x_0)||^2 \sigma_\varepsilon^2. \end{aligned} \tag{7.10}$$

Here $\mathbf{h}(x_0)$ is the N-vector of linear weights that produce the fit $\hat{f}_p(x_0) = x_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$, and hence $\text{Var}[\hat{f}_p(x_0)] = ||\mathbf{h}(x_0)||^2\sigma_\varepsilon^2$. While this variance changes with x_0, its average (over the sample values x_i) is $(p/N)\sigma_\varepsilon^2$, and hence

$$\frac{1}{N}\sum_{i=1}^{N} \text{Err}(x_i) = \sigma_\varepsilon^2 + \frac{1}{N}\sum_{i=1}^{N}[f(x_i) - E\hat{f}(x_i)]^2 + \frac{p}{N}\sigma_\varepsilon^2, \tag{7.11}$$

the *in-sample* error. Here model complexity is directly related to the number of parameters p.

The test error $\mathrm{Err}(x_0)$ for a ridge regression fit $\hat{f}_\alpha(x_0)$ is identical in form to (7.10), except the linear weights in the variance term are different: $\mathbf{h}(x_0) = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I})^{-1}x_0$. The bias term will also be different.

For a linear model family such as ridge regression, we can break down the bias more finely. Let β_* denote the parameters of the best-fitting linear approximation to f:

$$\beta_* = \arg\min_{\beta} \mathrm{E}\left(f(X) - \beta^T X\right)^2 . \tag{7.12}$$

Here the expectation is taken with respect to the distribution of the input variables X. Then we can write the average squared bias as

$$\begin{aligned} E_{x_0}\left[f(x_0) - E\hat{f}_\alpha(x_0)\right]^2 &= E_{x_0}\left[f(x_0) - \beta_*^T x_0\right]^2 + E_{x_0}\left[\beta_*^T x_0 - \mathrm{E}\hat{\beta}_\alpha^T x_0\right]^2 \\ &= \text{Ave[Model Bias]}^2 + \text{Ave[Estimation Bias]}^2 \end{aligned} \tag{7.13}$$

The first term on the right-hand side is the average squared *model bias*, the error between the best-fitting linear approximation and the true function. The second term is the average squared *estimation bias*, the error between the average estimate $\mathrm{E}(\hat{\beta}^T x_0)$ and the best fitting linear approximation.

For linear models fit by ordinary least squares, the estimation bias is zero. For restricted fits, such as ridge regression, it is positive, and we trade it off with the benefits of a reduced variance. The model bias can only be reduced by enlarging the class of linear models to a richer collection of models, by including interactions and transformations of the variables in the model.

Figure 7.2 shows the bias–variance tradeoff schematically. In the case of linear models, the model space is the set of all linear predictions from p inputs and the black dot labeled "closest fit" is $\beta_*^T x$. The blue-shaded region indicates the error σ_ε with which we see the truth in the training sample.

Also shown is the variance of the least squares fit, indicated by the large yellow circle centered at the black dot labelled "closest fit in population'. Now if we were to fit a model with fewer predictors, or regularize the coefficients by shrinking them toward zero (say), we would get the "shrunken fit" shown in the figure. This fit has an additional estimation bias, due to the fact that it is not the closest fit in the model space. On the other hand, it has smaller variance. If the decrease in variance exceeds the increase in (squared) bias, then this is worthwhile.

7.3.1 *Example: Bias–Variance Tradeoff*

Figure 7.3 shows the bias–variance tradeoff for two simulated examples. There are 50 observations and 20 predictors, uniformly distributed in the hypercube $[0, 1]^{20}$. The situations are as follows:

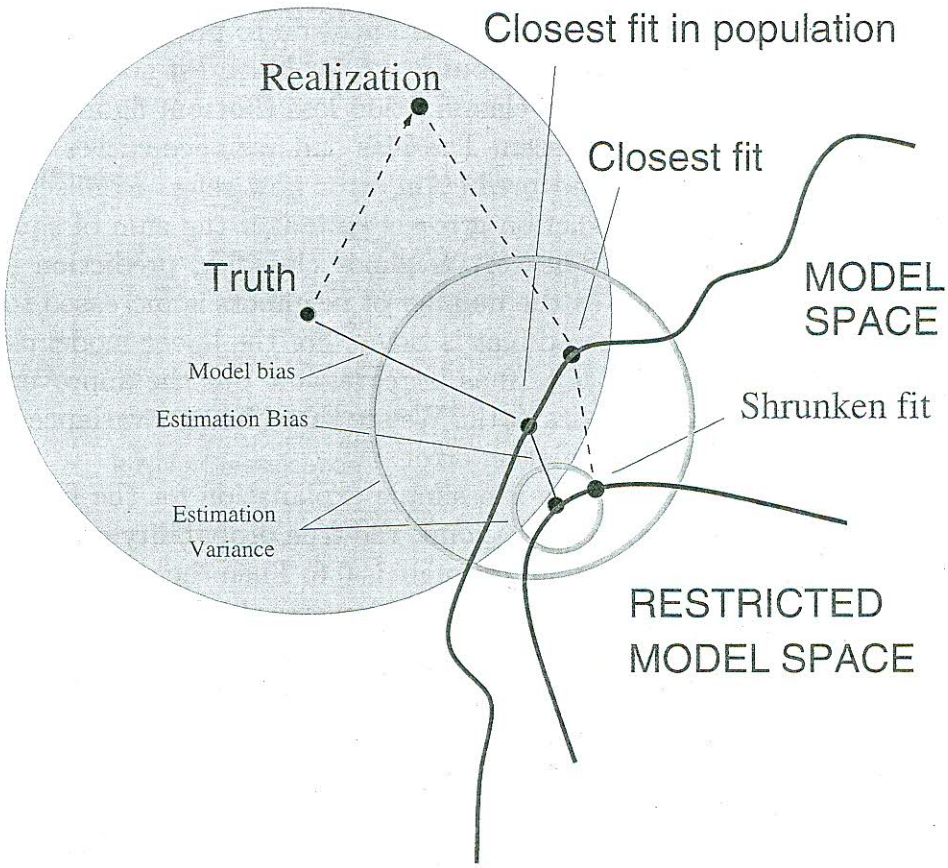

FIGURE 7.2. *Schematic of the behavior of bias and variance. The model space is the set of all possible predictions from the model, with the "closest fit" labeled with a black dot. The model bias from the truth is shown, along with the variance, indicated by the large yellow circle centered at the black dot labelled "closest fit in population". A shrunken or regularized fit is also shown, having additional estimation bias, but smaller prediction error due to its decreased variance.*

Left panels: Y is 0 if $X_1 \leq 1/2$ and 1 if $X_1 > 1/2$, and we apply k-nearest neighbors.

Right panels: Y is 1 if $\sum_{j=1}^{10} X_j$ is greater than 5 and 0 otherwise, and we use best subset linear regression of size p.

The top row is regression with squared error loss; the bottom row is classification with 0–1 loss. The figures show the prediction error (red), squared bias (green) and variance (blue), all computed for a large test sample.

In the regression problems, bias and variance add to produce the prediction error curves, with minima at about $k = 5$ for k-nearest neighbors, and $p \geq 10$ for the linear model. For classification loss (bottom figures), some interesting phenomena can be seen. The bias and variance curves are the same as in the top figures, and prediction error now refers to misclassification rate. We see that prediction error is no longer the sum of squared bias and variance. For the k-nearest neighbor classifier, prediction error decreases or stays the same as the number of neighbors is increased to 20, despite the fact that the squared bias is rising. For the linear model classifier the minimum occurs for $p \geq 10$ as in regression, but the improvement over the $p = 1$ model is more dramatic. We see that bias and variance seem to interact in determining prediction error.

Why does this happen? There is a simple explanation for the first phenomenon. Suppose at a given input point, the true probability of class 1 is 0.9 while the expected value of our estimate is 0.6. Then the squared bias—$(0.6 - 0.9)^2$—is considerable, but the prediction error is zero since we make the correct decision. In other words, estimation errors that leave us on the right side of the decision boundary don't hurt. Exercise 7.2 demonstrates this phenomenon analytically, and also shows the interaction effect between bias and variance.

The overall point is that the bias–variance tradeoff behaves differently for 0–1 loss than it does for squared error loss. This in turn means that the best choices of tuning parameters may differ substantially in the two settings. One should base the choice of tuning parameter on an estimate of prediction error, as described in the following sections.

7.4 Optimism of the Training Error Rate

Typically, the training error rate

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i)) \tag{7.14}$$

will be less than the true error $\text{Err} = \text{E}[L(Y, \hat{f}(X))]$, because the same data is being used to fit the method and assess its error. A fitting method

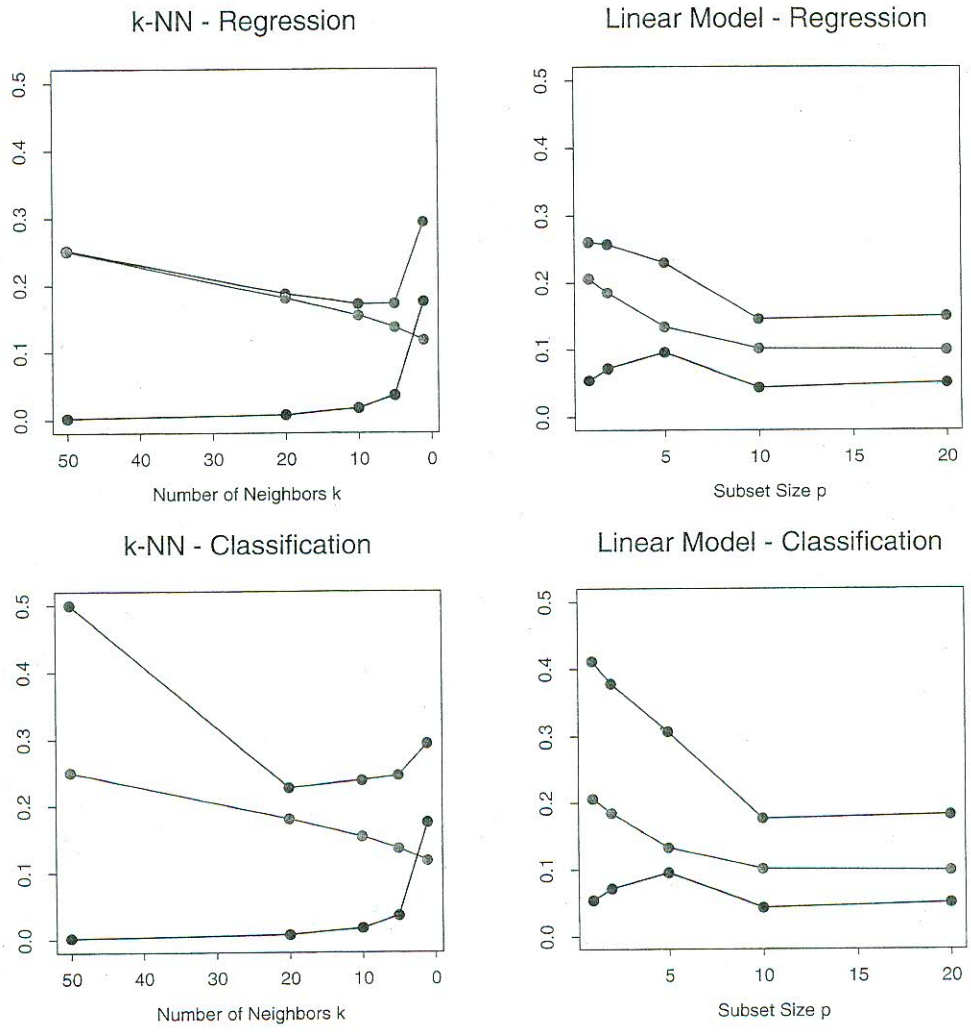

FIGURE 7.3. *Prediction error (red), squared bias (green) and variance (blue) for a simulated example. The top row is regression with squared error loss; the bottom row is classification with* 0–1 *loss. The models are k-nearest neighbors (left) and best subset regression of size p (right). The variance and bias curves are the same in regression and classification, but the prediction error curve is different.*

typically adapts to the training data, and hence the apparent or training error $\overline{\text{err}}$ will be an overly optimistic estimate of the generalization error Err.

Part of the discrepancy is due to where the evaluation points occur. Err is a kind of *extra-sample* error, since the test feature vectors don't need to coincide with the training feature vectors. The nature of the optimism in $\overline{\text{err}}$ is easiest to understand when we focus not on Err but on the *in-sample* error

$$\text{Err}_{\text{in}} = \frac{1}{N}\sum_{i=1}^{N} \text{E}_{\mathbf{y}}\text{E}_{Y^{\text{new}}} L(Y_i^{\text{new}}, \hat{f}(x_i)). \tag{7.15}$$

The Y^{new} notation indicates that we observe N new response values at each of the training points x_i, $i = 1, 2, \ldots, N$. We define the *optimism* as the expected difference between Err_{in} and the training error $\overline{\text{err}}$:

$$\text{op} \equiv \text{Err}_{\text{in}} - \text{E}_{\mathbf{y}}(\overline{\text{err}}). \tag{7.16}$$

This is typically positive since $\overline{\text{err}}$ is usually biased downward as an estimate of prediction error.

For squared error, 0–1, and other loss functions, one can show quite generally that

$$\text{op} = \frac{2}{N}\sum_{i=1}^{N} \text{Cov}(\hat{y}_i, y_i), \tag{7.17}$$

where Cov indicates covariance. Thus the amount by which $\overline{\text{err}}$ underestimates the true error depends on how strongly y_i affects its own prediction. The harder we fit the data, the greater $\text{Cov}(\hat{y}_i, y_i)$ will be, thereby increasing the optimism. Exercise 7.4 proves this result for squared error loss where $\hat{y}_i$ is the fitted value from the regression. For 0–1 loss, $\hat{y}_i \in \{0, 1\}$ is the classification at x_i, and for entropy loss, $\hat{y}_i \in [0, 1]$ is the fitted probability of class 1 at x_i.

In summary, we have the important relation

$$\text{Err}_{\text{in}} = \text{E}_{\mathbf{y}}(\overline{\text{err}}) + \frac{2}{N}\sum_{i=1}^{N} \text{Cov}(\hat{y}_i, y_i). \tag{7.18}$$

This expression simplifies if $\hat{y}_i$ is obtained by a linear fit with d inputs or basis functions. For example,

$$\sum_{i=1}^{N} \text{Cov}(\hat{y}_i, y_i) = d\sigma_\varepsilon^2 \tag{7.19}$$

for the additive error model $Y = f(X) + \varepsilon$, and so

$$\text{Err}_{\text{in}} = \text{E}_{\mathbf{y}}\overline{\text{err}} + 2 \cdot \frac{d}{N}\sigma_\varepsilon^2. \tag{7.20}$$

The optimism increases linearly with the number d of inputs or basis functions we use, but decreases as the training sample size increases. Versions of (7.20) hold approximately for other error models, such as binary data and entropy loss.

An obvious way to estimate prediction error is to estimate the optimism and then add it to the training error rate $\overline{\text{err}}$. The methods described in the next section—AIC, BIC and others—work in this way, for a special class of estimates that are linear in their parameters.

In contrast, the cross-validation and bootstrap methods, described later in the chapter, are direct estimates of the extra-sample error Err. These general tools can be used with any loss function, and with nonlinear, adaptive fitting techniques.

In-sample error is not usually of direct interest since future values of the features are not likely to coincide with with their training set values. But for comparison between models, in-sample error is convenient and often leads to effective model selection. The reason is that the relative (not absolute) size of the error is what matters.

7.5 Estimates of In-Sample Prediction Error

The general form of the in-sample estimates is

$$\widehat{\text{Err}}_{\text{in}} = \overline{\text{err}} + \widehat{\text{op}}, \tag{7.21}$$

where $\widehat{\text{op}}$ is an estimate of the optimism.

Using expression (7.20), applicable when d parameters are fit under squared error loss, leads to the so-called C_p statistic,

$$C_p = \overline{\text{err}} + 2 \cdot \frac{d}{N} \hat{\sigma}_\varepsilon^2. \tag{7.22}$$

Here $\hat{\sigma}_\varepsilon^2$ is an estimate of the noise variance, obtained from the mean-squared error of a low-bias model. Using this criterion we adjust the training error by a factor proportional to the number of basis functions used.

The *Akaike information criterion* is a similar but more generally applicable estimate of Err_{in} when a log-likelihood loss function is used. It relies on a relationship similar to (7.20) that holds asymptotically as $N \to \infty$:

$$-2 \cdot \text{E}[\log \Pr_{\hat{\theta}}(Y)] \approx -\frac{2}{N} \cdot \text{E}[\text{loglik}] + 2 \cdot \frac{d}{N}. \tag{7.23}$$

Here $\Pr_\theta(Y)$ is a family of densities for Y (containing the "true" density), $\hat{\theta}$ is the maximum-likelihood estimate of θ, and "loglik" is the maximized log-likelihood:

$$\text{loglik} = \sum_{i=1}^{N} \log \Pr_{\hat{\theta}}(y_i). \tag{7.24}$$

For example, for the logistic regression model, using the binomial log-likelihood, we have

$$\text{AIC} = -\frac{2}{N} \cdot \text{loglik} + 2 \cdot \frac{d}{N}. \tag{7.25}$$

For the Gaussian model (with variance $\sigma_\varepsilon^2 = \hat{\sigma}_\varepsilon^2$ assumed known), the AIC statistic is equivalent to C_p, and so we refer to them collectively as AIC.

To use AIC for model selection, we simply choose the model giving smallest AIC over the set of models considered. For nonlinear and other complex models, we need to replace d by some measure of model complexity. We discuss this in Section 7.6.

Given a set of models $f_\alpha(x)$ indexed by a tuning parameter α, denote by $\overline{\text{err}}(\alpha)$ and $d(\alpha)$ the training error and number of parameters for each model. Then for this set of models we define

$$\text{AIC}(\alpha) = \overline{\text{err}}(\alpha) + 2 \cdot \frac{d(\alpha)}{N} \hat{\sigma}_\varepsilon^2. \tag{7.26}$$

The function $\text{AIC}(\alpha)$ provides an estimate of the test error curve, and we find the tuning parameter $\hat{\alpha}$ that minimizes it. Our final chosen model is $f_{\hat{\alpha}}(x)$. Note that if the basis functions are chosen adaptively, (7.19) no longer holds. For example, if we have a total of p inputs, and we choose the best-fitting linear model with $d < p$ inputs, the optimism will exceed $(2d/N)\sigma_\varepsilon^2$. Put another way, by choosing the best-fitting model with d inputs, the *effective number of parameters* fit is more than d.

Figure 7.4 shows AIC in action for the phoneme recognition example of Section 5.2.3 on page 124. The input vector is the log-periodogram of the spoken vowel, quantized to 256 uniformly spaced frequencies. A linear logistic regression model is used to predict the phoneme class, with coefficient function $\beta(f) = \sum_{m=1}^{M} h_m(f)\theta_m$, an expansion in M spline basis functions. For any given M, a basis of natural cubic splines is used for the h_m, with knots chosen uniformly over the range of frequencies (so $d(\alpha) = d(M) = M$). Using AIC to select the number of basis functions will approximately minimize $\text{Err}(M)$ for both entropy and 0–1 loss.

The simple formula

$$(2/N)\sum_{i=1}^{N} \text{Cov}(\hat{y}_i, y_i) = (2d/N)\sigma_\varepsilon^2$$

holds exactly for linear models with additive errors and squared error loss, and approximately for linear models and log-likelihoods. In particular, the formula does not hold in general for 0–1 loss (Efron, 1986), although many authors nevertheless use it in that context (right panel of Figure 7.4).

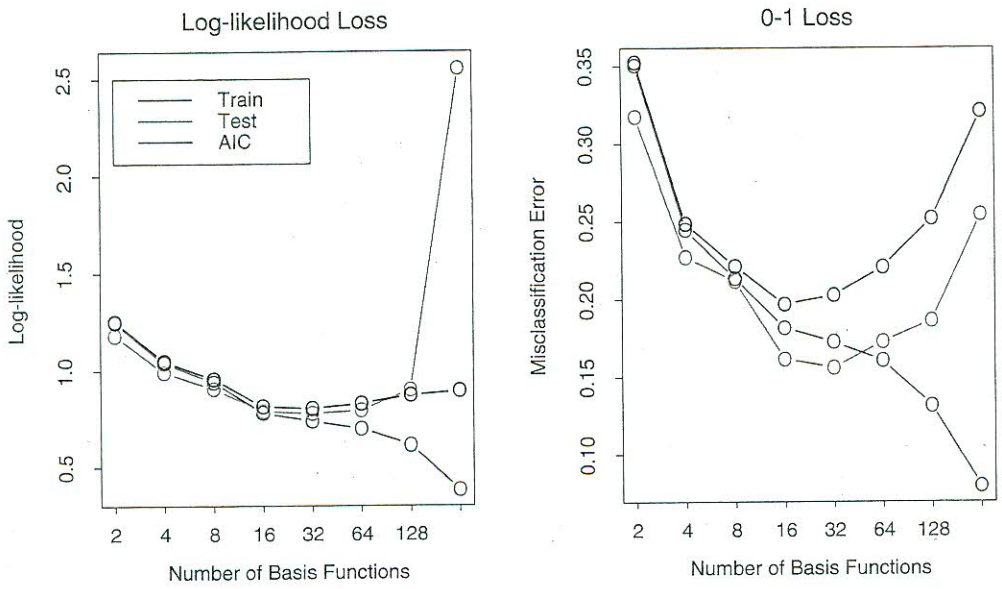

FIGURE 7.4. *AIC used for model selection for the phoneme recognition example of Section 5.2.3. The logistic regression coefficient function* $\beta(f) = \sum_{m=1}^{M} h_m(f)\theta_m$ *is modeled as an expansion in* M *spline basis functions. In the left panel we see the AIC statistic used to estimate* $\mathrm{Err}_{\mathrm{in}}$ *using log-likelihood loss. Included is an estimate of* Err *based on an independent test sample. It does well except for the extremely over-parametrized case (*$M = 256$ *parameters for* $N = 1000$ *observations). In the right panel the same is done for 0–1 loss. Although the AIC formula does not strictly apply here, it does a reasonable job in this case.*

7.6 The Effective Number of Parameters

The concept of "number of parameters" can be generalized, especially to models where regularization is used in the fitting. Suppose we stack the outcomes $y_1, y_2, \ldots, y_N$ into a vector $\mathbf{y}$, and similarly for the predictions $\hat{\mathbf{y}}$. Then a linear fitting method is one for which we can write

$$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}, \tag{7.27}$$

where $\mathbf{S}$ is an $N \times N$ matrix depending on the input vectors x_i but not on the y_i. Linear fitting methods include linear regression on the original features or on a derived basis set, and smoothing methods that use quadratic shrinkage, such as ridge regression and cubic smoothing splines. Then the *effective number of parameters* is defined as

$$d(\mathbf{S}) = \mathrm{trace}(\mathbf{S}), \tag{7.28}$$

the sum of the diagonal elements of $\mathbf{S}$. Note that if $\mathbf{S}$ is an orthogonal-projection matrix onto a basis set spanned by M features, then $\mathrm{trace}(\mathbf{S}) =$

M. It turns out that trace($\mathbf{S}$) is exactly the correct quantity to replace d as the number of parameters in the C_p statistic (7.22) (Exercise 7.4 and 7.5). We motivate $d = \text{trace}(\mathbf{S})$ in some detail in Section 5.4.1 on page 129.

For models like neural networks, in which we minimize an error function $R(w)$ with weight decay penalty (regularization) $\alpha \sum_m w_m^2$, the effective number of parameters has the form

$$d(\alpha) = \sum_{m=1}^{M} \frac{\theta_m}{\theta_m + \alpha}, \tag{7.29}$$

where the θ_m are the eigenvalues of the Hessian matrix $\partial^2 R(w)/\partial w \partial w^T$. Expression (7.29) follows from (7.28) if we make a quadratic approximation to the error function at the solution (Bishop, 1995).

7.7 The Bayesian Approach and BIC

The Bayesian information criterion (BIC), like AIC, is applicable in settings where the fitting is carried out by maximization of a log-likelihood. The generic form of BIC is

$$\text{BIC} = -2 \cdot \text{loglik} + (\log N) \cdot d. \tag{7.30}$$

The BIC statistic (times 1/2) is also known as the Schwartz criterion (Schwartz, 1979).

Under the Gaussian model, assuming the variance σ_ε^2 is known, $-2 \cdot$loglik equals (up to a constant) $\sum_i (y_i - \hat{f}(x_i))^2/\sigma_\varepsilon^2$, which is $N \cdot \overline{\text{err}}/\sigma_\varepsilon^2$ for squared error loss. Hence we can write

$$\text{BIC} = \frac{N}{\sigma_\varepsilon^2}\left[\overline{\text{err}} + (\log N) \cdot \frac{d}{N}\sigma_\varepsilon^2\right]. \tag{7.31}$$

Therefore BIC is proportional to AIC (C_p), with the factor 2 replaced by $\log N$. Assuming $N > e^2 \approx 7.4$, BIC tends to penalize complex models more heavily, giving preference to simpler models in selection. As with AIC, σ_ε^2 is typically estimated by the mean squared error of a low-bias model. For classification problems, use of the multinomial log-likelihood leads to a similar relationship with the AIC, using cross-entropy as the error measure. Note however that the misclassification error measure does not arise in the BIC context, since it does not correspond to the log-likelihood of the data under any probability model.

Despite its similarity with AIC, BIC is motivated in quite a different way. It arises in the Bayesian approach to model selection, which we now describe.

Suppose we have a set of candidate models $\mathcal{M}_m, m = 1, \ldots, M$ and corresponding model parameters θ_m, and we wish to choose a best model

from among them. Assuming we have a prior distribution $\Pr(\theta_m|\mathcal{M}_m)$ for the parameters of each model $\mathcal{M}_m$, the posterior probability of a given model is

$$\begin{aligned}\Pr(\mathcal{M}_m|\mathbf{Z}) &\propto \Pr(\mathcal{M}_m)\cdot\Pr(\mathbf{Z}|\mathcal{M}_m) \\ &\propto \Pr(\mathcal{M}_m)\cdot\int \Pr(\mathbf{Z}|\theta_m,\mathcal{M}_m)\Pr(\theta_m|\mathcal{M}_m)d\theta_m,\end{aligned} \tag{7.32}$$

where $\mathbf{Z}$ represents the training data $\{x_i, y_i\}_1^N$. To compare two models $\mathcal{M}_m$ and $\mathcal{M}_\ell$, we form the posterior odds

$$\frac{\Pr(\mathcal{M}_m|\mathbf{Z})}{\Pr(\mathcal{M}_\ell|\mathbf{Z})} = \frac{\Pr(\mathcal{M}_m)}{\Pr(\mathcal{M}_\ell)}\cdot\frac{\Pr(\mathbf{Z}|\mathcal{M}_m)}{\Pr(\mathbf{Z}|\mathcal{M}_\ell)}. \tag{7.33}$$

If the odds are greater than one we choose model m, otherwise we choose model ℓ. The rightmost quantity

$$\mathrm{BF}(\mathbf{Z}) = \frac{\Pr(\mathbf{Z}|\mathcal{M}_m)}{\Pr(\mathbf{Z}|\mathcal{M}_\ell)} \tag{7.34}$$

is called the *Bayes factor*, the contribution of the data toward the posterior odds.

Typically we assume that the prior over models is uniform, so that $\Pr(\mathcal{M}_m)$ is constant. We need some way of approximating $\Pr(\mathbf{Z}|\mathcal{M}_m)$. A so-called Laplace approximation to the integral followed by some other simplifications (Ripley, 1996, page 64) to (7.32) gives

$$\log\Pr(\mathbf{Z}|\mathcal{M}_m) = \log\Pr(\mathbf{Z}|\hat{\theta}_m,\mathcal{M}_m) - \frac{d_m}{2}\cdot\log N + O(1). \tag{7.35}$$

Here $\hat{\theta}_m$ is a maximum likelihood estimate and d_m is the number of free parameters in model $\mathcal{M}_m$. If we define our loss function to be

$$-2\log\Pr(\mathbf{Z}|\hat{\theta}_m,\mathcal{M}_m),$$

this is equivalent to the BIC criterion of equation (7.30).

Therefore, choosing the model with minimum BIC is equivalent to choosing the model with largest (approximate) posterior probability. But this framework gives us more. If we compute the BIC criterion for a set of M, models, giving BIC_m, $m = 1, 2, \ldots, M$, then we can estimate the posterior probability of each model $\mathcal{M}_m$ as

$$\frac{e^{-\frac{1}{2}\cdot\mathrm{BIC}_m}}{\sum_{\ell=1}^{M} e^{-\frac{1}{2}\cdot\mathrm{BIC}_\ell}}. \tag{7.36}$$

Thus we can estimate not only the best model, but also assess the relative merits of the models considered.

For model selection purposes, there is no clear choice between AIC and BIC. BIC is asymptotically consistent as a selection criterion. What this means is that given a family of models, including the true model, the probability that BIC will select the correct model approaches one as the sample size $N \to \infty$. This is not the case for AIC, which tends to choose models which are too complex as $N \to \infty$. On the other hand, for finite samples, BIC often chooses models that are too simple, because of its heavy penalty on complexity.

7.8 Minimum Description Length

The minimum description length (MDL) approach gives a selection criterion formally identical to the BIC approach, but is motivated from an optimal coding viewpoint. We first review the theory of coding for data compression, and then apply it to model selection.

We think of our datum z as a message that we want to encode and send to someone else (the "receiver"). We think of our model as a way of encoding the datum, and will choose the most parsimonious model, that is the shortest code, for the transmission.

Suppose first that the possible messages we might want to transmit are $z_1, z_2, \ldots, z_m$. Our code uses a finite alphabet of length A: for example, we might use a binary code $\{0, 1\}$ of length $A = 2$. Here is an example with four possible messages and a binary coding:

Message	z_1	z_2	z_3	z_4
Code	0	10	110	111

(7.37)

This code is known as an instantaneous prefix code: no code is the prefix of any other, and the receiver (who knows all of the possible codes), knows exactly when the message has been completely sent. We restrict our discussion to such instantaneous prefix codes.

One could use the coding in (7.37) or we could permute the codes, for example use codes $110, 10, 111, 0$ for z_1, z_2, z_3, z_4. How do we decide which to use? It depends on how often we will be sending each of the messages. If, for example, we will be sending z_1 most often, it makes sense to use the shortest code 0 for z_1. Using this kind of strategy—shorter codes for more frequent messages—the average message length will be shorter.

In general, if messages are sent with probabilities $\Pr(z_i), i = 1, 2, \ldots, 4$, a famous theorem due to Shannon says we should use code lengths $l_i = -\log_2 \Pr(z_i)$ and the average message length satisfies

$$\mathrm{E}(\text{length}) \geq -\sum \Pr(z_i) \log_2 (\Pr(z_i)). \tag{7.38}$$

The right-hand side above is also called the entropy of the distribution $\Pr(z_i)$. The inequality is an equality when the probabilities satisfy $p_i =$

A^{-l_i}. In our example, if $\Pr(z_i) = 1/2, 1/4, 1/8, 1/8$, respectively, then the coding shown in (7.37) is optimal and achieves the entropy lower bound.

In general the lower bound cannot be achieved, but procedures like the Huffmann coding scheme can get close to the bound. Note that with an infinite set of messages, the entropy is replaced by $-\int \Pr(z) \log_2 \Pr(z) dz$.

From this result we glean the following:

> *To transmit a random variable z having probability density function $\Pr(z)$, we require about $-\log_2 \Pr(z)$ bits of information.*

We henceforth change notation from $\log_2 \Pr(z)$ to $\log \Pr(z) = \log_e \Pr(z)$; this is for convenience, and just introduces an unimportant multiplicative constant.

Now we apply this result to the problem of model selection. We have a model M with parameters θ, and data $\mathbf{Z} = (\mathbf{X}, \mathbf{y})$ consisting of both inputs and outputs. Let the (conditional) probability of the outputs under the model be $\Pr(\mathbf{y}|\theta, M, \mathbf{X})$, assume the receiver knows all of the inputs, and we wish to transmit the outputs. Then the message length required to transmit the outputs is

$$\text{length} = -\log \Pr(\mathbf{y}|\theta, M, \mathbf{X}) - \log \Pr(\theta|M), \tag{7.39}$$

the log-probability of the target values given the inputs. The second term is the average code length for transmitting the model parameters θ, while the first term is the average code length for transmitting the discrepancy between the model and actual target values. For example suppose we have a single target y with $y \sim N(\theta, \sigma^2)$, parameter $\theta \sim N(0, 1)$ and no input (for simplicity). Then the message length is

$$\text{length} = \text{constant} + \log \sigma + \frac{(y-\theta)^2}{\sigma^2} + \frac{\theta^2}{2}. \tag{7.40}$$

Note that the smaller σ, is the shorter the message length, since y is more concentrated around θ.

The MDL principle says that we should choose the model that minimizes (7.39). We recognize (7.39) as the (negative) log-posterior distribution, and hence minimizing description length is equivalent to maximizing posterior probability. Hence the BIC criterion, derived as approximation to log-posterior probability, can also be viewed as a device for (approximate) model choice by minimum description length.

Note that we have ignored the precision with which a random variable z is coded. With a finite code length we cannot code a continuous variable exactly. However, if we code z within a tolerance δz, the message length needed is the log of the probability in the interval $[z, z+\delta z]$ which is well approximated by $\delta z \Pr(z)$ if δz is small. Since $\log \delta z \Pr(z) = \log \delta z + \log \Pr(z)$, this means we can just ignore the constant $\log \delta z$ and use $\log \Pr(z)$ as our measure of message length, as we did above.

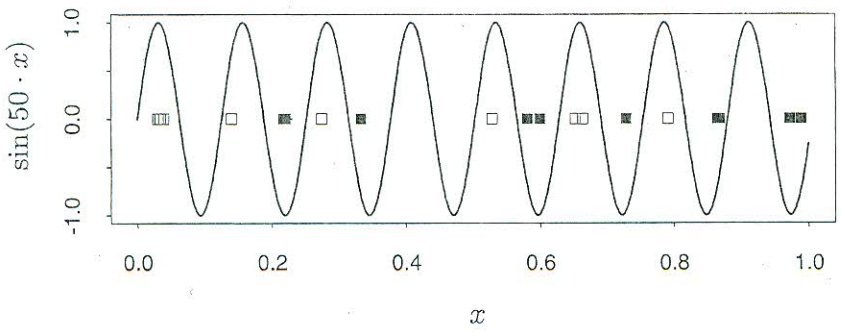

FIGURE 7.5. *The solid curve is the function* $\sin(50x)$ *for* $x \in [0, 1]$. *The blue (solid) and green (hollow) points illustrate how the associated indicator function* $I(\sin(\alpha x) > 0)$ *can shatter (separate) an arbitrarily large number of points by choosing an appropriately high frequency* α.

The preceding view of MDL for model selection says that we should choose the model with highest posterior probability. However many Bayesians would instead do inference by sampling from the posterior distribution.

7.9 Vapnik–Chernovenkis Dimension

A difficulty in using estimates of in-sample error is the need to specify the number of parameters (or the complexity) d used in the fit. Although the effective number of parameters introduced in Section 7.6 is useful for some nonlinear models, it is not fully general. The Vapnik–Chernovenkis (VC) theory provides such a general measure of complexity, and gives associated bounds on the optimism. Here we give a brief review of this theory.

Suppose we have a class of functions $\{f(x, \alpha)\}$ indexed by a parameter vector α, with $x \in \mathbb{R}^p$. Assume for now that f is an indicator function, that is, takes the values 0 or 1. If $\alpha = (\alpha_0, \alpha_1)$ and f is the linear indicator function $I(\alpha_0 + \alpha_1^T x > 0)$, then it seems reasonable to say that the complexity of the class f is the number of parameters $p + 1$. But what about $f(x, \alpha) = I(\sin \alpha \cdot x)$ where α is any real number and $x \in \mathbb{R}$? The function $\sin(50 \cdot x)$ is shown in Figure 7.5. This is a very wiggly function that gets even rougher as the frequency α increases, but it has only one parameter: despite this, it doesn't seem reasonable to conclude that it has less complexity than the linear indicator function $I(\alpha_0 + \alpha_1 x)$ in $p = 1$ dimension.

The Vapnik–Chernovenkis dimension is a way of measuring the complexity of a class of functions by assessing how wiggly its members can be.

> *The* VC dimension *of the class* $\{f(x, \alpha)\}$ *is defined to be the largest number of points (in some configuration) that can be* shattered *by members of* $\{f(x, \alpha)\}$.