



PATTERN ANALYSIS AND MACHINE INTELLIGENCE 2015-2016
prof. Matteo Matteucci

Course book

Project Repository

Click

Contents:

Book	Chapter	Title	Page
ISL*	2	Statistical learning	2
ISL	3	Linear regression	33
ISL	4	Classification	86
ESL*	4.5	Separating Hyperplanes	117
ISL	6	Linear Model Selection and Regularization	124
ISL	9	Support Vector Machines	170
ESL	12.1-3	Support Vector Machines and Flexible Discriminants	195

ISL* — "An Introduction to Statistical Learning: with Applications in R" by Gareth James
Selected chapters are given without labs and with conceptual exercises

ESL* — "The Elements of Statistical Learning" by Trevor Hastie

2

Statistical Learning

2.1 What Is Statistical Learning?

In order to motivate our study of statistical learning, we begin with a simple example. Suppose that we are statistical consultants hired by a client to provide advice on how to improve sales of a particular product. The **Advertising** data set consists of the **sales** of that product in 200 different markets, along with advertising budgets for the product in each of those markets for three different media: **TV**, **radio**, and **newspaper**. The data are displayed in Figure 2.1. It is not possible for our client to directly increase sales of the product. On the other hand, they can control the advertising expenditure in each of the three media. Therefore, if we determine that there is an association between advertising and sales, then we can instruct our client to adjust advertising budgets, thereby indirectly increasing sales. In other words, our goal is to develop an accurate model that can be used to predict sales on the basis of the three media budgets.

In this setting, the advertising budgets are *input variables* while **sales** is an *output variable*. The input variables are typically denoted using the symbol X , with a subscript to distinguish them. So X_1 might be the **TV** budget, X_2 the **radio** budget, and X_3 the **newspaper** budget. The inputs go by different names, such as *predictors*, *independent variables*, *features*, or sometimes just *variables*. The output variable—in this case, **sales**—is often called the *response* or *dependent variable*, and is typically denoted using the symbol Y . Throughout this book, we will use all of these terms interchangeably.

input variable	sales
output variable	TV radio newspaper
predictor	X_1 X_2 X_3
independent variable	X_1 X_2 X_3
feature variable	X_1 X_2 X_3
response variable	Y
dependent variable	Y

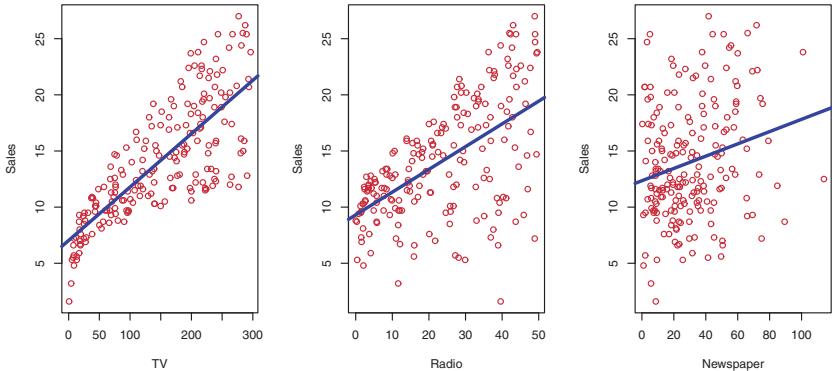


FIGURE 2.1. The `Advertising` data set. The plot displays `sales`, in thousands of units, as a function of `TV`, `radio`, and `newspaper` budgets, in thousands of dollars, for 200 different markets. In each plot we show the simple least squares fit of `sales` to that variable, as described in Chapter 3. In other words, each blue line represents a simple model that can be used to predict `sales` using `TV`, `radio`, and `newspaper`, respectively.

More generally, suppose that we observe a quantitative response Y and p different predictors, X_1, X_2, \dots, X_p . We assume that there is some relationship between Y and $X = (X_1, X_2, \dots, X_p)$, which can be written in the very general form

$$Y = f(X) + \epsilon. \quad (2.1)$$

Here f is some fixed but unknown function of X_1, \dots, X_p , and ϵ is a random *error term*, which is independent of X and has mean zero. In this formulation, f represents the *systematic* information that X provides about Y .

As another example, consider the left-hand panel of Figure 2.2, a plot of `income` versus `years of education` for 30 individuals in the `Income` data set. The plot suggests that one might be able to predict `income` using `years of education`. However, the function f that connects the input variable to the output variable is in general unknown. In this situation one must estimate f based on the observed points. Since `Income` is a simulated data set, f is known and is shown by the blue curve in the right-hand panel of Figure 2.2. The vertical lines represent the error terms ϵ . We note that some of the 30 observations lie above the blue curve and some lie below it; overall, the errors have approximately mean zero.

In general, the function f may involve more than one input variable. In Figure 2.3 we plot `income` as a function of `years of education` and `seniority`. Here f is a two-dimensional surface that must be estimated based on the observed data.

error term
systematic

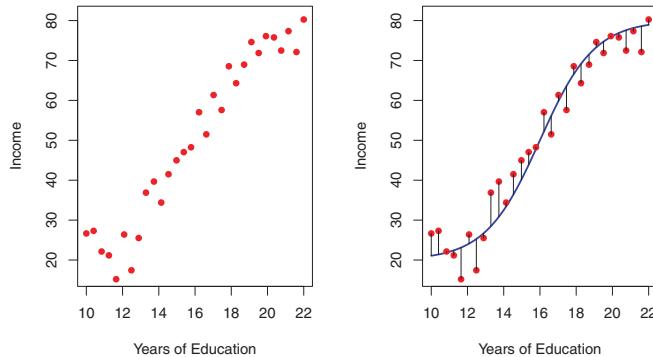


FIGURE 2.2. The `Income` data set. Left: The red dots are the observed values of `income` (in tens of thousands of dollars) and `years of education` for 30 individuals. Right: The blue curve represents the true underlying relationship between `income` and `years of education`, which is generally unknown (but is known in this case because the data were simulated). The black lines represent the error associated with each observation. Note that some errors are positive (if an observation lies above the blue curve) and some are negative (if an observation lies below the curve). Overall, these errors have approximately mean zero.

In essence, statistical learning refers to a set of approaches for estimating f . In this chapter we outline some of the key theoretical concepts that arise in estimating f , as well as tools for evaluating the estimates obtained.

2.1.1 Why Estimate f ?

There are two main reasons that we may wish to estimate f : *prediction* and *inference*. We discuss each in turn.

Prediction

In many situations, a set of inputs X are readily available, but the output Y cannot be easily obtained. In this setting, since the error term averages to zero, we can predict Y using

$$\hat{Y} = \hat{f}(X), \quad (2.2)$$

where \hat{f} represents our estimate for f , and \hat{Y} represents the resulting prediction for Y . In this setting, \hat{f} is often treated as a *black box*, in the sense that one is not typically concerned with the exact form of \hat{f} , provided that it yields accurate predictions for Y .

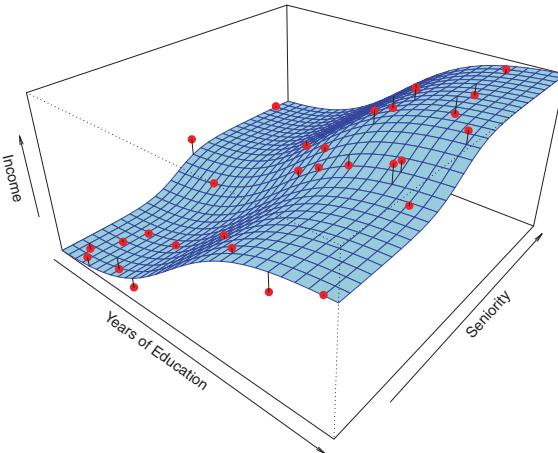


FIGURE 2.3. The plot displays `income` as a function of `years of education` and `seniority` in the `Income` data set. The blue surface represents the true underlying relationship between `income` and `years of education` and `seniority`, which is known since the data are simulated. The red dots indicate the observed values of these quantities for 30 individuals.

As an example, suppose that X_1, \dots, X_p are characteristics of a patient's blood sample that can be easily measured in a lab, and Y is a variable encoding the patient's risk for a severe adverse reaction to a particular drug. It is natural to seek to predict Y using X , since we can then avoid giving the drug in question to patients who are at high risk of an adverse reaction—that is, patients for whom the estimate of Y is high.

The accuracy of \hat{Y} as a prediction for Y depends on two quantities, which we will call the *reducible error* and the *irreducible error*. In general, \hat{f} will not be a perfect estimate for f , and this inaccuracy will introduce some error. This error is *reducible* because we can potentially improve the accuracy of \hat{f} by using the most appropriate statistical learning technique to estimate f . However, even if it were possible to form a perfect estimate for f , so that our estimated response took the form $\hat{Y} = f(X)$, our prediction would still have some error in it! This is because Y is also a function of ϵ , which, by definition, cannot be predicted using X . Therefore, variability associated with ϵ also affects the accuracy of our predictions. This is known as the *irreducible error*, because no matter how well we estimate f , we cannot reduce the error introduced by ϵ .

reducible
 error
 irreducible
 error

Why is the irreducible error larger than zero? The quantity ϵ may contain unmeasured variables that are useful in predicting Y : since we don't measure them, f cannot use them for its prediction. The quantity ϵ may also contain unmeasurable variation. For example, the risk of an adverse reaction might vary for a given patient on a given day, depending on

manufacturing variation in the drug itself or the patient's general feeling of well-being on that day.

Consider a given estimate \hat{f} and a set of predictors X , which yields the prediction $\hat{Y} = \hat{f}(X)$. Assume for a moment that both \hat{f} and X are fixed. Then, it is easy to show that

$$\begin{aligned} E(Y - \hat{Y})^2 &= E[f(X) + \epsilon - \hat{f}(X)]^2 \\ &= \underbrace{[f(X) - \hat{f}(X)]^2}_{\text{Reducible}} + \underbrace{\text{Var}(\epsilon)}_{\text{Irreducible}}, \end{aligned} \quad (2.3)$$

where $E(Y - \hat{Y})^2$ represents the average, or *expected value*, of the squared difference between the predicted and actual value of Y , and $\text{Var}(\epsilon)$ represents the *variance* associated with the error term ϵ .

The focus of this book is on techniques for estimating f with the aim of minimizing the reducible error. It is important to keep in mind that the irreducible error will always provide an upper bound on the accuracy of our prediction for Y . This bound is almost always unknown in practice.

Inference

We are often interested in understanding the way that Y is affected as X_1, \dots, X_p change. In this situation we wish to estimate f , but our goal is not necessarily to make predictions for Y . We instead want to understand the relationship between X and Y , or more specifically, to understand how Y changes as a function of X_1, \dots, X_p . Now \hat{f} cannot be treated as a black box, because we need to know its exact form. In this setting, one may be interested in answering the following questions:

- *Which predictors are associated with the response?* It is often the case that only a small fraction of the available predictors are substantially associated with Y . Identifying the few *important* predictors among a large set of possible variables can be extremely useful, depending on the application.
- *What is the relationship between the response and each predictor?* Some predictors may have a positive relationship with Y , in the sense that increasing the predictor is associated with increasing values of Y . Other predictors may have the opposite relationship. Depending on the complexity of f , the relationship between the response and a given predictor may also depend on the values of the other predictors.
- *Can the relationship between Y and each predictor be adequately summarized using a linear equation, or is the relationship more complicated?* Historically, most methods for estimating f have taken a linear form. In some situations, such an assumption is reasonable or even desirable. But often the true relationship is more complicated, in which case a linear model may not provide an accurate representation of the relationship between the input and output variables.

In this book, we will see a number of examples that fall into the prediction setting, the inference setting, or a combination of the two.

For instance, consider a company that is interested in conducting a direct-marketing campaign. The goal is to identify individuals who will respond positively to a mailing, based on observations of demographic variables measured on each individual. In this case, the demographic variables serve as predictors, and response to the marketing campaign (either positive or negative) serves as the outcome. The company is not interested in obtaining a deep understanding of the relationships between each individual predictor and the response; instead, the company simply wants an accurate model to predict the response using the predictors. This is an example of modeling for prediction.

In contrast, consider the **Advertising** data illustrated in Figure 2.1. One may be interested in answering questions such as:

- *Which media contribute to sales?*
- *Which media generate the biggest boost in sales?* or
- *How much increase in sales is associated with a given increase in TV advertising?*

This situation falls into the inference paradigm. Another example involves modeling the brand of a product that a customer might purchase based on variables such as price, store location, discount levels, competition price, and so forth. In this situation one might really be most interested in how each of the individual variables affects the probability of purchase. For instance, *what effect will changing the price of a product have on sales?* This is an example of modeling for inference.

Finally, some modeling could be conducted both for prediction and inference. For example, in a real estate setting, one may seek to relate values of homes to inputs such as crime rate, zoning, distance from a river, air quality, schools, income level of community, size of houses, and so forth. In this case one might be interested in how the individual input variables affect the prices—that is, *how much extra will a house be worth if it has a view of the river?* This is an inference problem. Alternatively, one may simply be interested in predicting the value of a home given its characteristics: *is this house under- or over-valued?* This is a prediction problem.

Depending on whether our ultimate goal is prediction, inference, or a combination of the two, different methods for estimating f may be appropriate. For example, *linear models* allow for relatively simple and interpretable inference, but may not yield as accurate predictions as some other approaches. In contrast, some of the highly non-linear approaches that we discuss in the later chapters of this book can potentially provide quite accurate predictions for Y , but this comes at the expense of a less interpretable model for which inference is more challenging.

linear model

2.1.2 How Do We Estimate f ?

Throughout this book, we explore many linear and non-linear approaches for estimating f . However, these methods generally share certain characteristics. We provide an overview of these shared characteristics in this section. We will always assume that we have observed a set of n different data points. For example in Figure 2.2 we observed $n = 30$ data points. These observations are called the *training data* because we will use these observations to train, or teach, our method how to estimate f . Let x_{ij} represent the value of the j th predictor, or input, for observation i , where $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, p$. Correspondingly, let y_i represent the response variable for the i th observation. Then our training data consist of $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ where $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})^T$.

Our goal is to apply a statistical learning method to the training data in order to estimate the unknown function f . In other words, we want to find a function \hat{f} such that $Y \approx \hat{f}(X)$ for any observation (X, Y) . Broadly speaking, most statistical learning methods for this task can be characterized as either *parametric* or *non-parametric*. We now briefly discuss these two types of approaches.

Parametric Methods

Parametric methods involve a two-step model-based approach.

1. First, we make an assumption about the functional form, or shape, of f . For example, one very simple assumption is that f is linear in X :

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p. \quad (2.4)$$

This is a *linear model*, which will be discussed extensively in Chapter 3. Once we have assumed that f is linear, the problem of estimating f is greatly simplified. Instead of having to estimate an entirely arbitrary p -dimensional function $f(X)$, one only needs to estimate the $p + 1$ coefficients $\beta_0, \beta_1, \dots, \beta_p$.

2. After a model has been selected, we need a procedure that uses the training data to *fit* or *train* the model. In the case of the linear model (2.4), we need to estimate the parameters $\beta_0, \beta_1, \dots, \beta_p$. That is, we want to find values of these parameters such that

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p.$$

The most common approach to fitting the model (2.4) is referred to as (*ordinary*) *least squares*, which we discuss in Chapter 3. However, least squares is one of many possible ways to fit the linear model. In Chapter 6, we discuss other approaches for estimating the parameters in (2.4).

The model-based approach just described is referred to as *parametric*; it reduces the problem of estimating f down to one of estimating a set of

training data

parametric
non-
parametric

least squares

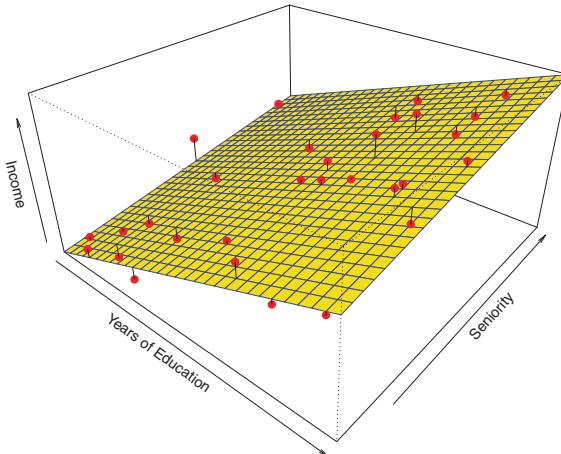


FIGURE 2.4. A linear model fit by least squares to the `Income` data from Figure 2.3. The observations are shown in red, and the yellow plane indicates the least squares fit to the data.

parameters. Assuming a parametric form for f simplifies the problem of estimating f because it is generally much easier to estimate a set of parameters, such as $\beta_0, \beta_1, \dots, \beta_p$ in the linear model (2.4), than it is to fit an entirely arbitrary function f . The potential disadvantage of a parametric approach is that the model we choose will usually not match the true unknown form of f . If the chosen model is too far from the true f , then our estimate will be poor. We can try to address this problem by choosing *flexible* models that can fit many different possible functional forms for f . But in general, fitting a more flexible model requires estimating a greater number of parameters. These more complex models can lead to a phenomenon known as *overfitting* the data, which essentially means they follow the errors, or *noise*, too closely. These issues are discussed throughout this book.

Figure 2.4 shows an example of the parametric approach applied to the `Income` data from Figure 2.3. We have fit a linear model of the form

$$\text{income} \approx \beta_0 + \beta_1 \times \text{education} + \beta_2 \times \text{seniority}.$$

Since we have assumed a linear relationship between the response and the two predictors, the entire fitting problem reduces to estimating β_0 , β_1 , and β_2 , which we do using least squares linear regression. Comparing Figure 2.3 to Figure 2.4, we can see that the linear fit given in Figure 2.4 is not quite right: the true f has some curvature that is not captured in the linear fit. However, the linear fit still appears to do a reasonable job of capturing the positive relationship between `years of education` and `income`, as well as the

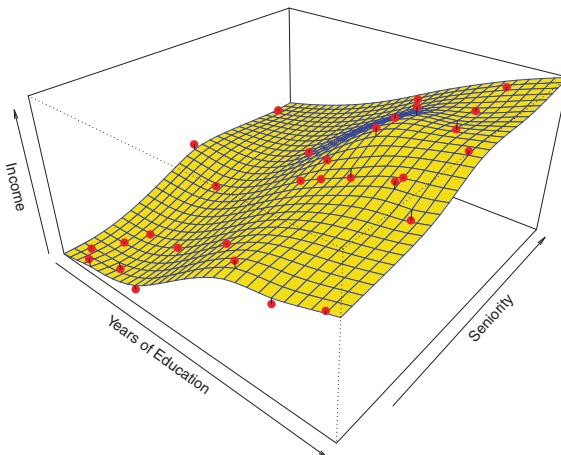


FIGURE 2.5. A smooth thin-plate spline fit to the `Income` data from Figure 2.3 is shown in yellow; the observations are displayed in red. Splines are discussed in Chapter 7.

slightly less positive relationship between `seniority` and `income`. It may be that with such a small number of observations, this is the best we can do.

Non-parametric Methods

Non-parametric methods do not make explicit assumptions about the functional form of f . Instead they seek an estimate of f that gets as close to the data points as possible without being too rough or wiggly. Such approaches can have a major advantage over parametric approaches: by avoiding the assumption of a particular functional form for f , they have the potential to accurately fit a wider range of possible shapes for f . Any parametric approach brings with it the possibility that the functional form used to estimate f is very different from the true f , in which case the resulting model will not fit the data well. In contrast, non-parametric approaches completely avoid this danger, since essentially no assumption about the form of f is made. But non-parametric approaches do suffer from a major disadvantage: since they do not reduce the problem of estimating f to a small number of parameters, a very large number of observations (far more than is typically needed for a parametric approach) is required in order to obtain an accurate estimate for f .

An example of a non-parametric approach to fitting the `Income` data is shown in Figure 2.5. A *thin-plate spline* is used to estimate f . This approach does not impose any pre-specified model on f . It instead attempts to produce an estimate for f that is as close as possible to the observed data, subject to the fit—that is, the yellow surface in Figure 2.5—being

thin-plate
spline

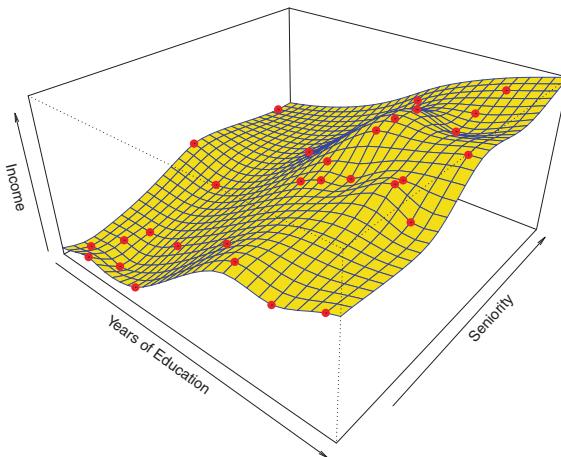


FIGURE 2.6. A rough thin-plate spline fit to the `Income` data from Figure 2.3. This fit makes zero errors on the training data.

smooth. In this case, the non-parametric fit has produced a remarkably accurate estimate of the true f shown in Figure 2.3. In order to fit a thin-plate spline, the data analyst must select a level of smoothness. Figure 2.6 shows the same thin-plate spline fit using a lower level of smoothness, allowing for a rougher fit. The resulting estimate fits the observed data perfectly! However, the spline fit shown in Figure 2.6 is far more variable than the true function f , from Figure 2.3. This is an example of overfitting the data, which we discussed previously. It is an undesirable situation because the fit obtained will not yield accurate estimates of the response on new observations that were not part of the original training data set. We discuss methods for choosing the *correct* amount of smoothness in Chapter 5. Splines are discussed in Chapter 7.

As we have seen, there are advantages and disadvantages to parametric and non-parametric methods for statistical learning. We explore both types of methods throughout this book.

2.1.3 The Trade-Off Between Prediction Accuracy and Model Interpretability

Of the many methods that we examine in this book, some are less flexible, or more restrictive, in the sense that they can produce just a relatively small range of shapes to estimate f . For example, linear regression is a relatively inflexible approach, because it can only generate linear functions such as the lines shown in Figure 2.1 or the plane shown in Figure 2.4.

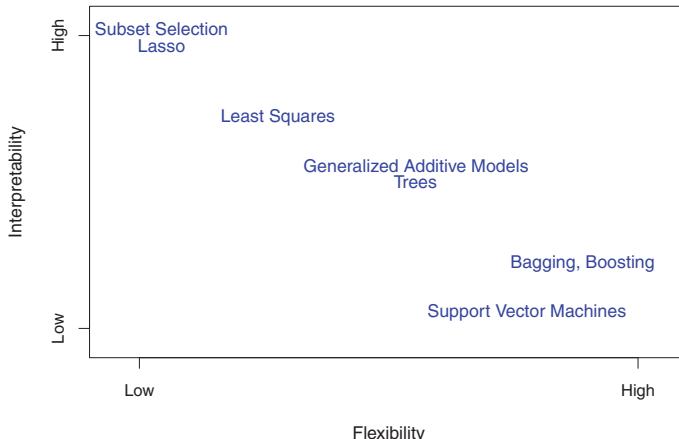


FIGURE 2.7. A representation of the tradeoff between flexibility and interpretability, using different statistical learning methods. In general, as the flexibility of a method increases, its interpretability decreases.

Other methods, such as the thin plate splines shown in Figures 2.5 and 2.6, are considerably more flexible because they can generate a much wider range of possible shapes to estimate f .

One might reasonably ask the following question: *why would we ever choose to use a more restrictive method instead of a very flexible approach?* There are several reasons that we might prefer a more restrictive model. If we are mainly interested in inference, then restrictive models are much more interpretable. For instance, when inference is the goal, the linear model may be a good choice since it will be quite easy to understand the relationship between Y and X_1, X_2, \dots, X_p . In contrast, very flexible approaches, such as the splines discussed in Chapter 7 and displayed in Figures 2.5 and 2.6, and the boosting methods discussed in Chapter 8, can lead to such complicated estimates of f that it is difficult to understand how any individual predictor is associated with the response.

Figure 2.7 provides an illustration of the trade-off between flexibility and interpretability for some of the methods that we cover in this book. Least squares linear regression, discussed in Chapter 3, is relatively inflexible but is quite interpretable. The *lasso*, discussed in Chapter 6, relies upon the linear model (2.4) but uses an alternative fitting procedure for estimating the coefficients $\beta_0, \beta_1, \dots, \beta_p$. The new procedure is more restrictive in estimating the coefficients, and sets a number of them to exactly zero. Hence in this sense the lasso is a less flexible approach than linear regression. It is also more interpretable than linear regression, because in the final model the response variable will only be related to a small subset of the predictors—namely, those with nonzero coefficient estimates. *Generalized*

additive models (GAMs), discussed in Chapter 7, instead extend the linear model (2.4) to allow for certain non-linear relationships. Consequently, GAMs are more flexible than linear regression. They are also somewhat less interpretable than linear regression, because the relationship between each predictor and the response is now modeled using a curve. Finally, fully non-linear methods such as *bagging*, *boosting*, and *support vector machines* with non-linear kernels, discussed in Chapters 8 and 9, are highly flexible approaches that are harder to interpret.

generalized
additive
model

We have established that when inference is the goal, there are clear advantages to using simple and relatively inflexible statistical learning methods. In some settings, however, we are only interested in prediction, and the interpretability of the predictive model is simply not of interest. For instance, if we seek to develop an algorithm to predict the price of a stock, our sole requirement for the algorithm is that it predict accurately—interpretability is not a concern. In this setting, we might expect that it will be best to use the most flexible model available. Surprisingly, this is not always the case! We will often obtain more accurate predictions using a less flexible method. This phenomenon, which may seem counterintuitive at first glance, has to do with the potential for overfitting in highly flexible methods. We saw an example of overfitting in Figure 2.6. We will discuss this very important concept further in Section 2.2 and throughout this book.

bagging
boosting
support
vector
machine

2.1.4 Supervised Versus Unsupervised Learning

Most statistical learning problems fall into one of two categories: *supervised* or *unsupervised*. The examples that we have discussed so far in this chapter all fall into the supervised learning domain. For each observation of the predictor measurement(s) x_i , $i = 1, \dots, n$ there is an associated response measurement y_i . We wish to fit a model that relates the response to the predictors, with the aim of accurately predicting the response for future observations (prediction) or better understanding the relationship between the response and the predictors (inference). Many classical statistical learning methods such as linear regression and *logistic regression* (Chapter 4), as well as more modern approaches such as GAM, boosting, and support vector machines, operate in the supervised learning domain. The vast majority of this book is devoted to this setting.

supervised
unsupervised

logistic
regression

In contrast, unsupervised learning describes the somewhat more challenging situation in which for every observation $i = 1, \dots, n$, we observe a vector of measurements x_i but no associated response y_i . It is not possible to fit a linear regression model, since there is no response variable to predict. In this setting, we are in some sense working blind; the situation is referred to as *unsupervised* because we lack a response variable that can supervise our analysis. What sort of statistical analysis is

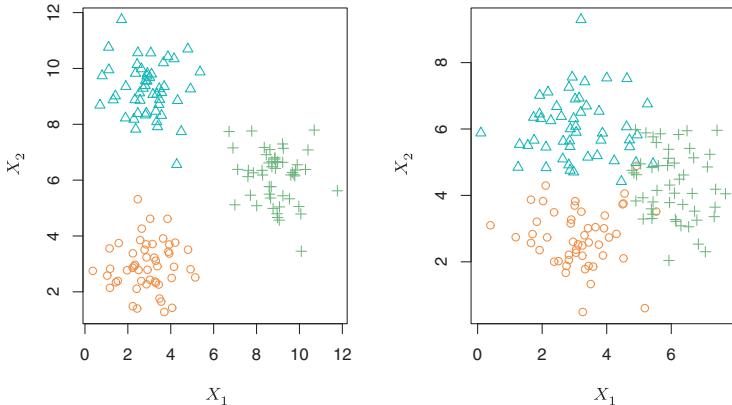


FIGURE 2.8. A clustering data set involving three groups. Each group is shown using a different colored symbol. Left: The three groups are well-separated. In this setting, a clustering approach should successfully identify the three groups. Right: There is some overlap among the groups. Now the clustering task is more challenging.

possible? We can seek to understand the relationships between the variables or between the observations. One statistical learning tool that we may use in this setting is *cluster analysis*, or clustering. The goal of cluster analysis is to ascertain, on the basis of x_1, \dots, x_n , whether the observations fall into relatively distinct groups. For example, in a market segmentation study we might observe multiple characteristics (variables) for potential customers, such as zip code, family income, and shopping habits. We might believe that the customers fall into different groups, such as big spenders versus low spenders. If the information about each customer's spending patterns were available, then a supervised analysis would be possible. However, this information is not available—that is, we do not know whether each potential customer is a big spender or not. In this setting, we can try to cluster the customers on the basis of the variables measured, in order to identify distinct groups of potential customers. Identifying such groups can be of interest because it might be that the groups differ with respect to some property of interest, such as spending habits.

cluster analysis

Figure 2.8 provides a simple illustration of the clustering problem. We have plotted 150 observations with measurements on two variables, X_1 and X_2 . Each observation corresponds to one of three distinct groups. For illustrative purposes, we have plotted the members of each group using different colors and symbols. However, in practice the group memberships are unknown, and the goal is to determine the group to which each observation belongs. In the left-hand panel of Figure 2.8, this is a relatively easy task because the groups are well-separated. In contrast, the right-hand panel illustrates a more challenging problem in which there is some overlap

between the groups. A clustering method could not be expected to assign all of the overlapping points to their correct group (blue, green, or orange).

In the examples shown in Figure 2.8, there are only two variables, and so one can simply visually inspect the scatterplots of the observations in order to identify clusters. However, in practice, we often encounter data sets that contain many more than two variables. In this case, we cannot easily plot the observations. For instance, if there are p variables in our data set, then $p(p - 1)/2$ distinct scatterplots can be made, and visual inspection is simply not a viable way to identify clusters. For this reason, automated clustering methods are important. We discuss clustering and other unsupervised learning approaches in Chapter 10.

Many problems fall naturally into the supervised or unsupervised learning paradigms. However, sometimes the question of whether an analysis should be considered supervised or unsupervised is less clear-cut. For instance, suppose that we have a set of n observations. For m of the observations, where $m < n$, we have both predictor measurements and a response measurement. For the remaining $n - m$ observations, we have predictor measurements but no response measurement. Such a scenario can arise if the predictors can be measured relatively cheaply but the corresponding responses are much more expensive to collect. We refer to this setting as a *semi-supervised learning* problem. In this setting, we wish to use a statistical learning method that can incorporate the m observations for which response measurements are available as well as the $n - m$ observations for which they are not. Although this is an interesting topic, it is beyond the scope of this book.

semi-supervised learning

2.1.5 Regression Versus Classification Problems

Variables can be characterized as either *quantitative* or *qualitative* (also known as *categorical*). Quantitative variables take on numerical values. Examples include a person's age, height, or income, the value of a house, and the price of a stock. In contrast, qualitative variables take on values in one of K different *classes*, or categories. Examples of qualitative variables include a person's gender (male or female), the brand of product purchased (brand A, B, or C), whether a person defaults on a debt (yes or no), or a cancer diagnosis (Acute Myelogenous Leukemia, Acute Lymphoblastic Leukemia, or No Leukemia). We tend to refer to problems with a quantitative response as *regression* problems, while those involving a qualitative response are often referred to as *classification* problems. However, the distinction is not always that crisp. Least squares linear regression (Chapter 3) is used with a quantitative response, whereas logistic regression (Chapter 4) is typically used with a qualitative (two-class, or *binary*) response. As such it is often used as a classification method. But since it estimates class probabilities, it can be thought of as a regression

quantitative
qualitative
categorical

class

regression
classification

binary

method as well. Some statistical methods, such as K -nearest neighbors (Chapters 2 and 4) and boosting (Chapter 8), can be used in the case of either quantitative or qualitative responses.

We tend to select statistical learning methods on the basis of whether the response is quantitative or qualitative; i.e. we might use linear regression when quantitative and logistic regression when qualitative. However, whether the *predictors* are qualitative or quantitative is generally considered less important. Most of the statistical learning methods discussed in this book can be applied regardless of the predictor variable type, provided that any qualitative predictors are properly *coded* before the analysis is performed. This is discussed in Chapter 3.

2.2 Assessing Model Accuracy

One of the key aims of this book is to introduce the reader to a wide range of statistical learning methods that extend far beyond the standard linear regression approach. Why is it necessary to introduce so many different statistical learning approaches, rather than just a single *best* method? *There is no free lunch in statistics*: no one method dominates all others over all possible data sets. On a particular data set, one specific method may work best, but some other method may work better on a similar but different data set. Hence it is an important task to decide for any given set of data which method produces the best results. Selecting the best approach can be one of the most challenging parts of performing statistical learning in practice.

In this section, we discuss some of the most important concepts that arise in selecting a statistical learning procedure for a specific data set. As the book progresses, we will explain how the concepts presented here can be applied in practice.

2.2.1 Measuring the Quality of Fit

In order to evaluate the performance of a statistical learning method on a given data set, we need some way to measure how well its predictions actually match the observed data. That is, we need to quantify the extent to which the predicted response value for a given observation is close to the true response value for that observation. In the regression setting, the most commonly-used measure is the *mean squared error* (MSE), given by

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}(x_i))^2, \quad (2.5)$$

mean
squared
error

where $\hat{f}(x_i)$ is the prediction that \hat{f} gives for the i th observation. The MSE will be small if the predicted responses are very close to the true responses, and will be large if for some of the observations, the predicted and true responses differ substantially.

The MSE in (2.5) is computed using the training data that was used to fit the model, and so should more accurately be referred to as the *training MSE*. But in general, we do not really care how well the method works on the training data. Rather, *we are interested in the accuracy of the predictions that we obtain when we apply our method to previously unseen test data*. Why is this what we care about? Suppose that we are interested in developing an algorithm to predict a stock's price based on previous stock returns. We can train the method using stock returns from the past 6 months. But we don't really care how well our method predicts last week's stock price. We instead care about how well it will predict tomorrow's price or next month's price. On a similar note, suppose that we have clinical measurements (e.g. weight, blood pressure, height, age, family history of disease) for a number of patients, as well as information about whether each patient has diabetes. We can use these patients to train a statistical learning method to predict risk of diabetes based on clinical measurements. In practice, we want this method to accurately predict diabetes risk for *future patients* based on their clinical measurements. We are not very interested in whether or not the method accurately predicts diabetes risk for patients used to train the model, since we already know which of those patients have diabetes.

To state it more mathematically, suppose that we fit our statistical learning method on our training observations $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, and we obtain the estimate \hat{f} . We can then compute $\hat{f}(x_1), \hat{f}(x_2), \dots, \hat{f}(x_n)$. If these are approximately equal to y_1, y_2, \dots, y_n , then the training MSE given by (2.5) is small. However, we are really not interested in whether $\hat{f}(x_i) \approx y_i$; instead, we want to know whether $\hat{f}(x_0)$ is approximately equal to y_0 , where (x_0, y_0) is a *previously unseen test observation not used to train the statistical learning method*. We want to choose the method that gives the lowest *test MSE*, as opposed to the lowest training MSE. In other words, if we had a large number of test observations, we could compute

$$\text{Ave}(y_0 - \hat{f}(x_0))^2, \quad (2.6)$$

the average squared prediction error for these test observations (x_0, y_0) . We'd like to select the model for which the average of this quantity—the test MSE—is as small as possible.

How can we go about trying to select a method that minimizes the test MSE? In some settings, we may have a test data set available—that is, we may have access to a set of observations that were not used to train the statistical learning method. We can then simply evaluate (2.6) on the test observations, and select the learning method for which the test MSE is

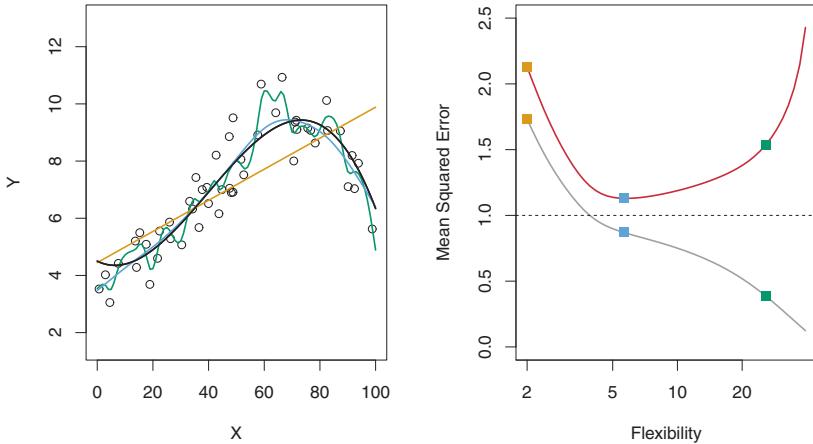


FIGURE 2.9. Left: Data simulated from f , shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves). Right: Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.

smallest. But what if no test observations are available? In that case, one might imagine simply selecting a statistical learning method that minimizes the training MSE (2.5). This seems like it might be a sensible approach, since the training MSE and the test MSE appear to be closely related. Unfortunately, there is a fundamental problem with this strategy: there is no guarantee that the method with the lowest training MSE will also have the lowest test MSE. Roughly speaking, the problem is that many statistical methods specifically estimate coefficients so as to minimize the training set MSE. For these methods, the training set MSE can be quite small, but the test MSE is often much larger.

Figure 2.9 illustrates this phenomenon on a simple example. In the left-hand panel of Figure 2.9, we have generated observations from (2.1) with the true f given by the black curve. The orange, blue and green curves illustrate three possible estimates for f obtained using methods with increasing levels of flexibility. The orange line is the linear regression fit, which is relatively inflexible. The blue and green curves were produced using *smoothing splines*, discussed in Chapter 7, with different levels of smoothness. It is clear that as the level of flexibility increases, the curves fit the observed data more closely. The green curve is the most flexible and matches the data very well; however, we observe that it fits the true f (shown in black) poorly because it is too wiggly. By adjusting the level of flexibility of the smoothing spline fit, we can produce many different fits to this data.

smoothing
spline

We now move on to the right-hand panel of Figure 2.9. The grey curve displays the average training MSE as a function of flexibility, or more formally the *degrees of freedom*, for a number of smoothing splines. The degrees of freedom is a quantity that summarizes the flexibility of a curve; it is discussed more fully in Chapter 7. The orange, blue and green squares indicate the MSEs associated with the corresponding curves in the left-hand panel. A more restricted and hence smoother curve has fewer degrees of freedom than a wiggly curve—note that in Figure 2.9, linear regression is at the most restrictive end, with two degrees of freedom. The training MSE declines monotonically as flexibility increases. In this example the true f is non-linear, and so the orange linear fit is not flexible enough to estimate f well. The green curve has the lowest training MSE of all three methods, since it corresponds to the most flexible of the three curves fit in the left-hand panel.

In this example, we know the true function f , and so we can also compute the test MSE over a very large test set, as a function of flexibility. (Of course, in general f is unknown, so this will not be possible.) The test MSE is displayed using the red curve in the right-hand panel of Figure 2.9. As with the training MSE, the test MSE initially declines as the level of flexibility increases. However, at some point the test MSE levels off and then starts to increase again. Consequently, the orange and green curves both have high test MSE. The blue curve minimizes the test MSE, which should not be surprising given that visually it appears to estimate f the best in the left-hand panel of Figure 2.9. The horizontal dashed line indicates $\text{Var}(\epsilon)$, the irreducible error in (2.3), which corresponds to the lowest achievable test MSE among all possible methods. Hence, the smoothing spline represented by the blue curve is close to optimal.

In the right-hand panel of Figure 2.9, as the flexibility of the statistical learning method increases, we observe a monotone decrease in the training MSE and a *U-shape* in the test MSE. This is a fundamental property of statistical learning that holds regardless of the particular data set at hand and regardless of the statistical method being used. As model flexibility increases, training MSE will decrease, but the test MSE may not. When a given method yields a small training MSE but a large test MSE, we are said to be *overfitting* the data. This happens because our statistical learning procedure is working too hard to find patterns in the training data, and may be picking up some patterns that are just caused by random chance rather than by true properties of the unknown function f . When we overfit the training data, the test MSE will be very large because the supposed patterns that the method found in the training data simply don't exist in the test data. Note that regardless of whether or not overfitting has occurred, we almost always expect the training MSE to be smaller than the test MSE because most statistical learning methods either directly or indirectly seek to minimize the training MSE. Overfitting refers specifically to the case in which a less flexible model would have yielded a smaller test MSE.

degrees of freedom

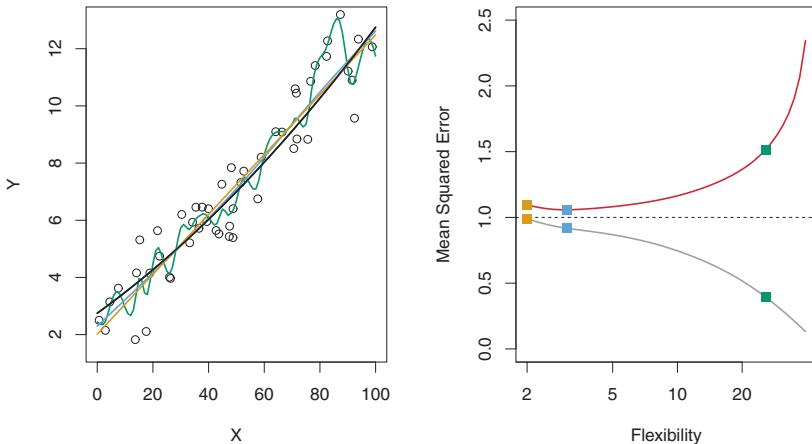


FIGURE 2.10. Details are as in Figure 2.9, using a different true f that is much closer to linear. In this setting, linear regression provides a very good fit to the data.

Figure 2.10 provides another example in which the true f is approximately linear. Again we observe that the training MSE decreases monotonically as the model flexibility increases, and that there is a U-shape in the test MSE. However, because the truth is close to linear, the test MSE only decreases slightly before increasing again, so that the orange least squares fit is substantially better than the highly flexible green curve. Finally, Figure 2.11 displays an example in which f is highly non-linear. The training and test MSE curves still exhibit the same general patterns, but now there is a rapid decrease in both curves before the test MSE starts to increase slowly.

In practice, one can usually compute the training MSE with relative ease, but estimating test MSE is considerably more difficult because usually no test data are available. As the previous three examples illustrate, the flexibility level corresponding to the model with the minimal test MSE can vary considerably among data sets. Throughout this book, we discuss a variety of approaches that can be used in practice to estimate this minimum point. One important method is *cross-validation* (Chapter 5), which is a method for estimating test MSE using the training data.

cross-validation

2.2.2 The Bias-Variance Trade-Off

The U-shape observed in the test MSE curves (Figures 2.9–2.11) turns out to be the result of two competing properties of statistical learning methods. Though the mathematical proof is beyond the scope of this book, it is possible to show that the expected test MSE, for a given value x_0 , can

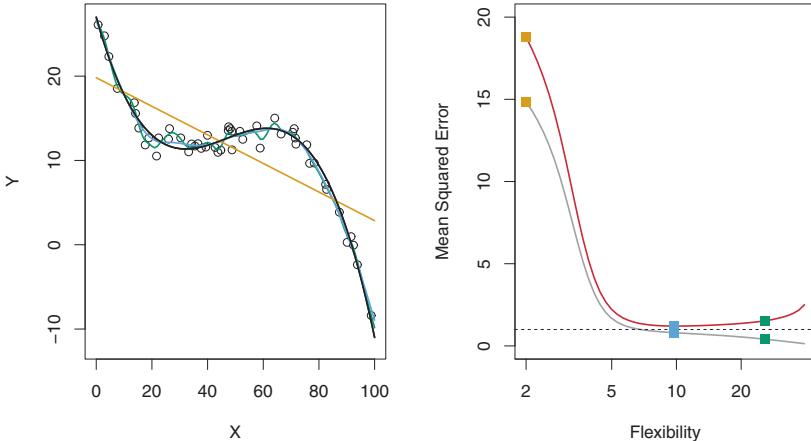


FIGURE 2.11. Details are as in Figure 2.9, using a different f that is far from linear. In this setting, linear regression provides a very poor fit to the data.

always be decomposed into the sum of three fundamental quantities: the variance of $\hat{f}(x_0)$, the squared bias of $\hat{f}(x_0)$ and the variance of the error terms ϵ . That is,

$$E \left(y_0 - \hat{f}(x_0) \right)^2 = \text{Var}(\hat{f}(x_0)) + [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\epsilon). \quad (2.7)$$

Here the notation $E \left(y_0 - \hat{f}(x_0) \right)^2$ defines the *expected test MSE*, and refers to the average test MSE that we would obtain if we repeatedly estimated f using a large number of training sets, and tested each at x_0 . The overall expected test MSE can be computed by averaging $E \left(y_0 - \hat{f}(x_0) \right)^2$ over all possible values of x_0 in the test set.

Equation 2.7 tells us that in order to minimize the expected test error, we need to select a statistical learning method that simultaneously achieves *low variance* and *low bias*. Note that variance is inherently a nonnegative quantity, and squared bias is also nonnegative. Hence, we see that the expected test MSE can never lie below $\text{Var}(\epsilon)$, the irreducible error from (2.3).

What do we mean by the *variance* and *bias* of a statistical learning method? *Variance* refers to the amount by which \hat{f} would change if we estimated it using a different training data set. Since the training data are used to fit the statistical learning method, different training data sets will result in a different \hat{f} . But ideally the estimate for f should not vary too much between training sets. However, if a method has high variance then small changes in the training data can result in large changes in \hat{f} . In general, more flexible statistical methods have higher variance. Consider the

green and orange curves in Figure 2.9. The flexible green curve is following the observations very closely. It has high variance because changing any one of these data points may cause the estimate \hat{f} to change considerably. In contrast, the orange least squares line is relatively inflexible and has low variance, because moving any single observation will likely cause only a small shift in the position of the line.

On the other hand, *bias* refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model. For example, linear regression assumes that there is a linear relationship between Y and X_1, X_2, \dots, X_p . It is unlikely that any real-life problem truly has such a simple linear relationship, and so performing linear regression will undoubtedly result in some bias in the estimate of f . In Figure 2.11, the true f is substantially non-linear, so no matter how many training observations we are given, it will not be possible to produce an accurate estimate using linear regression. In other words, linear regression results in high bias in this example. However, in Figure 2.10 the true f is very close to linear, and so given enough data, it should be possible for linear regression to produce an accurate estimate. Generally, more flexible methods result in less bias.

As a general rule, as we use more flexible methods, the variance will increase and the bias will decrease. The relative rate of change of these two quantities determines whether the test MSE increases or decreases. As we increase the flexibility of a class of methods, the bias tends to initially decrease faster than the variance increases. Consequently, the expected test MSE declines. However, at some point increasing flexibility has little impact on the bias but starts to significantly increase the variance. When this happens the test MSE increases. Note that we observed this pattern of decreasing test MSE followed by increasing test MSE in the right-hand panels of Figures 2.9–2.11.

The three plots in Figure 2.12 illustrate Equation 2.7 for the examples in Figures 2.9–2.11. In each case the blue solid curve represents the squared bias, for different levels of flexibility, while the orange curve corresponds to the variance. The horizontal dashed line represents $\text{Var}(\epsilon)$, the irreducible error. Finally, the red curve, corresponding to the test set MSE, is the sum of these three quantities. In all three cases, the variance increases and the bias decreases as the method's flexibility increases. However, the flexibility level corresponding to the optimal test MSE differs considerably among the three data sets, because the squared bias and variance change at different rates in each of the data sets. In the left-hand panel of Figure 2.12, the bias initially decreases rapidly, resulting in an initial sharp decrease in the expected test MSE. On the other hand, in the center panel of Figure 2.12 the true f is close to linear, so there is only a small decrease in bias as flexibility increases, and the test MSE only declines slightly before increasing rapidly as the variance increases. Finally, in the right-hand panel of Figure 2.12, as flexibility increases, there is a dramatic decline in bias because

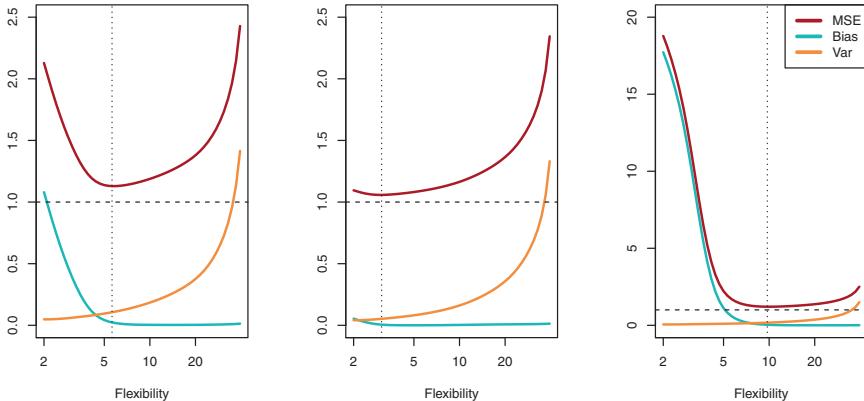


FIGURE 2.12. Squared bias (blue curve), variance (orange curve), $\text{Var}(\epsilon)$ (dashed line), and test MSE (red curve) for the three data sets in Figures 2.9–2.11. The vertical dotted line indicates the flexibility level corresponding to the smallest test MSE.

the true f is very non-linear. There is also very little increase in variance as flexibility increases. Consequently, the test MSE declines substantially before experiencing a small increase as model flexibility increases.

The relationship between bias, variance, and test set MSE given in Equation 2.7 and displayed in Figure 2.12 is referred to as the *bias-variance trade-off*. Good test set performance of a statistical learning method requires low variance as well as low squared bias. This is referred to as a trade-off because it is easy to obtain a method with extremely low bias but high variance (for instance, by drawing a curve that passes through every single training observation) or a method with very low variance but high bias (by fitting a horizontal line to the data). The challenge lies in finding a method for which both the variance and the squared bias are low. This trade-off is one of the most important recurring themes in this book.

bias-variance
trade-off

In a real-life situation in which f is unobserved, it is generally not possible to explicitly compute the test MSE, bias, or variance for a statistical learning method. Nevertheless, one should always keep the bias-variance trade-off in mind. In this book we explore methods that are extremely flexible and hence can essentially eliminate bias. However, this does not guarantee that they will outperform a much simpler method such as linear regression. To take an extreme example, suppose that the true f is linear. In this situation linear regression will have no bias, making it very hard for a more flexible method to compete. In contrast, if the true f is highly non-linear and we have an ample number of training observations, then we may do better using a highly flexible approach, as in Figure 2.11. In Chapter 5 we discuss cross-validation, which is a way to estimate the test MSE using the training data.

2.2.3 The Classification Setting

Thus far, our discussion of model accuracy has been focused on the regression setting. But many of the concepts that we have encountered, such as the bias-variance trade-off, transfer over to the classification setting with only some modifications due to the fact that y_i is no longer numerical. Suppose that we seek to estimate f on the basis of training observations $\{(x_1, y_1), \dots, (x_n, y_n)\}$, where now y_1, \dots, y_n are qualitative. The most common approach for quantifying the accuracy of our estimate \hat{f} is the training *error rate*, the proportion of mistakes that are made if we apply our estimate \hat{f} to the training observations:

$$\frac{1}{n} \sum_{i=1}^n I(y_i \neq \hat{y}_i). \quad (2.8)$$

Here \hat{y}_i is the predicted class label for the i th observation using \hat{f} . And $I(y_i \neq \hat{y}_i)$ is an *indicator variable* that equals 1 if $y_i \neq \hat{y}_i$ and zero if $y_i = \hat{y}_i$. If $I(y_i \neq \hat{y}_i) = 0$ then the i th observation was classified correctly by our classification method; otherwise it was misclassified. Hence Equation 2.8 computes the fraction of incorrect classifications.

Equation 2.8 is referred to as the *training error* rate because it is computed based on the data that was used to train our classifier. As in the regression setting, we are most interested in the error rates that result from applying our classifier to test observations that were not used in training. The *test error* rate associated with a set of test observations of the form (x_0, y_0) is given by

$$\text{Ave}(I(y_0 \neq \hat{y}_0)), \quad (2.9)$$

where \hat{y}_0 is the predicted class label that results from applying the classifier to the test observation with predictor x_0 . A *good* classifier is one for which the test error (2.9) is smallest.

The Bayes Classifier

It is possible to show (though the proof is outside of the scope of this book) that the test error rate given in (2.9) is minimized, on average, by a very simple classifier that *assigns each observation to the most likely class, given its predictor values*. In other words, we should simply assign a test observation with predictor vector x_0 to the class j for which

$$\Pr(Y = j | X = x_0) \quad (2.10)$$

is largest. Note that (2.10) is a *conditional probability*: it is the probability that $Y = j$, given the observed predictor vector x_0 . This very simple classifier is called the *Bayes classifier*. In a two-class problem where there are only two possible response values, say *class 1* or *class 2*, the Bayes classifier

conditional probability

Bayes classifier

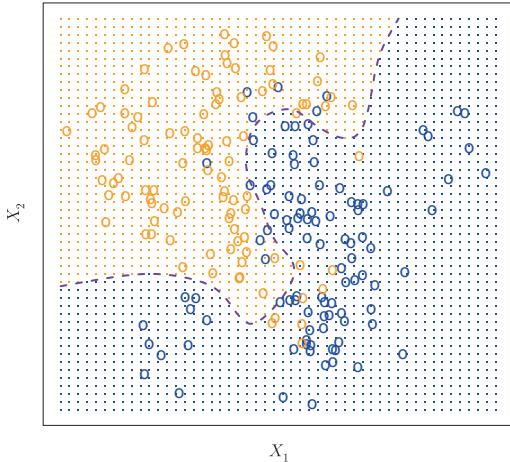


FIGURE 2.13. A simulated data set consisting of 100 observations in each of two groups, indicated in blue and in orange. The purple dashed line represents the Bayes decision boundary. The orange background grid indicates the region in which a test observation will be assigned to the orange class, and the blue background grid indicates the region in which a test observation will be assigned to the blue class.

corresponds to predicting class one if $\Pr(Y = 1|X = x_0) > 0.5$, and class two otherwise.

Figure 2.13 provides an example using a simulated data set in a two-dimensional space consisting of predictors X_1 and X_2 . The orange and blue circles correspond to training observations that belong to two different classes. For each value of X_1 and X_2 , there is a different probability of the response being orange or blue. Since this is simulated data, we know how the data were generated and we can calculate the conditional probabilities for each value of X_1 and X_2 . The orange shaded region reflects the set of points for which $\Pr(Y = \text{orange}|X)$ is greater than 50 %, while the blue shaded region indicates the set of points for which the probability is below 50 %. The purple dashed line represents the points where the probability is exactly 50 %. This is called the *Bayes decision boundary*. The Bayes classifier's prediction is determined by the Bayes decision boundary; an observation that falls on the orange side of the boundary will be assigned to the orange class, and similarly an observation on the blue side of the boundary will be assigned to the blue class.

The Bayes classifier produces the lowest possible test error rate, called the *Bayes error rate*. Since the Bayes classifier will always choose the class for which (2.10) is largest, the error rate at $X = x_0$ will be $1 - \max_j \Pr(Y = j|X = x_0)$. In general, the overall Bayes error rate is given by

$$1 - E \left(\max_j \Pr(Y = j|X) \right), \quad (2.11)$$

where the expectation averages the probability over all possible values of X . For our simulated data, the Bayes error rate is 0.1304. It is greater than zero, because the classes overlap in the true population so $\max_j \Pr(Y = j|X = x_0) < 1$ for some values of x_0 . The Bayes error rate is analogous to the irreducible error, discussed earlier.

K-Nearest Neighbors

In theory we would always like to predict qualitative responses using the Bayes classifier. But for real data, we do not know the conditional distribution of Y given X , and so computing the Bayes classifier is impossible. Therefore, the Bayes classifier serves as an unattainable gold standard against which to compare other methods. Many approaches attempt to estimate the conditional distribution of Y given X , and then classify a given observation to the class with highest *estimated* probability. One such method is the *K-nearest neighbors* (KNN) classifier. Given a positive integer K and a test observation x_0 , the KNN classifier first identifies the K points in the training data that are closest to x_0 , represented by \mathcal{N}_0 . It then estimates the conditional probability for class j as the fraction of points in \mathcal{N}_0 whose response values equal j :

$$\Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j). \quad (2.12)$$

*K-nearest
neighbors*

Finally, KNN applies Bayes rule and classifies the test observation x_0 to the class with the largest probability.

Figure 2.14 provides an illustrative example of the KNN approach. In the left-hand panel, we have plotted a small training data set consisting of six blue and six orange observations. Our goal is to make a prediction for the point labeled by the black cross. Suppose that we choose $K = 3$. Then KNN will first identify the three observations that are closest to the cross. This neighborhood is shown as a circle. It consists of two blue points and one orange point, resulting in estimated probabilities of $2/3$ for the blue class and $1/3$ for the orange class. Hence KNN will predict that the black cross belongs to the blue class. In the right-hand panel of Figure 2.14 we have applied the KNN approach with $K = 3$ at all of the possible values for X_1 and X_2 , and have drawn in the corresponding KNN decision boundary.

Despite the fact that it is a very simple approach, KNN can often produce classifiers that are surprisingly close to the optimal Bayes classifier. Figure 2.15 displays the KNN decision boundary, using $K = 10$, when applied to the larger simulated data set from Figure 2.13. Notice that even though the true distribution is not known by the KNN classifier, the KNN decision boundary is very close to that of the Bayes classifier. The test error rate using KNN is 0.1363, which is close to the Bayes error rate of 0.1304.

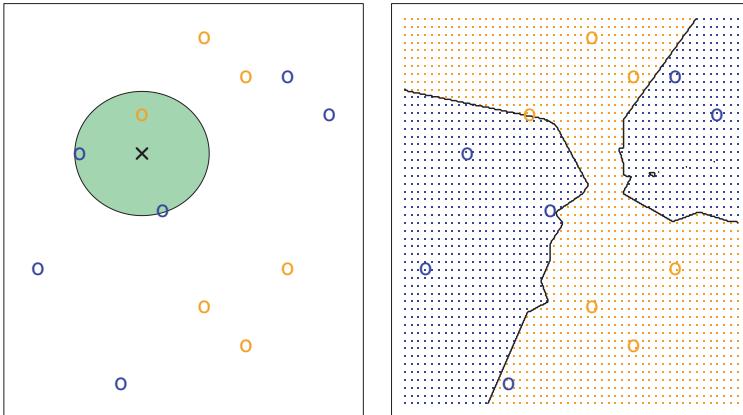


FIGURE 2.14. The KNN approach, using $K = 3$, is illustrated in a simple situation with six blue observations and six orange observations. Left: a test observation at which a predicted class label is desired is shown as a black cross. The three closest points to the test observation are identified, and it is predicted that the test observation belongs to the most commonly-occurring class, in this case blue. Right: The KNN decision boundary for this example is shown in black. The blue grid indicates the region in which a test observation will be assigned to the blue class, and the orange grid indicates the region in which it will be assigned to the orange class.

The choice of K has a drastic effect on the KNN classifier obtained. Figure 2.16 displays two KNN fits to the simulated data from Figure 2.13, using $K = 1$ and $K = 100$. When $K = 1$, the decision boundary is overly flexible and finds patterns in the data that don't correspond to the Bayes decision boundary. This corresponds to a classifier that has low bias but very high variance. As K grows, the method becomes less flexible and produces a decision boundary that is close to linear. This corresponds to a low-variance but high-bias classifier. On this simulated data set, neither $K = 1$ nor $K = 100$ give good predictions: they have test error rates of 0.1695 and 0.1925, respectively.

Just as in the regression setting, there is not a strong relationship between the training error rate and the test error rate. With $K = 1$, the KNN training error rate is 0, but the test error rate may be quite high. In general, as we use more flexible classification methods, the training error rate will decline but the test error rate may not. In Figure 2.17, we have plotted the KNN test and training errors as a function of $1/K$. As $1/K$ increases, the method becomes more flexible. As in the regression setting, the training error rate consistently declines as the flexibility increases. However, the test error exhibits a characteristic U-shape, declining at first (with a minimum at approximately $K = 10$) before increasing again when the method becomes excessively flexible and overfits.

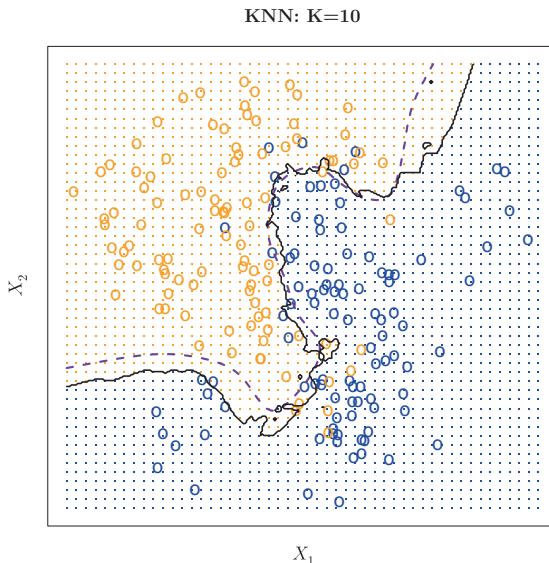


FIGURE 2.15. The black curve indicates the KNN decision boundary on the data from Figure 2.13, using $K = 10$. The Bayes decision boundary is shown as a purple dashed line. The KNN and Bayes decision boundaries are very similar.

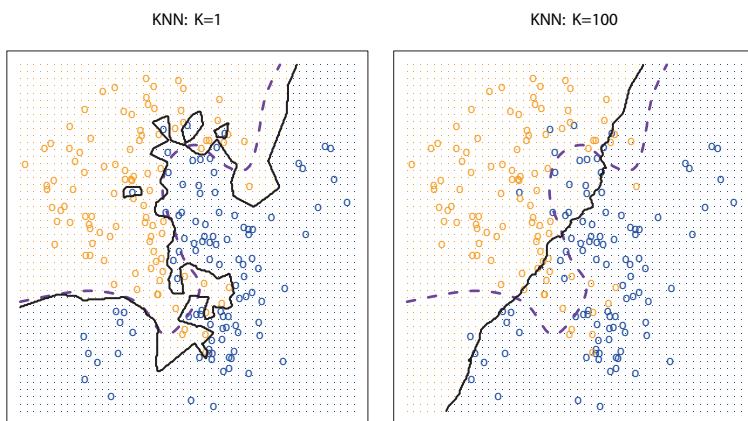


FIGURE 2.16. A comparison of the KNN decision boundaries (solid black curves) obtained using $K = 1$ and $K = 100$ on the data from Figure 2.13. With $K = 1$, the decision boundary is overly flexible, while with $K = 100$ it is not sufficiently flexible. The Bayes decision boundary is shown as a purple dashed line.

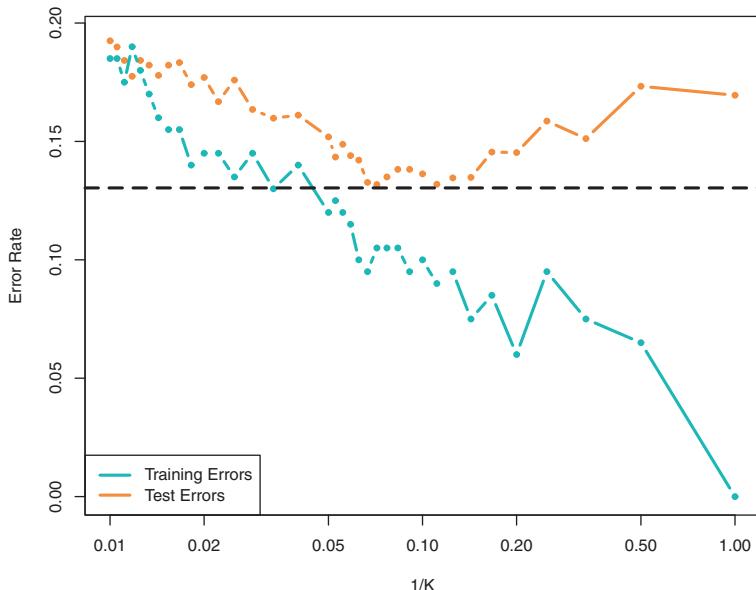


FIGURE 2.17. The KNN training error rate (blue, 200 observations) and test error rate (orange, 5,000 observations) on the data from Figure 2.13, as the level of flexibility (assessed using $1/K$) increases, or equivalently as the number of neighbors K decreases. The black dashed line indicates the Bayes error rate. The jumpiness of the curves is due to the small size of the training data set.

In both the regression and classification settings, choosing the correct level of flexibility is critical to the success of any statistical learning method. The bias-variance tradeoff, and the resulting U-shape in the test error, can make this a difficult task. In Chapter 5, we return to this topic and discuss various methods for estimating test error rates and thereby choosing the optimal level of flexibility for a given statistical learning method.

2.3 Lab: Introduction to R

In this lab, we will introduce some simple R commands. The best way to learn a new language is to try out the commands. R can be downloaded from

<http://cran.r-project.org/>

2.3.1 Basic Commands

R uses *functions* to perform operations. To run a function called `funcname`, we type `funcname(input1, input2)`, where the inputs (or *arguments*) `input1` and `input2` are arguments.

2.4 Exercises

Conceptual

1. For each of parts (a) through (d), indicate whether we would generally expect the performance of a flexible statistical learning method to be better or worse than an inflexible method. Justify your answer.
 - (a) The sample size n is extremely large, and the number of predictors p is small.
 - (b) The number of predictors p is extremely large, and the number of observations n is small.
 - (c) The relationship between the predictors and response is highly non-linear.
 - (d) The variance of the error terms, i.e. $\sigma^2 = \text{Var}(\epsilon)$, is extremely high.
2. Explain whether each scenario is a classification or regression problem, and indicate whether we are most interested in inference or prediction. Finally, provide n and p .
 - (a) We collect a set of data on the top 500 firms in the US. For each firm we record profit, number of employees, industry and the CEO salary. We are interested in understanding which factors affect CEO salary.
 - (b) We are considering launching a new product and wish to know whether it will be a *success* or a *failure*. We collect data on 20 similar products that were previously launched. For each product we have recorded whether it was a success or failure, price charged for the product, marketing budget, competition price, and ten other variables.
 - (c) We are interesting in predicting the % change in the US dollar in relation to the weekly changes in the world stock markets. Hence we collect weekly data for all of 2012. For each week we record the % change in the dollar, the % change in the US market, the % change in the British market, and the % change in the German market.
3. We now revisit the bias-variance decomposition.
 - (a) Provide a sketch of typical (squared) bias, variance, training error, test error, and Bayes (or irreducible) error curves, on a single plot, as we go from less flexible statistical learning methods towards more flexible approaches. The x -axis should represent

the amount of flexibility in the method, and the y -axis should represent the values for each curve. There should be five curves. Make sure to label each one.

- (b) Explain why each of the five curves has the shape displayed in part (a).
4. You will now think of some real-life applications for statistical learning.
- (a) Describe three real-life applications in which *classification* might be useful. Describe the response, as well as the predictors. Is the goal of each application inference or prediction? Explain your answer.
 - (b) Describe three real-life applications in which *regression* might be useful. Describe the response, as well as the predictors. Is the goal of each application inference or prediction? Explain your answer.
 - (c) Describe three real-life applications in which *cluster analysis* might be useful.
5. What are the advantages and disadvantages of a very flexible (versus a less flexible) approach for regression or classification? Under what circumstances might a more flexible approach be preferred to a less flexible approach? When might a less flexible approach be preferred?
6. Describe the differences between a parametric and a non-parametric statistical learning approach. What are the advantages of a parametric approach to regression or classification (as opposed to a non-parametric approach)? What are its disadvantages?
7. The table below provides a training data set containing six observations, three predictors, and one qualitative response variable.

Obs.	X_1	X_2	X_3	Y
1	0	3	0	Red
2	2	0	0	Red
3	0	1	3	Red
4	0	1	2	Green
5	-1	0	1	Green
6	1	1	1	Red

Suppose we wish to use this data set to make a prediction for Y when $X_1 = X_2 = X_3 = 0$ using K -nearest neighbors.

- (a) Compute the Euclidean distance between each observation and the test point, $X_1 = X_2 = X_3 = 0$.

- (b) What is our prediction with $K = 1$? Why?
- (c) What is our prediction with $K = 3$? Why?
- (d) If the Bayes decision boundary in this problem is highly non-linear, then would we expect the *best* value for K to be large or small? Why?

Applied

8. This exercise relates to the `College` data set, which can be found in the file `College.csv`. It contains a number of variables for 777 different universities and colleges in the US. The variables are

- `Private` : Public/private indicator
- `Apps` : Number of applications received
- `Accept` : Number of applicants accepted
- `Enroll` : Number of new students enrolled
- `Top10perc` : New students from top 10 % of high school class
- `Top25perc` : New students from top 25 % of high school class
- `F.Undergrad` : Number of full-time undergraduates
- `P.Undergrad` : Number of part-time undergraduates
- `Outstate` : Out-of-state tuition
- `Room.Board` : Room and board costs
- `Books` : Estimated book costs
- `Personal` : Estimated personal spending
- `PhD` : Percent of faculty with Ph.D.'s
- `Terminal` : Percent of faculty with terminal degree
- `S.F.Ratio` : Student/faculty ratio
- `perc.alumni` : Percent of alumni who donate
- `Expend` : Instructional expenditure per student
- `Grad.Rate` : Graduation rate

Before reading the data into `R`, it can be viewed in Excel or a text editor.

- (a) Use the `read.csv()` function to read the data into `R`. Call the loaded data `college`. Make sure that you have the directory set to the correct location for the data.
- (b) Look at the data using the `fix()` function. You should notice that the first column is just the name of each university. We don't really want `R` to treat this as data. However, it may be handy to have these names for later. Try the following commands:

3

Linear Regression

This chapter is about *linear regression*, a very simple approach for supervised learning. In particular, linear regression is a useful tool for predicting a quantitative response. Linear regression has been around for a long time and is the topic of innumerable textbooks. Though it may seem somewhat dull compared to some of the more modern statistical learning approaches described in later chapters of this book, linear regression is still a useful and widely used statistical learning method. Moreover, it serves as a good jumping-off point for newer approaches: as we will see in later chapters, many fancy statistical learning approaches can be seen as generalizations or extensions of linear regression. Consequently, the importance of having a good understanding of linear regression before studying more complex learning methods cannot be overstated. In this chapter, we review some of the key ideas underlying the linear regression model, as well as the least squares approach that is most commonly used to fit this model.

Recall the **Advertising** data from Chapter 2. Figure 2.1 displays **sales** (in thousands of units) for a particular product as a function of advertising budgets (in thousands of dollars) for **TV**, **radio**, and **newspaper** media. Suppose that in our role as statistical consultants we are asked to suggest, on the basis of this data, a marketing plan for next year that will result in high product sales. What information would be useful in order to provide such a recommendation? Here are a few important questions that we might seek to address:

1. *Is there a relationship between advertising budget and sales?*

Our first goal should be to determine whether the data provide

evidence of an association between advertising expenditure and sales. If the evidence is weak, then one might argue that no money should be spent on advertising!

2. *How strong is the relationship between advertising budget and sales?*

Assuming that there is a relationship between advertising and sales, we would like to know the strength of this relationship. In other words, given a certain advertising budget, can we predict sales with a high level of accuracy? This would be a strong relationship. Or is a prediction of sales based on advertising expenditure only slightly better than a random guess? This would be a weak relationship.

3. *Which media contribute to sales?*

Do all three media—TV, radio, and newspaper—contribute to sales, or do just one or two of the media contribute? To answer this question, we must find a way to separate out the individual effects of each medium when we have spent money on all three media.

4. *How accurately can we estimate the effect of each medium on sales?*

For every dollar spent on advertising in a particular medium, by what amount will sales increase? How accurately can we predict this amount of increase?

5. *How accurately can we predict future sales?*

For any given level of television, radio, or newspaper advertising, what is our prediction for sales, and what is the accuracy of this prediction?

6. *Is the relationship linear?*

If there is approximately a straight-line relationship between advertising expenditure in the various media and sales, then linear regression is an appropriate tool. If not, then it may still be possible to transform the predictor or the response so that linear regression can be used.

7. *Is there synergy among the advertising media?*

Perhaps spending \$50,000 on television advertising and \$50,000 on radio advertising results in more sales than allocating \$100,000 to either television or radio individually. In marketing, this is known as a *synergy* effect, while in statistics it is called an *interaction* effect.

synergy
interaction

It turns out that linear regression can be used to answer each of these questions. We will first discuss all of these questions in a general context, and then return to them in this specific context in Section 3.4.

3.1 Simple Linear Regression

Simple linear regression lives up to its name: it is a very straightforward approach for predicting a quantitative response Y on the basis of a single predictor variable X . It assumes that there is approximately a linear relationship between X and Y . Mathematically, we can write this linear relationship as

$$Y \approx \beta_0 + \beta_1 X. \quad (3.1)$$

You might read “ \approx ” as “*is approximately modeled as*”. We will sometimes describe (3.1) by saying that we are *regressing Y on X* (or Y onto X). For example, X may represent **TV** advertising and Y may represent **sales**. Then we can regress **sales** onto **TV** by fitting the model

$$\text{sales} \approx \beta_0 + \beta_1 \times \text{TV}.$$

In Equation 3.1, β_0 and β_1 are two unknown constants that represent the *intercept* and *slope* terms in the linear model. Together, β_0 and β_1 are known as the model *coefficients* or *parameters*. Once we have used our training data to produce estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ for the model coefficients, we can predict future sales on the basis of a particular value of TV advertising by computing

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x, \quad (3.2)$$

where \hat{y} indicates a prediction of Y on the basis of $X = x$. Here we use a *hat* symbol, $\hat{}$, to denote the estimated value for an unknown parameter or coefficient, or to denote the predicted value of the response.

3.1.1 Estimating the Coefficients

In practice, β_0 and β_1 are unknown. So before we can use (3.1) to make predictions, we must use data to estimate the coefficients. Let

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$$

represent n observation pairs, each of which consists of a measurement of X and a measurement of Y . In the **Advertising** example, this data set consists of the TV advertising budget and product sales in $n = 200$ different markets. (Recall that the data are displayed in Figure 2.1.) Our goal is to obtain coefficient estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ such that the linear model (3.1) fits the available data well—that is, so that $y_i \approx \hat{\beta}_0 + \hat{\beta}_1 x_i$ for $i = 1, \dots, n$. In other words, we want to find an intercept $\hat{\beta}_0$ and a slope $\hat{\beta}_1$ such that the resulting line is as close as possible to the $n = 200$ data points. There are a number of ways of measuring *closeness*. However, by far the most common approach involves minimizing the *least squares* criterion, and we take that approach in this chapter. Alternative approaches will be considered in Chapter 6.

simple linear regression

intercept
slope
coefficient
parameter

least squares

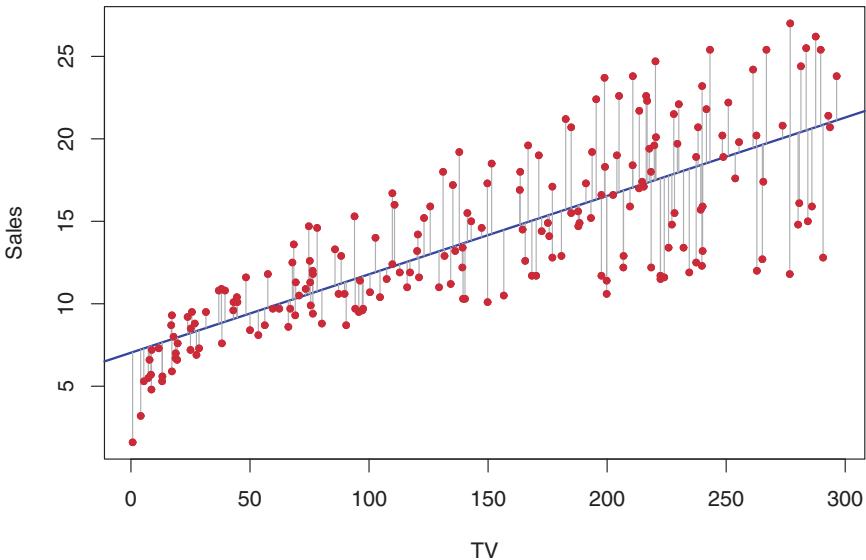


FIGURE 3.1. For the [Advertising](#) data, the least squares fit for the regression of [sales](#) onto [TV](#) is shown. The fit is found by minimizing the sum of squared errors. Each grey line segment represents an error, and the fit makes a compromise by averaging their squares. In this case a linear fit captures the essence of the relationship, although it is somewhat deficient in the left of the plot.

Let $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ be the prediction for Y based on the i th value of X . Then $e_i = y_i - \hat{y}_i$ represents the i th *residual*—this is the difference between the i th observed response value and the i th response value that is predicted by our linear model. We define the *residual sum of squares* (RSS) as

$$\text{RSS} = e_1^2 + e_2^2 + \dots + e_n^2,$$

residual
residual sum
of squares

or equivalently as

$$\text{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \dots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2. \quad (3.3)$$

The least squares approach chooses $\hat{\beta}_0$ and $\hat{\beta}_1$ to minimize the RSS. Using some calculus, one can show that the minimizers are

$$\begin{aligned} \hat{\beta}_1 &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}, \\ \hat{\beta}_0 &= \bar{y} - \hat{\beta}_1 \bar{x}, \end{aligned} \quad (3.4)$$

where $\bar{y} \equiv \frac{1}{n} \sum_{i=1}^n y_i$ and $\bar{x} \equiv \frac{1}{n} \sum_{i=1}^n x_i$ are the sample means. In other words, (3.4) defines the *least squares coefficient estimates* for simple linear regression.

Figure 3.1 displays the simple linear regression fit to the [Advertising](#) data, where $\hat{\beta}_0 = 7.03$ and $\hat{\beta}_1 = 0.0475$. In other words, according to

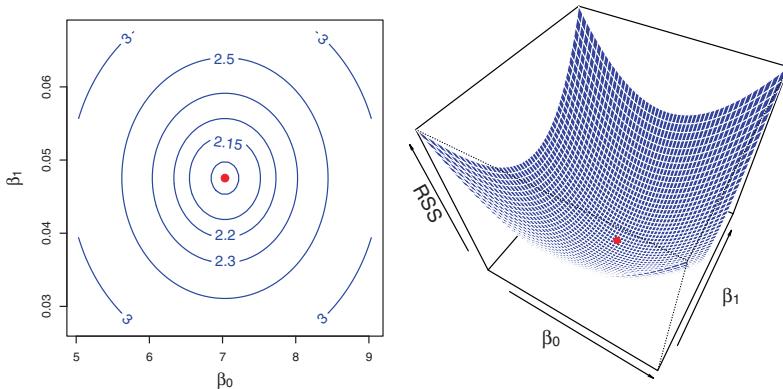


FIGURE 3.2. Contour and three-dimensional plots of the RSS on the Advertising data, using sales as the response and TV as the predictor. The red dots correspond to the least squares estimates $\hat{\beta}_0$ and $\hat{\beta}_1$, given by (3.4).

this approximation, an additional \$1,000 spent on TV advertising is associated with selling approximately 47.5 additional units of the product. In Figure 3.2, we have computed RSS for a number of values of β_0 and β_1 , using the advertising data with sales as the response and TV as the predictor. In each plot, the red dot represents the pair of least squares estimates $(\hat{\beta}_0, \hat{\beta}_1)$ given by (3.4). These values clearly minimize the RSS.

3.1.2 Assessing the Accuracy of the Coefficient Estimates

Recall from (2.1) that we assume that the *true* relationship between X and Y takes the form $Y = f(X) + \epsilon$ for some unknown function f , where ϵ is a mean-zero random error term. If f is to be approximated by a linear function, then we can write this relationship as

$$Y = \beta_0 + \beta_1 X + \epsilon. \quad (3.5)$$

Here β_0 is the intercept term—that is, the expected value of Y when $X = 0$, and β_1 is the slope—the average increase in Y associated with a one-unit increase in X . The error term is a catch-all for what we miss with this simple model: the true relationship is probably not linear, there may be other variables that cause variation in Y , and there may be measurement error. We typically assume that the error term is independent of X .

The model given by (3.5) defines the *population regression line*, which is the best linear approximation to the true relationship between X and Y .¹ The least squares regression coefficient estimates (3.4) characterize the *least squares line* (3.2). The left-hand panel of Figure 3.3 displays these

population
regression
line

least squares
line

¹The assumption of linearity is often a useful working model. However, despite what many textbooks might tell us, we seldom believe that the true relationship is linear.

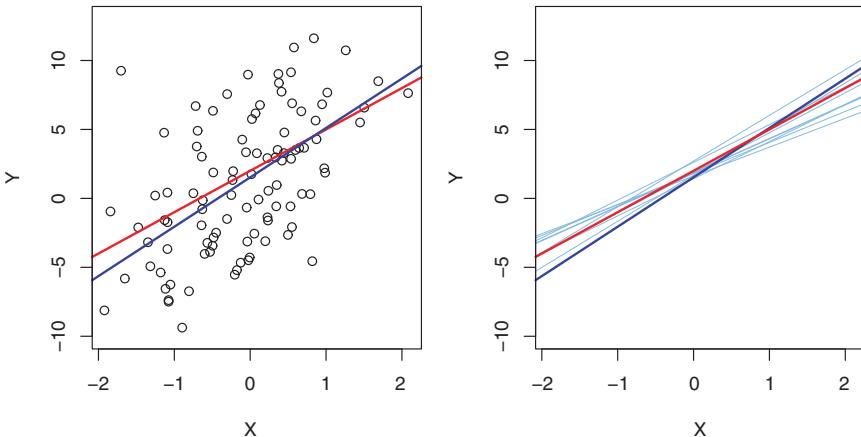


FIGURE 3.3. A simulated data set. Left: The red line represents the true relationship, $f(X) = 2 + 3X$, which is known as the population regression line. The blue line is the least squares line; it is the least squares estimate for $f(X)$ based on the observed data, shown in black. Right: The population regression line is again shown in red, and the least squares line in dark blue. In light blue, ten least squares lines are shown, each computed on the basis of a separate random set of observations. Each least squares line is different, but on average, the least squares lines are quite close to the population regression line.

two lines in a simple simulated example. We created 100 random X s, and generated 100 corresponding Y s from the model

$$Y = 2 + 3X + \epsilon, \quad (3.6)$$

where ϵ was generated from a normal distribution with mean zero. The red line in the left-hand panel of Figure 3.3 displays the *true* relationship, $f(X) = 2 + 3X$, while the blue line is the least squares estimate based on the observed data. The true relationship is generally not known for real data, but the least squares line can always be computed using the coefficient estimates given in (3.4). In other words, in real applications, we have access to a set of observations from which we can compute the least squares line; however, the population regression line is unobserved. In the right-hand panel of Figure 3.3 we have generated ten different data sets from the model given by (3.6) and plotted the corresponding ten least squares lines. Notice that different data sets generated from the same true model result in slightly different least squares lines, but the unobserved population regression line does not change.

At first glance, the difference between the population regression line and the least squares line may seem subtle and confusing. We only have one data set, and so what does it mean that two different lines describe the relationship between the predictor and the response? Fundamentally, the

concept of these two lines is a natural extension of the standard statistical approach of using information from a sample to estimate characteristics of a large population. For example, suppose that we are interested in knowing the population mean μ of some random variable Y . Unfortunately, μ is unknown, but we do have access to n observations from Y , which we can write as y_1, \dots, y_n , and which we can use to estimate μ . A reasonable estimate is $\hat{\mu} = \bar{y}$, where $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ is the sample mean. The sample mean and the population mean are different, but in general the sample mean will provide a good estimate of the population mean. In the same way, the unknown coefficients β_0 and β_1 in linear regression define the population regression line. We seek to estimate these unknown coefficients using $\hat{\beta}_0$ and $\hat{\beta}_1$ given in (3.4). These coefficient estimates define the least squares line.

The analogy between linear regression and estimation of the mean of a random variable is an apt one based on the concept of *bias*. If we use the sample mean $\hat{\mu}$ to estimate μ , this estimate is *unbiased*, in the sense that on average, we expect $\hat{\mu}$ to equal μ . What exactly does this mean? It means that on the basis of one particular set of observations y_1, \dots, y_n , $\hat{\mu}$ might overestimate μ , and on the basis of another set of observations, $\hat{\mu}$ might underestimate μ . But if we could average a huge number of estimates of μ obtained from a huge number of sets of observations, then this average would *exactly* equal μ . Hence, an unbiased estimator does not *systematically* over- or under-estimate the true parameter. The property of unbiasedness holds for the least squares coefficient estimates given by (3.4) as well: if we estimate β_0 and β_1 on the basis of a particular data set, then our estimates won't be exactly equal to β_0 and β_1 . But if we could average the estimates obtained over a huge number of data sets, then the average of these estimates would be spot on! In fact, we can see from the right-hand panel of Figure 3.3 that the average of many least squares lines, each estimated from a separate data set, is pretty close to the true population regression line.

We continue the analogy with the estimation of the population mean μ of a random variable Y . A natural question is as follows: how accurate is the sample mean $\hat{\mu}$ as an estimate of μ ? We have established that the average of $\hat{\mu}$'s over many data sets will be very close to μ , but that a single estimate $\hat{\mu}$ may be a substantial underestimate or overestimate of μ . How far off will that single estimate of $\hat{\mu}$ be? In general, we answer this question by computing the *standard error* of $\hat{\mu}$, written as $SE(\hat{\mu})$. We have the well-known formula

$$\text{Var}(\hat{\mu}) = SE(\hat{\mu})^2 = \frac{\sigma^2}{n}, \quad (3.7)$$

where σ is the standard deviation of each of the realizations y_i of Y .² Roughly speaking, the standard error tells us the average amount that this estimate $\hat{\mu}$ differs from the actual value of μ . Equation 3.7 also tells us how this deviation shrinks with n —the more observations we have, the smaller the standard error of $\hat{\mu}$. In a similar vein, we can wonder how close $\hat{\beta}_0$ and $\hat{\beta}_1$ are to the true values β_0 and β_1 . To compute the standard errors associated with $\hat{\beta}_0$ and $\hat{\beta}_1$, we use the following formulas:

$$\text{SE}(\hat{\beta}_0)^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right], \quad \text{SE}(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2}, \quad (3.8)$$

where $\sigma^2 = \text{Var}(\epsilon)$. For these formulas to be strictly valid, we need to assume that the errors ϵ_i for each observation are uncorrelated with common variance σ^2 . This is clearly not true in Figure 3.1, but the formula still turns out to be a good approximation. Notice in the formula that $\text{SE}(\hat{\beta}_1)$ is smaller when the x_i are more spread out; intuitively we have more *leverage* to estimate a slope when this is the case. We also see that $\text{SE}(\hat{\beta}_0)$ would be the same as $\text{SE}(\hat{\mu})$ if \bar{x} were zero (in which case $\hat{\beta}_0$ would be equal to \bar{y}). In general, σ^2 is not known, but can be estimated from the data. The estimate of σ is known as the *residual standard error*, and is given by the formula $\text{RSE} = \sqrt{\text{RSS}/(n - 2)}$. Strictly speaking, when σ^2 is estimated from the data we should write $\widehat{\text{SE}}(\hat{\beta}_1)$ to indicate that an estimate has been made, but for simplicity of notation we will drop this extra “hat”.

residual
standard
error

Standard errors can be used to compute *confidence intervals*. A 95 % confidence interval is defined as a range of values such that with 95 % probability, the range will contain the true unknown value of the parameter. The range is defined in terms of lower and upper limits computed from the sample of data. For linear regression, the 95 % confidence interval for β_1 approximately takes the form

$$\hat{\beta}_1 \pm 2 \cdot \text{SE}(\hat{\beta}_1). \quad (3.9)$$

confidence
interval

That is, there is approximately a 95 % chance that the interval

$$[\hat{\beta}_1 - 2 \cdot \text{SE}(\hat{\beta}_1), \hat{\beta}_1 + 2 \cdot \text{SE}(\hat{\beta}_1)] \quad (3.10)$$

will contain the true value of β_1 .³ Similarly, a confidence interval for β_0 approximately takes the form

$$\hat{\beta}_0 \pm 2 \cdot \text{SE}(\hat{\beta}_0). \quad (3.11)$$

²This formula holds provided that the n observations are uncorrelated.

³Approximately for several reasons. Equation 3.10 relies on the assumption that the errors are Gaussian. Also, the factor of 2 in front of the $\text{SE}(\hat{\beta}_1)$ term will vary slightly depending on the number of observations n in the linear regression. To be precise, rather than the number 2, (3.10) should contain the 97.5 % quantile of a t -distribution with $n - 2$ degrees of freedom. Details of how to compute the 95 % confidence interval precisely in R will be provided later in this chapter.

In the case of the advertising data, the 95 % confidence interval for β_0 is [6,130, 7,935] and the 95 % confidence interval for β_1 is [0.042, 0.053]. Therefore, we can conclude that in the absence of any advertising, sales will, on average, fall somewhere between 6,130 and 7,940 units. Furthermore, for each \$1,000 increase in television advertising, there will be an average increase in sales of between 42 and 53 units.

Standard errors can also be used to perform *hypothesis tests* on the coefficients. The most common hypothesis test involves testing the *null hypothesis* of

$$H_0 : \text{There is no relationship between } X \text{ and } Y \quad (3.12)$$

versus the *alternative hypothesis*

$$H_a : \text{There is some relationship between } X \text{ and } Y. \quad (3.13)$$

Mathematically, this corresponds to testing

$$H_0 : \beta_1 = 0$$

versus

$$H_a : \beta_1 \neq 0,$$

since if $\beta_1 = 0$ then the model (3.5) reduces to $Y = \beta_0 + \epsilon$, and X is not associated with Y . To test the null hypothesis, we need to determine whether $\hat{\beta}_1$, our estimate for β_1 , is sufficiently far from zero that we can be confident that β_1 is non-zero. How far is far enough? This of course depends on the accuracy of $\hat{\beta}_1$ —that is, it depends on $\text{SE}(\hat{\beta}_1)$. If $\text{SE}(\hat{\beta}_1)$ is small, then even relatively small values of $\hat{\beta}_1$ may provide strong evidence that $\beta_1 \neq 0$, and hence that there is a relationship between X and Y . In contrast, if $\text{SE}(\hat{\beta}_1)$ is large, then $\hat{\beta}_1$ must be large in absolute value in order for us to reject the null hypothesis. In practice, we compute a *t-statistic*,

$$t = \frac{\hat{\beta}_1 - 0}{\text{SE}(\hat{\beta}_1)}, \quad (3.14)$$

which measures the number of standard deviations that $\hat{\beta}_1$ is away from 0. If there really is no relationship between X and Y , then we expect that (3.14) will have a *t*-distribution with $n - 2$ degrees of freedom. The *t*-distribution has a bell shape and for values of n greater than approximately 30 it is quite similar to the normal distribution. Consequently, it is a simple matter to compute the probability of observing any value equal to $|t|$ or larger, assuming $\beta_1 = 0$. We call this probability the *p-value*. Roughly speaking, we interpret the p-value as follows: a small p-value indicates that it is unlikely to observe such a substantial association between the predictor and the response due to chance, in the absence of any real association between the predictor and the response. Hence, if we see a small p-value,

then we can infer that there is an association between the predictor and the response. We *reject the null hypothesis*—that is, we declare a relationship to exist between X and Y —if the p-value is small enough. Typical p-value cutoffs for rejecting the null hypothesis are 5 or 1%. When $n = 30$, these correspond to t-statistics (3.14) of around 2 and 2.75, respectively.

	Coefficient	Std. error	t-statistic	p-value
Intercept	7.0325	0.4578	15.36	< 0.0001
TV	0.0475	0.0027	17.67	< 0.0001

TABLE 3.1. For the `Advertising` data, coefficients of the least squares model for the regression of number of units sold on TV advertising budget. An increase of \$1,000 in the TV advertising budget is associated with an increase in sales by around 50 units (Recall that the `sales` variable is in thousands of units, and the `TV` variable is in thousands of dollars).

Table 3.1 provides details of the least squares model for the regression of number of units sold on TV advertising budget for the `Advertising` data. Notice that the coefficients for $\hat{\beta}_0$ and $\hat{\beta}_1$ are very large relative to their standard errors, so the t-statistics are also large; the probabilities of seeing such values if H_0 is true are virtually zero. Hence we can conclude that $\beta_0 \neq 0$ and $\beta_1 \neq 0$.⁴

3.1.3 Assessing the Accuracy of the Model

Once we have rejected the null hypothesis (3.12) in favor of the alternative hypothesis (3.13), it is natural to want to quantify *the extent to which the model fits the data*. The quality of a linear regression fit is typically assessed using two related quantities: the *residual standard error* (RSE) and the R^2 statistic.

Table 3.2 displays the RSE, the R^2 statistic, and the F-statistic (to be described in Section 3.2.2) for the linear regression of number of units sold on TV advertising budget.

Residual Standard Error

Recall from the model (3.5) that associated with each observation is an error term ϵ . Due to the presence of these error terms, even if we knew the true regression line (i.e. even if β_0 and β_1 were known), we would not be able to perfectly predict Y from X . The RSE is an estimate of the standard

⁴In Table 3.1, a small p-value for the intercept indicates that we can reject the null hypothesis that $\beta_0 = 0$, and a small p-value for `TV` indicates that we can reject the null hypothesis that $\beta_1 = 0$. Rejecting the latter null hypothesis allows us to conclude that there is a relationship between `TV` and `sales`. Rejecting the former allows us to conclude that in the absence of `TV` expenditure, `sales` are non-zero.

Quantity	Value
Residual standard error	3.26
R^2	0.612
F-statistic	312.1

TABLE 3.2. For the *Advertising* data, more information about the least squares model for the regression of number of units sold on TV advertising budget.

deviation of ϵ . Roughly speaking, it is the average amount that the response will deviate from the true regression line. It is computed using the formula

$$\text{RSE} = \sqrt{\frac{1}{n-2} \text{RSS}} = \sqrt{\frac{1}{n-2} \sum_{i=1}^n (y_i - \hat{y}_i)^2}. \quad (3.15)$$

Note that RSS was defined in Section 3.1.1, and is given by the formula

$$\text{RSS} = \sum_{i=1}^n (y_i - \hat{y}_i)^2. \quad (3.16)$$

In the case of the advertising data, we see from the linear regression output in Table 3.2 that the RSE is 3.26. In other words, actual sales in each market deviate from the true regression line by approximately 3,260 units, on average. Another way to think about this is that even if the model were correct and the true values of the unknown coefficients β_0 and β_1 were known exactly, any prediction of sales on the basis of TV advertising would still be off by about 3,260 units on average. Of course, whether or not 3,260 units is an acceptable prediction error depends on the problem context. In the advertising data set, the mean value of *sales* over all markets is approximately 14,000 units, and so the percentage error is $3,260/14,000 = 23\%$.

The RSE is considered a measure of the *lack of fit* of the model (3.5) to the data. If the predictions obtained using the model are very close to the true outcome values—that is, if $\hat{y}_i \approx y_i$ for $i = 1, \dots, n$ —then (3.15) will be small, and we can conclude that the model fits the data very well. On the other hand, if \hat{y}_i is very far from y_i for one or more observations, then the RSE may be quite large, indicating that the model doesn't fit the data well.

R^2 Statistic

The RSE provides an absolute measure of lack of fit of the model (3.5) to the data. But since it is measured in the units of Y , it is not always clear what constitutes a good RSE. The R^2 statistic provides an alternative measure of fit. It takes the form of a *proportion*—the proportion of variance explained—and so it always takes on a value between 0 and 1, and is independent of the scale of Y .

To calculate R^2 , we use the formula

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}} \quad (3.17)$$

where $\text{TSS} = \sum(y_i - \bar{y})^2$ is the *total sum of squares*, and RSS is defined in (3.16). TSS measures the total variance in the response Y , and can be thought of as the amount of variability inherent in the response before the regression is performed. In contrast, RSS measures the amount of variability that is left unexplained after performing the regression. Hence, $\text{TSS} - \text{RSS}$ measures the amount of variability in the response that is explained (or removed) by performing the regression, and R^2 measures the *proportion of variability in Y that can be explained using X* . An R^2 statistic that is close to 1 indicates that a large proportion of the variability in the response has been explained by the regression. A number near 0 indicates that the regression did not explain much of the variability in the response; this might occur because the linear model is wrong, or the inherent error σ^2 is high, or both. In Table 3.2, the R^2 was 0.61, and so just under two-thirds of the variability in **sales** is explained by a linear regression on **TV**.

The R^2 statistic (3.17) has an interpretational advantage over the RSE (3.15), since unlike the RSE, it always lies between 0 and 1. However, it can still be challenging to determine what is a *good* R^2 value, and in general, this will depend on the application. For instance, in certain problems in physics, we may know that the data truly comes from a linear model with a small residual error. In this case, we would expect to see an R^2 value that is extremely close to 1, and a substantially smaller R^2 value might indicate a serious problem with the experiment in which the data were generated. On the other hand, in typical applications in biology, psychology, marketing, and other domains, the linear model (3.5) is at best an extremely rough approximation to the data, and residual errors due to other unmeasured factors are often very large. In this setting, we would expect only a very small proportion of the variance in the response to be explained by the predictor, and an R^2 value well below 0.1 might be more realistic!

The R^2 statistic is a measure of the linear relationship between X and Y . Recall that *correlation*, defined as

$$\text{Cor}(X, Y) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}, \quad (3.18)$$

correlation

is also a measure of the linear relationship between X and Y .⁵ This suggests that we might be able to use $r = \text{Cor}(X, Y)$ instead of R^2 in order to assess the fit of the linear model. In fact, it can be shown that in the simple linear regression setting, $R^2 = r^2$. In other words, the squared correlation

⁵We note that in fact, the right-hand side of (3.18) is the sample correlation; thus, it would be more correct to write $\widehat{\text{Cor}}(X, Y)$; however, we omit the “hat” for ease of notation.

and the R^2 statistic are identical. However, in the next section we will discuss the multiple linear regression problem, in which we use several predictors simultaneously to predict the response. The concept of correlation between the predictors and the response does not extend automatically to this setting, since correlation quantifies the association between a single pair of variables rather than between a larger number of variables. We will see that R^2 fills this role.

3.2 Multiple Linear Regression

Simple linear regression is a useful approach for predicting a response on the basis of a single predictor variable. However, in practice we often have more than one predictor. For example, in the **Advertising** data, we have examined the relationship between sales and TV advertising. We also have data for the amount of money spent advertising on the radio and in newspapers, and we may want to know whether either of these two media is associated with sales. How can we extend our analysis of the advertising data in order to accommodate these two additional predictors?

One option is to run three separate simple linear regressions, each of which uses a different advertising medium as a predictor. For instance, we can fit a simple linear regression to predict sales on the basis of the amount spent on radio advertisements. Results are shown in Table 3.3 (top table). We find that a \$1,000 increase in spending on radio advertising is associated with an increase in sales by around 203 units. Table 3.3 (bottom table) contains the least squares coefficients for a simple linear regression of sales onto newspaper advertising budget. A \$1,000 increase in newspaper advertising budget is associated with an increase in sales by approximately 55 units.

However, the approach of fitting a separate simple linear regression model for each predictor is not entirely satisfactory. First of all, it is unclear how to make a single prediction of sales given levels of the three advertising media budgets, since each of the budgets is associated with a separate regression equation. Second, each of the three regression equations ignores the other two media in forming estimates for the regression coefficients. We will see shortly that if the media budgets are correlated with each other in the 200 markets that constitute our data set, then this can lead to very misleading estimates of the individual media effects on sales.

Instead of fitting a separate simple linear regression model for each predictor, a better approach is to extend the simple linear regression model (3.5) so that it can directly accommodate multiple predictors. We can do this by giving each predictor a separate slope coefficient in a single model. In general, suppose that we have p distinct predictors. Then the multiple linear regression model takes the form

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon, \quad (3.19)$$

Simple regression of `sales` on `radio`

	Coefficient	Std. error	t-statistic	p-value
Intercept	9.312	0.563	16.54	< 0.0001
radio	0.203	0.020	9.92	< 0.0001

Simple regression of `sales` on `newspaper`

	Coefficient	Std. error	t-statistic	p-value
Intercept	12.351	0.621	19.88	< 0.0001
newspaper	0.055	0.017	3.30	< 0.0001

TABLE 3.3. More simple linear regression models for the `Advertising` data. Coefficients of the simple linear regression model for number of units sold on Top: radio advertising budget and Bottom: newspaper advertising budget. A \$1,000 increase in spending on radio advertising is associated with an average increase in sales by around 203 units, while the same increase in spending on newspaper advertising is associated with an average increase in sales by around 55 units (Note that the `sales` variable is in thousands of units, and the `radio` and `newspaper` variables are in thousands of dollars).

where X_j represents the j th predictor and β_j quantifies the association between that variable and the response. We interpret β_j as the *average effect* on Y of a one unit increase in X_j , *holding all other predictors fixed*. In the advertising example, (3.19) becomes

$$\text{sales} = \beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{radio} + \beta_3 \times \text{newspaper} + \epsilon. \quad (3.20)$$

3.2.1 Estimating the Regression Coefficients

As was the case in the simple linear regression setting, the regression coefficients $\beta_0, \beta_1, \dots, \beta_p$ in (3.19) are unknown, and must be estimated. Given estimates $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$, we can make predictions using the formula

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p. \quad (3.21)$$

The parameters are estimated using the same least squares approach that we saw in the context of simple linear regression. We choose $\beta_0, \beta_1, \dots, \beta_p$ to minimize the sum of squared residuals

$$\begin{aligned} \text{RSS} &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2. \end{aligned} \quad (3.22)$$

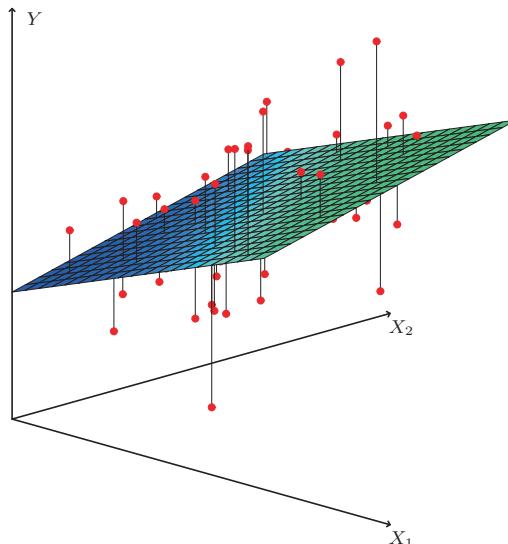


FIGURE 3.4. In a three-dimensional setting, with two predictors and one response, the least squares regression line becomes a plane. The plane is chosen to minimize the sum of the squared vertical distances between each observation (shown in red) and the plane.

The values $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$ that minimize (3.22) are the multiple least squares regression coefficient estimates. Unlike the simple linear regression estimates given in (3.4), the multiple regression coefficient estimates have somewhat complicated forms that are most easily represented using matrix algebra. For this reason, we do not provide them here. Any statistical software package can be used to compute these coefficient estimates, and later in this chapter we will show how this can be done in R. Figure 3.4 illustrates an example of the least squares fit to a toy data set with $p = 2$ predictors.

Table 3.4 displays the multiple regression coefficient estimates when TV, radio, and newspaper advertising budgets are used to predict product sales using the **Advertising** data. We interpret these results as follows: for a given amount of TV and newspaper advertising, spending an additional \$1,000 on radio advertising leads to an increase in sales by approximately 189 units. Comparing these coefficient estimates to those displayed in Tables 3.1 and 3.3, we notice that the multiple regression coefficient estimates for **TV** and **radio** are pretty similar to the simple linear regression coefficient estimates. However, while the **newspaper** regression coefficient estimate in Table 3.3 was significantly non-zero, the coefficient estimate for **newspaper** in the multiple regression model is close to zero, and the corresponding p-value is no longer significant, with a value around 0.86. This illustrates

	Coefficient	Std. error	t-statistic	p-value
Intercept	2.939	0.3119	9.42	< 0.0001
TV	0.046	0.0014	32.81	< 0.0001
radio	0.189	0.0086	21.89	< 0.0001
newspaper	-0.001	0.0059	-0.18	0.8599

TABLE 3.4. For the `Advertising` data, least squares coefficient estimates of the multiple linear regression of number of units sold on radio, TV, and newspaper advertising budgets.

that the simple and multiple regression coefficients can be quite different. This difference stems from the fact that in the simple regression case, the slope term represents the average effect of a \$1,000 increase in newspaper advertising, ignoring other predictors such as `TV` and `radio`. In contrast, in the multiple regression setting, the coefficient for `newspaper` represents the average effect of increasing newspaper spending by \$1,000 while holding `TV` and `radio` fixed.

Does it make sense for the multiple regression to suggest no relationship between `sales` and `newspaper` while the simple linear regression implies the opposite? In fact it does. Consider the correlation matrix for the three predictor variables and response variable, displayed in Table 3.5. Notice that the correlation between `radio` and `newspaper` is 0.35. This reveals a tendency to spend more on newspaper advertising in markets where more is spent on radio advertising. Now suppose that the multiple regression is correct and newspaper advertising has no direct impact on sales, but radio advertising does increase sales. Then in markets where we spend more on radio our sales will tend to be higher, and as our correlation matrix shows, we also tend to spend more on newspaper advertising in those same markets. Hence, in a simple linear regression which only examines `sales` versus `newspaper`, we will observe that higher values of `newspaper` tend to be associated with higher values of `sales`, even though newspaper advertising does not actually affect sales. So `newspaper` sales are a surrogate for `radio` advertising; `newspaper` gets “credit” for the effect of `radio` on `sales`.

This slightly counterintuitive result is very common in many real life situations. Consider an absurd example to illustrate the point. Running a regression of shark attacks versus ice cream sales for data collected at a given beach community over a period of time would show a positive relationship, similar to that seen between `sales` and `newspaper`. Of course no one (yet) has suggested that ice creams should be banned at beaches to reduce shark attacks. In reality, higher temperatures cause more people to visit the beach, which in turn results in more ice cream sales and more shark attacks. A multiple regression of attacks versus ice cream sales and temperature reveals that, as intuition implies, the former predictor is no longer significant after adjusting for temperature.

	TV	radio	newspaper	sales
TV	1.0000	0.0548	0.0567	0.7822
radio		1.0000	0.3541	0.5762
newspaper			1.0000	0.2283
sales				1.0000

TABLE 3.5. Correlation matrix for TV, radio, newspaper, and sales for the Advertising data.

3.2.2 Some Important Questions

When we perform multiple linear regression, we usually are interested in answering a few important questions.

1. Is at least one of the predictors X_1, X_2, \dots, X_p useful in predicting the response?
2. Do all the predictors help to explain Y , or is only a subset of the predictors useful?
3. How well does the model fit the data?
4. Given a set of predictor values, what response value should we predict, and how accurate is our prediction?

We now address each of these questions in turn.

One: Is There a Relationship Between the Response and Predictors?

Recall that in the simple linear regression setting, in order to determine whether there is a relationship between the response and the predictor we can simply check whether $\beta_1 = 0$. In the multiple regression setting with p predictors, we need to ask whether all of the regression coefficients are zero, i.e. whether $\beta_1 = \beta_2 = \dots = \beta_p = 0$. As in the simple linear regression setting, we use a hypothesis test to answer this question. We test the null hypothesis,

$$H_0 : \beta_1 = \beta_2 = \dots = \beta_p = 0$$

versus the alternative

$$H_a : \text{at least one } \beta_j \text{ is non-zero.}$$

This hypothesis test is performed by computing the *F-statistic*,

$$F = \frac{(\text{TSS} - \text{RSS})/p}{\text{RSS}/(n - p - 1)}, \quad (3.23)$$

F-statistic

Quantity	Value
Residual standard error	1.69
R^2	0.897
F-statistic	570

TABLE 3.6. More information about the least squares model for the regression of number of units sold on TV, newspaper, and radio advertising budgets in the **Advertising** data. Other information about this model was displayed in Table 3.4.

where, as with simple linear regression, $TSS = \sum(y_i - \bar{y})^2$ and $RSS = \sum(y_i - \hat{y}_i)^2$. If the linear model assumptions are correct, one can show that

$$E\{\text{RSS}/(n - p - 1)\} = \sigma^2$$

and that, provided H_0 is true,

$$E\{(TSS - RSS)/p\} = \sigma^2.$$

Hence, when there is no relationship between the response and predictors, one would expect the F-statistic to take on a value close to 1. On the other hand, if H_a is true, then $E\{(TSS - RSS)/p\} > \sigma^2$, so we expect F to be greater than 1.

The F-statistic for the multiple linear regression model obtained by regressing **sales** onto **radio**, **TV**, and **newspaper** is shown in Table 3.6. In this example the F-statistic is 570. Since this is far larger than 1, it provides compelling evidence against the null hypothesis H_0 . In other words, the large F-statistic suggests that at least one of the advertising media must be related to **sales**. However, what if the F-statistic had been closer to 1? How large does the F-statistic need to be before we can reject H_0 and conclude that there is a relationship? It turns out that the answer depends on the values of n and p . When n is large, an F-statistic that is just a little larger than 1 might still provide evidence against H_0 . In contrast, a larger F-statistic is needed to reject H_0 if n is small. When H_0 is true and the errors ϵ_i have a normal distribution, the F-statistic follows an F-distribution.⁶ For any given value of n and p , any statistical software package can be used to compute the p-value associated with the F-statistic using this distribution. Based on this p-value, we can determine whether or not to reject H_0 . For the advertising data, the p-value associated with the F-statistic in Table 3.6 is essentially zero, so we have extremely strong evidence that at least one of the media is associated with increased **sales**.

In (3.23) we are testing H_0 that all the coefficients are zero. Sometimes we want to test that a particular subset of q of the coefficients are zero. This corresponds to a null hypothesis

$$H_0 : \beta_{p-q+1} = \beta_{p-q+2} = \dots = \beta_p = 0,$$

⁶Even if the errors are not normally-distributed, the F-statistic approximately follows an F-distribution provided that the sample size n is large.

where for convenience we have put the variables chosen for omission at the end of the list. In this case we fit a second model that uses all the variables *except* those last q . Suppose that the residual sum of squares for that model is RSS_0 . Then the appropriate F-statistic is

$$F = \frac{(\text{RSS}_0 - \text{RSS})/q}{\text{RSS}/(n - p - 1)}. \quad (3.24)$$

Notice that in Table 3.4, for each individual predictor a t-statistic and a p-value were reported. These provide information about whether each individual predictor is related to the response, after adjusting for the other predictors. It turns out that each of these are exactly equivalent⁷ to the F-test that omits that single variable from the model, leaving all the others in—i.e. $q=1$ in (3.24). So it reports the *partial effect* of adding that variable to the model. For instance, as we discussed earlier, these p-values indicate that **TV** and **radio** are related to **sales**, but that there is no evidence that **newspaper** is associated with **sales**, in the presence of these two.

Given these individual p-values for each variable, why do we need to look at the overall F-statistic? After all, it seems likely that if any one of the p-values for the individual variables is very small, then *at least one of the predictors is related to the response*. However, this logic is flawed, especially when the number of predictors p is large.

For instance, consider an example in which $p = 100$ and $H_0 : \beta_1 = \beta_2 = \dots = \beta_p = 0$ is true, so no variable is truly associated with the response. In this situation, about 5% of the p-values associated with each variable (of the type shown in Table 3.4) will be below 0.05 by chance. In other words, we expect to see approximately five *small* p-values even in the absence of any true association between the predictors and the response. In fact, we are almost guaranteed that we will observe at least one p-value below 0.05 by chance! Hence, if we use the individual t-statistics and associated p-values in order to decide whether or not there is any association between the variables and the response, there is a very high chance that we will incorrectly conclude that there is a relationship. However, the F-statistic does not suffer from this problem because it adjusts for the number of predictors. Hence, if H_0 is true, there is only a 5% chance that the F-statistic will result in a p-value below 0.05, regardless of the number of predictors or the number of observations.

The approach of using an F-statistic to test for any association between the predictors and the response works when p is relatively small, and certainly small compared to n . However, sometimes we have a very large number of variables. If $p > n$ then there are more coefficients β_j to estimate than observations from which to estimate them. In this case we cannot even fit the multiple linear regression model using least squares, so the

⁷The square of each t-statistic is the corresponding F-statistic.

F-statistic cannot be used, and neither can most of the other concepts that we have seen so far in this chapter. When p is large, some of the approaches discussed in the next section, such as *forward selection*, can be used. This *high-dimensional* setting is discussed in greater detail in Chapter 6.

high-dimensional

Two: Deciding on Important Variables

As discussed in the previous section, the first step in a multiple regression analysis is to compute the F-statistic and to examine the associated p-value. If we conclude on the basis of that p-value that at least one of the predictors is related to the response, then it is natural to wonder *which* are the guilty ones! We could look at the individual p-values as in Table 3.4, but as discussed, if p is large we are likely to make some false discoveries.

It is possible that all of the predictors are associated with the response, but it is more often the case that the response is only related to a subset of the predictors. The task of determining which predictors are associated with the response, in order to fit a single model involving only those predictors, is referred to as *variable selection*. The variable selection problem is studied extensively in Chapter 6, and so here we will provide only a brief outline of some classical approaches.

variable selection

Ideally, we would like to perform variable selection by trying out a lot of different models, each containing a different subset of the predictors. For instance, if $p = 2$, then we can consider four models: (1) a model containing no variables, (2) a model containing X_1 only, (3) a model containing X_2 only, and (4) a model containing both X_1 and X_2 . We can then select the *best* model out of all of the models that we have considered. How do we determine which model is best? Various statistics can be used to judge the quality of a model. These include *Mallow's C_p* , *Akaike information criterion* (AIC), *Bayesian information criterion* (BIC), and *adjusted R^2* . These are discussed in more detail in Chapter 6. We can also determine which model is best by plotting various model outputs, such as the residuals, in order to search for patterns.

Mallow's C_p
Akaike
information
criterion
Bayesian
information
criterion
adjusted R^2

Unfortunately, there are a total of 2^p models that contain subsets of p variables. This means that even for moderate p , trying out every possible subset of the predictors is infeasible. For instance, we saw that if $p = 2$, then there are $2^2 = 4$ models to consider. But if $p = 30$, then we must consider $2^{30} = 1,073,741,824$ models! This is not practical. Therefore, unless p is very small, we cannot consider all 2^p models, and instead we need an automated and efficient approach to choose a smaller set of models to consider. There are three classical approaches for this task:

- *Forward selection*. We begin with the *null model*—a model that contains an intercept but no predictors. We then fit p simple linear regressions and add to the null model the variable that results in the lowest RSS. We then add to that model the variable that results

forward
selection
null model

in the lowest RSS for the new two-variable model. This approach is continued until some stopping rule is satisfied.

- *Backward selection.* We start with all variables in the model, and remove the variable with the largest p-value—that is, the variable that is the least statistically significant. The new $(p - 1)$ -variable model is fit, and the variable with the largest p-value is removed. This procedure continues until a stopping rule is reached. For instance, we may stop when all remaining variables have a p-value below some threshold.
- *Mixed selection.* This is a combination of forward and backward selection. We start with no variables in the model, and as with forward selection, we add the variable that provides the best fit. We continue to add variables one-by-one. Of course, as we noted with the **Advertising** example, the p-values for variables can become larger as new predictors are added to the model. Hence, if at any point the p-value for one of the variables in the model rises above a certain threshold, then we remove that variable from the model. We continue to perform these forward and backward steps until all variables in the model have a sufficiently low p-value, and all variables outside the model would have a large p-value if added to the model.

Backward selection cannot be used if $p > n$, while forward selection can always be used. Forward selection is a greedy approach, and might include variables early that later become redundant. Mixed selection can remedy this.

Three: Model Fit

Two of the most common numerical measures of model fit are the RSE and R^2 , the fraction of variance explained. These quantities are computed and interpreted in the same fashion as for simple linear regression.

Recall that in simple regression, R^2 is the square of the correlation of the response and the variable. In multiple linear regression, it turns out that it equals $\text{Cor}(Y, \hat{Y})^2$, the square of the correlation between the response and the fitted linear model; in fact one property of the fitted linear model is that it maximizes this correlation among all possible linear models.

An R^2 value close to 1 indicates that the model explains a large portion of the variance in the response variable. As an example, we saw in Table 3.6 that for the **Advertising** data, the model that uses all three advertising media to predict **sales** has an R^2 of 0.8972. On the other hand, the model that uses only **TV** and **radio** to predict **sales** has an R^2 value of 0.89719. In other words, there is a *small* increase in R^2 if we include newspaper advertising in the model that already contains TV and radio advertising, even though we saw earlier that the p-value for newspaper advertising in Table 3.4 is not significant. It turns out that R^2 will always increase when more variables

are added to the model, even if those variables are only weakly associated with the response. This is due to the fact that adding another variable to the least squares equations must allow us to fit the training data (though not necessarily the testing data) more accurately. Thus, the R^2 statistic, which is also computed on the training data, must increase. The fact that adding newspaper advertising to the model containing only TV and radio advertising leads to just a tiny increase in R^2 provides additional evidence that `newspaper` can be dropped from the model. Essentially, `newspaper` provides no real improvement in the model fit to the training samples, and its inclusion will likely lead to poor results on independent test samples due to overfitting.

In contrast, the model containing only `TV` as a predictor had an R^2 of 0.61 (Table 3.2). Adding `radio` to the model leads to a substantial improvement in R^2 . This implies that a model that uses TV and radio expenditures to predict sales is substantially better than one that uses only TV advertising. We could further quantify this improvement by looking at the p-value for the `radio` coefficient in a model that contains only `TV` and `radio` as predictors.

The model that contains only `TV` and `radio` as predictors has an RSE of 1.681, and the model that also contains `newspaper` as a predictor has an RSE of 1.686 (Table 3.6). In contrast, the model that contains only `TV` has an RSE of 3.26 (Table 3.2). This corroborates our previous conclusion that a model that uses TV and radio expenditures to predict sales is much more accurate (on the training data) than one that only uses TV spending. Furthermore, given that TV and radio expenditures are used as predictors, there is no point in also using newspaper spending as a predictor in the model. The observant reader may wonder how RSE can increase when `newspaper` is added to the model given that RSS must decrease. In general RSE is defined as

$$\text{RSE} = \sqrt{\frac{1}{n-p-1} \text{RSS}}, \quad (3.25)$$

which simplifies to (3.15) for a simple linear regression. Thus, models with more variables can have higher RSE if the decrease in RSS is small relative to the increase in p .

In addition to looking at the RSE and R^2 statistics just discussed, it can be useful to plot the data. Graphical summaries can reveal problems with a model that are not visible from numerical statistics. For example, Figure 3.5 displays a three-dimensional plot of `TV` and `radio` versus `sales`. We see that some observations lie above and some observations lie below the least squares regression plane. In particular, the linear model seems to overestimate `sales` for instances in which most of the advertising money was spent exclusively on either `TV` or `radio`. It underestimates `sales` for instances where the budget was split between the two media. This pronounced non-linear pattern cannot be modeled accurately using linear re-

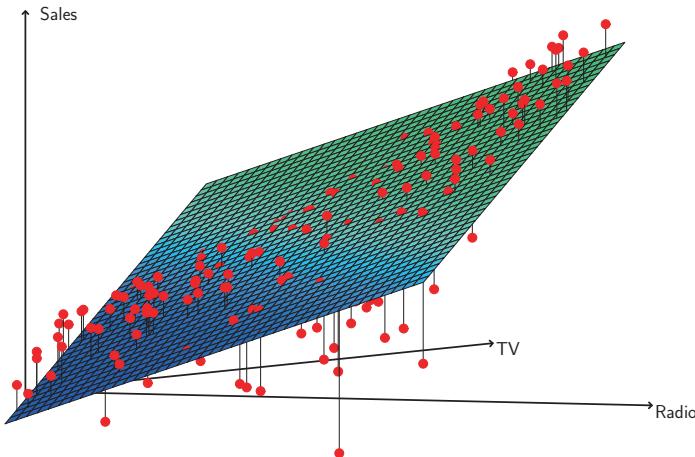


FIGURE 3.5. For the *Advertising* data, a linear regression fit to `sales` using `TV` and `radio` as predictors. From the pattern of the residuals, we can see that there is a pronounced non-linear relationship in the data. The positive residuals (those visible above the surface), tend to lie along the 45-degree line, where `TV` and `Radio` budgets are split evenly. The negative residuals (most not visible), tend to lie away from this line, where budgets are more lopsided.

gression. It suggests a *synergy* or *interaction* effect between the advertising media, whereby combining the media together results in a bigger boost to sales than using any single medium. In Section 3.3.2, we will discuss extending the linear model to accommodate such synergistic effects through the use of interaction terms.

Four: Predictions

Once we have fit the multiple regression model, it is straightforward to apply (3.21) in order to predict the response Y on the basis of a set of values for the predictors X_1, X_2, \dots, X_p . However, there are three sorts of uncertainty associated with this prediction.

1. The coefficient estimates $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$ are estimates for $\beta_0, \beta_1, \dots, \beta_p$. That is, the *least squares plane*

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_p X_p$$

is only an estimate for the *true population regression plane*

$$f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

The inaccuracy in the coefficient estimates is related to the *reducible error* from Chapter 2. We can compute a *confidence interval* in order to determine how close \hat{Y} will be to $f(X)$.

2. Of course, in practice assuming a linear model for $f(X)$ is almost always an approximation of reality, so there is an additional source of potentially reducible error which we call *model bias*. So when we use a linear model, we are in fact estimating the best linear approximation to the true surface. However, here we will ignore this discrepancy, and operate as if the linear model were correct.
3. Even if we knew $f(X)$ —that is, even if we knew the true values for $\beta_0, \beta_1, \dots, \beta_p$ —the response value cannot be predicted perfectly because of the random error ϵ in the model (3.21). In Chapter 2, we referred to this as the *irreducible error*. How much will Y vary from \hat{Y} ? We use *prediction intervals* to answer this question. Prediction intervals are always wider than confidence intervals, because they incorporate both the error in the estimate for $f(X)$ (the reducible error) and the uncertainty as to how much an individual point will differ from the population regression plane (the irreducible error).

We use a *confidence interval* to quantify the uncertainty surrounding the *average sales* over a large number of cities. For example, given that \$100,000 is spent on **TV** advertising and \$20,000 is spent on **radio** advertising in each city, the 95 % confidence interval is [10,985, 11,528]. We interpret this to mean that 95 % of intervals of this form will contain the true value of $f(X)$.⁸ On the other hand, a *prediction interval* can be used to quantify the uncertainty surrounding *sales* for a *particular* city. Given that \$100,000 is spent on **TV** advertising and \$20,000 is spent on **radio** advertising in that city the 95 % prediction interval is [7,930, 14,580]. We interpret this to mean that 95 % of intervals of this form will contain the true value of Y for this city. Note that both intervals are centered at 11,256, but that the prediction interval is substantially wider than the confidence interval, reflecting the increased uncertainty about *sales* for a given city in comparison to the average *sales* over many locations.

confidence interval

prediction interval

3.3 Other Considerations in the Regression Model

3.3.1 Qualitative Predictors

In our discussion so far, we have assumed that all variables in our linear regression model are *quantitative*. But in practice, this is not necessarily the case; often some predictors are *qualitative*.

⁸In other words, if we collect a large number of data sets like the **Advertising** data set, and we construct a confidence interval for the average *sales* on the basis of each data set (given \$100,000 in **TV** and \$20,000 in **radio** advertising), then 95 % of these confidence intervals will contain the true value of average *sales*.

For example, the `Credit` data set displayed in Figure 3.6 records `balance` (average credit card debt for a number of individuals) as well as several quantitative predictors: `age`, `cards` (number of credit cards), `education` (years of education), `income` (in thousands of dollars), `limit` (credit limit), and `rating` (credit rating). Each panel of Figure 3.6 is a scatterplot for a pair of variables whose identities are given by the corresponding row and column labels. For example, the scatterplot directly to the right of the word “Balance” depicts `balance` versus `age`, while the plot directly to the right of “Age” corresponds to `age` versus `cards`. In addition to these quantitative variables, we also have four qualitative variables: `gender`, `student` (student status), `status` (marital status), and `ethnicity` (Caucasian, African American or Asian).

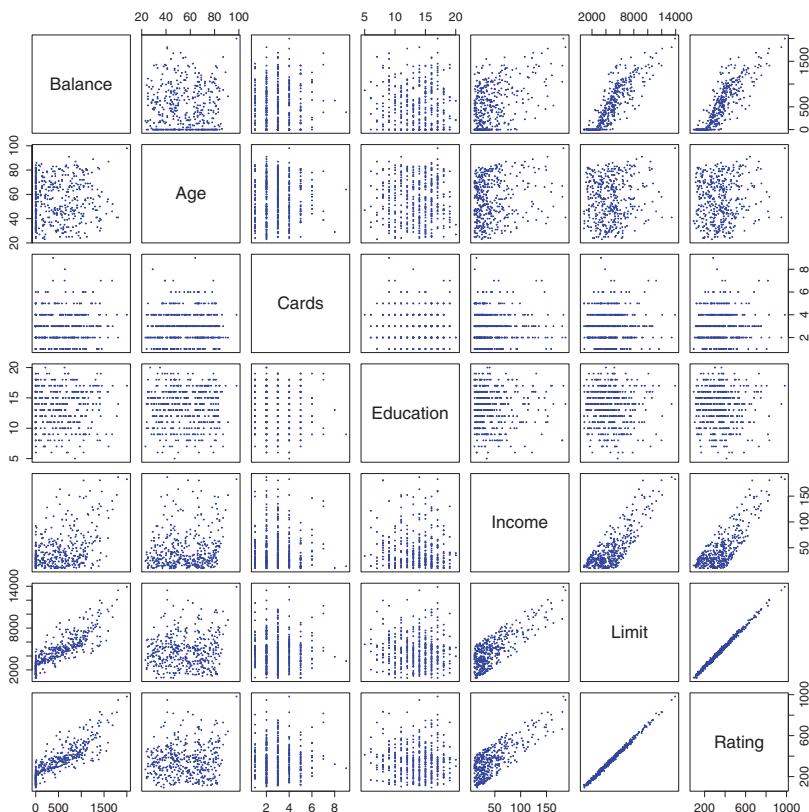


FIGURE 3.6. The `Credit` data set contains information about `balance`, `age`, `cards`, `education`, `income`, `limit`, and `rating` for a number of potential customers.

	Coefficient	Std. error	t-statistic	p-value
Intercept	509.80	33.13	15.389	< 0.0001
gender [Female]	19.73	46.05	0.429	0.6690

TABLE 3.7. Least squares coefficient estimates associated with the regression of balance onto gender in the Credit data set. The linear model is given in (3.27). That is, gender is encoded as a dummy variable, as in (3.26).

Predictors with Only Two Levels

Suppose that we wish to investigate differences in credit card balance between males and females, ignoring the other variables for the moment. If a qualitative predictor (also known as a *factor*) only has two *levels*, or possible values, then incorporating it into a regression model is very simple. We simply create an indicator or *dummy variable* that takes on two possible numerical values. For example, based on the gender variable, we can create a new variable that takes the form

$$x_i = \begin{cases} 1 & \text{if } i\text{th person is female} \\ 0 & \text{if } i\text{th person is male,} \end{cases} \quad (3.26)$$

and use this variable as a predictor in the regression equation. This results in the model

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i\text{th person is female} \\ \beta_0 + \epsilon_i & \text{if } i\text{th person is male.} \end{cases} \quad (3.27)$$

Now β_0 can be interpreted as the average credit card balance among males, $\beta_0 + \beta_1$ as the average credit card balance among females, and β_1 as the average difference in credit card balance between females and males.

Table 3.7 displays the coefficient estimates and other information associated with the model (3.27). The average credit card debt for males is estimated to be \$509.80, whereas females are estimated to carry \$19.73 in additional debt for a total of $\$509.80 + \$19.73 = \$529.53$. However, we notice that the p-value for the dummy variable is very high. This indicates that there is no statistical evidence of a difference in average credit card balance between the genders.

The decision to code females as 1 and males as 0 in (3.27) is arbitrary, and has no effect on the regression fit, but does alter the interpretation of the coefficients. If we had coded males as 1 and females as 0, then the estimates for β_0 and β_1 would have been 529.53 and -19.73 , respectively, leading once again to a prediction of credit card debt of $\$529.53 - \$19.73 = \$509.80$ for males and a prediction of \$529.53 for females. Alternatively, instead of a 0/1 coding scheme, we could create a dummy variable

$$x_i = \begin{cases} 1 & \text{if } i\text{th person is female} \\ -1 & \text{if } i\text{th person is male} \end{cases}$$

and use this variable in the regression equation. This results in the model

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i\text{th person is female} \\ \beta_0 - \beta_1 + \epsilon_i & \text{if } i\text{th person is male.} \end{cases}$$

Now β_0 can be interpreted as the overall average credit card balance (ignoring the gender effect), and β_1 is the amount that females are above the average and males are below the average. In this example, the estimate for β_0 would be \$519.665, halfway between the male and female averages of \$509.80 and \$529.53. The estimate for β_1 would be \$9.865, which is half of \$19.73, the average difference between females and males. It is important to note that the final predictions for the credit balances of males and females will be identical regardless of the coding scheme used. The only difference is in the way that the coefficients are interpreted.

Qualitative Predictors with More than Two Levels

When a qualitative predictor has more than two levels, a single dummy variable cannot represent all possible values. In this situation, we can create additional dummy variables. For example, for the **ethnicity** variable we create two dummy variables. The first could be

$$x_{i1} = \begin{cases} 1 & \text{if } i\text{th person is Asian} \\ 0 & \text{if } i\text{th person is not Asian,} \end{cases} \quad (3.28)$$

and the second could be

$$x_{i2} = \begin{cases} 1 & \text{if } i\text{th person is Caucasian} \\ 0 & \text{if } i\text{th person is not Caucasian.} \end{cases} \quad (3.29)$$

Then both of these variables can be used in the regression equation, in order to obtain the model

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i\text{th person is Asian} \\ \beta_0 + \beta_2 + \epsilon_i & \text{if } i\text{th person is Caucasian} \\ \beta_0 + \epsilon_i & \text{if } i\text{th person is African American.} \end{cases} \quad (3.30)$$

Now β_0 can be interpreted as the average credit card balance for African Americans, β_1 can be interpreted as the difference in the average balance between the Asian and African American categories, and β_2 can be interpreted as the difference in the average balance between the Caucasian and

	Coefficient	Std. error	t-statistic	p-value
Intercept	531.00	46.32	11.464	< 0.0001
ethnicity [Asian]	-18.69	65.02	-0.287	0.7740
ethnicity [Caucasian]	-12.50	56.68	-0.221	0.8260

TABLE 3.8. Least squares coefficient estimates associated with the regression of `balance` onto `ethnicity` in the `Credit` data set. The linear model is given in (3.30). That is, ethnicity is encoded via two dummy variables (3.28) and (3.29).

African American categories. There will always be one fewer dummy variable than the number of levels. The level with no dummy variable—African American in this example—is known as the *baseline*.

From Table 3.8, we see that the estimated `balance` for the baseline, African American, is \$531.00. It is estimated that the Asian category will have \$18.69 less debt than the African American category, and that the Caucasian category will have \$12.50 less debt than the African American category. However, the p-values associated with the coefficient estimates for the two dummy variables are very large, suggesting no statistical evidence of a real difference in credit card balance between the ethnicities. Once again, the level selected as the baseline category is arbitrary, and the final predictions for each group will be the same regardless of this choice. However, the coefficients and their p-values do depend on the choice of dummy variable coding. Rather than rely on the individual coefficients, we can use an F-test to test $H_0 : \beta_1 = \beta_2 = 0$; this does not depend on the coding. This F-test has a p-value of 0.96, indicating that we cannot reject the null hypothesis that there is no relationship between `balance` and `ethnicity`.

Using this dummy variable approach presents no difficulties when incorporating both quantitative and qualitative predictors. For example, to regress `balance` on both a quantitative variable such as `income` and a qualitative variable such as `student`, we must simply create a dummy variable for `student` and then fit a multiple regression model using `income` and the dummy variable as predictors for credit card balance.

There are many different ways of coding qualitative variables besides the dummy variable approach taken here. All of these approaches lead to equivalent model fits, but the coefficients are different and have different interpretations, and are designed to measure particular *contrasts*. This topic is beyond the scope of the book, and so we will not pursue it further.

*baseline**contrast*

3.3.2 Extensions of the Linear Model

The standard linear regression model (3.19) provides interpretable results and works quite well on many real-world problems. However, it makes several highly restrictive assumptions that are often violated in practice. Two of the most important assumptions state that the relationship between the predictors and response are *additive* and *linear*. The additive assumption

additive linear

means that the effect of changes in a predictor X_j on the response Y is independent of the values of the other predictors. The linear assumption states that the change in the response Y due to a one-unit change in X_j is constant, regardless of the value of X_j . In this book, we examine a number of sophisticated methods that relax these two assumptions. Here, we briefly examine some common classical approaches for extending the linear model.

Removing the Additive Assumption

In our previous analysis of the **Advertising** data, we concluded that both **TV** and **radio** seem to be associated with **sales**. The linear models that formed the basis for this conclusion assumed that the effect on **sales** of increasing one advertising medium is independent of the amount spent on the other media. For example, the linear model (3.20) states that the average effect on **sales** of a one-unit increase in **TV** is always β_1 , regardless of the amount spent on **radio**.

However, this simple model may be incorrect. Suppose that spending money on radio advertising actually increases the effectiveness of TV advertising, so that the slope term for **TV** should increase as **radio** increases. In this situation, given a fixed budget of \$100,000, spending half on **radio** and half on **TV** may increase **sales** more than allocating the entire amount to either **TV** or to **radio**. In marketing, this is known as a *synergy* effect, and in statistics it is referred to as an *interaction* effect. Figure 3.5 suggests that such an effect may be present in the advertising data. Notice that when levels of either **TV** or **radio** are low, then the true **sales** are lower than predicted by the linear model. But when advertising is split between the two media, then the model tends to underestimate **sales**.

Consider the standard linear regression model with two variables,

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon.$$

According to this model, if we increase X_1 by one unit, then Y will increase by an average of β_1 units. Notice that the presence of X_2 does not alter this statement—that is, regardless of the value of X_2 , a one-unit increase in X_1 will lead to a β_1 -unit increase in Y . One way of extending this model to allow for interaction effects is to include a third predictor, called an *interaction term*, which is constructed by computing the product of X_1 and X_2 . This results in the model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon. \quad (3.31)$$

How does inclusion of this interaction term relax the additive assumption? Notice that (3.31) can be rewritten as

$$\begin{aligned} Y &= \beta_0 + (\beta_1 + \beta_3 X_2) X_1 + \beta_2 X_2 + \epsilon \\ &= \beta_0 + \tilde{\beta}_1 X_1 + \beta_2 X_2 + \epsilon \end{aligned} \quad (3.32)$$

	Coefficient	Std. error	t-statistic	p-value
Intercept	6.7502	0.248	27.23	< 0.0001
TV	0.0191	0.002	12.70	< 0.0001
radio	0.0289	0.009	3.24	0.0014
TV×radio	0.0011	0.000	20.73	< 0.0001

TABLE 3.9. For the `Advertising` data, least squares coefficient estimates associated with the regression of `sales` onto `TV` and `radio`, with an interaction term, as in (3.33).

where $\tilde{\beta}_1 = \beta_1 + \beta_3 X_2$. Since $\tilde{\beta}_1$ changes with X_2 , the effect of X_1 on Y is no longer constant: adjusting X_2 will change the impact of X_1 on Y .

For example, suppose that we are interested in studying the productivity of a factory. We wish to predict the number of `units` produced on the basis of the number of production `lines` and the total number of `workers`. It seems likely that the effect of increasing the number of production lines will depend on the number of workers, since if no workers are available to operate the lines, then increasing the number of lines will not increase production. This suggests that it would be appropriate to include an interaction term between `lines` and `workers` in a linear model to predict `units`. Suppose that when we fit the model, we obtain

$$\begin{aligned}\text{units} &\approx 1.2 + 3.4 \times \text{lines} + 0.22 \times \text{workers} + 1.4 \times (\text{lines} \times \text{workers}) \\ &= 1.2 + (3.4 + 1.4 \times \text{workers}) \times \text{lines} + 0.22 \times \text{workers}.\end{aligned}$$

In other words, adding an additional line will increase the number of units produced by $3.4 + 1.4 \times \text{workers}$. Hence the more `workers` we have, the stronger will be the effect of `lines`.

We now return to the `Advertising` example. A linear model that uses `radio`, `TV`, and an interaction between the two to predict `sales` takes the form

$$\begin{aligned}\text{sales} &= \beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{radio} + \beta_3 \times (\text{radio} \times \text{TV}) + \epsilon \\ &= \beta_0 + (\beta_1 + \beta_3 \times \text{radio}) \times \text{TV} + \beta_2 \times \text{radio} + \epsilon.\end{aligned}\quad (3.33)$$

We can interpret β_3 as the increase in the effectiveness of TV advertising for a one unit increase in radio advertising (or vice-versa). The coefficients that result from fitting the model (3.33) are given in Table 3.9.

The results in Table 3.9 strongly suggest that the model that includes the interaction term is superior to the model that contains only *main effects*. The p-value for the interaction term, `TV×radio`, is extremely low, indicating that there is strong evidence for $H_a : \beta_3 \neq 0$. In other words, it is clear that the true relationship is not additive. The R^2 for the model (3.33) is 96.8 %, compared to only 89.7 % for the model that predicts `sales` using `TV` and `radio` without an interaction term. This means that $(96.8 - 89.7)/(100 - 89.7) = 69\%$ of the variability in `sales` that remains after fitting the additive model has been explained by the interaction term. The coefficient

main effect

estimates in Table 3.9 suggest that an increase in TV advertising of \$1,000 is associated with increased sales of $(\hat{\beta}_1 + \hat{\beta}_3 \times \text{radio}) \times 1,000 = 19 + 1.1 \times \text{radio}$ units. And an increase in radio advertising of \$1,000 will be associated with an increase in sales of $(\hat{\beta}_2 + \hat{\beta}_3 \times \text{TV}) \times 1,000 = 29 + 1.1 \times \text{TV}$ units.

In this example, the p-values associated with **TV**, **radio**, and the interaction term all are statistically significant (Table 3.9), and so it is obvious that all three variables should be included in the model. However, it is sometimes the case that an interaction term has a very small p-value, but the associated main effects (in this case, **TV** and **radio**) do not. The *hierarchical principle* states that *if we include an interaction in a model, we should also include the main effects, even if the p-values associated with their coefficients are not significant*. In other words, if the interaction between X_1 and X_2 seems important, then we should include both X_1 and X_2 in the model even if their coefficient estimates have large p-values. The rationale for this principle is that if $X_1 \times X_2$ is related to the response, then whether or not the coefficients of X_1 or X_2 are exactly zero is of little interest. Also $X_1 \times X_2$ is typically correlated with X_1 and X_2 , and so leaving them out tends to alter the meaning of the interaction.

In the previous example, we considered an interaction between **TV** and **radio**, both of which are quantitative variables. However, the concept of interactions applies just as well to qualitative variables, or to a combination of quantitative and qualitative variables. In fact, an interaction between a qualitative variable and a quantitative variable has a particularly nice interpretation. Consider the **Credit** data set from Section 3.3.1, and suppose that we wish to predict **balance** using the **income** (quantitative) and **student** (qualitative) variables. In the absence of an interaction term, the model takes the form

$$\begin{aligned} \text{balance}_i &\approx \beta_0 + \beta_1 \times \text{income}_i + \begin{cases} \beta_2 & \text{if } i\text{th person is a student} \\ 0 & \text{if } i\text{th person is not a student} \end{cases} \\ &= \beta_1 \times \text{income}_i + \begin{cases} \beta_0 + \beta_2 & \text{if } i\text{th person is a student} \\ \beta_0 & \text{if } i\text{th person is not a student.} \end{cases} \end{aligned} \tag{3.34}$$

Notice that this amounts to fitting two parallel lines to the data, one for students and one for non-students. The lines for students and non-students have different intercepts, $\beta_0 + \beta_2$ versus β_0 , but the same slope, β_1 . This is illustrated in the left-hand panel of Figure 3.7. The fact that the lines are parallel means that the average effect on **balance** of a one-unit increase in **income** does not depend on whether or not the individual is a student. This represents a potentially serious limitation of the model, since in fact a change in **income** may have a very different effect on the credit card balance of a student versus a non-student.

This limitation can be addressed by adding an interaction variable, created by multiplying **income** with the dummy variable for **student**. Our

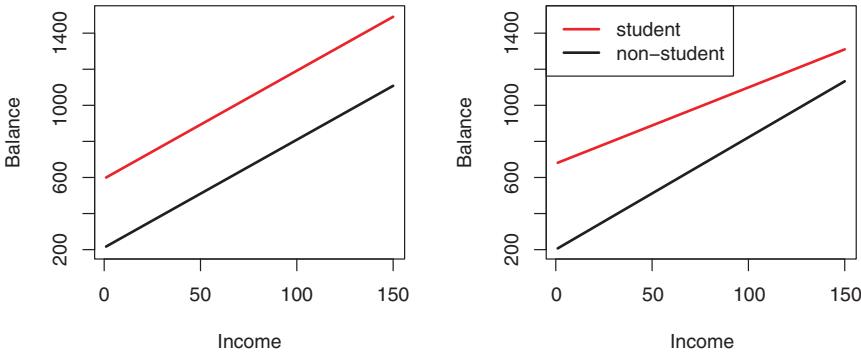


FIGURE 3.7. For the `Credit` data, the least squares lines are shown for prediction of `balance` from `income` for students and non-students. Left: The model (3.34) was fit. There is no interaction between `income` and `student`. Right: The model (3.35) was fit. There is an interaction term between `income` and `student`.

model now becomes

$$\begin{aligned} \text{balance}_i &\approx \beta_0 + \beta_1 \times \text{income}_i + \begin{cases} \beta_2 + \beta_3 \times \text{income}_i & \text{if student} \\ 0 & \text{if not student} \end{cases} \\ &= \begin{cases} (\beta_0 + \beta_2) + (\beta_1 + \beta_3) \times \text{income}_i & \text{if student} \\ \beta_0 + \beta_1 \times \text{income}_i & \text{if not student} \end{cases} \end{aligned} \quad (3.35)$$

Once again, we have two different regression lines for the students and the non-students. But now those regression lines have different intercepts, $\beta_0 + \beta_2$ versus β_0 , as well as different slopes, $\beta_1 + \beta_3$ versus β_1 . This allows for the possibility that changes in income may affect the credit card balances of students and non-students differently. The right-hand panel of Figure 3.7 shows the estimated relationships between `income` and `balance` for students and non-students in the model (3.35). We note that the slope for students is lower than the slope for non-students. This suggests that increases in income are associated with smaller increases in credit card balance among students as compared to non-students.

Non-linear Relationships

As discussed previously, the linear regression model (3.19) assumes a linear relationship between the response and predictors. But in some cases, the true relationship between the response and the predictors may be non-linear. Here we present a very simple way to directly extend the linear model to accommodate non-linear relationships, using *polynomial regression*. In later chapters, we will present more complex approaches for performing non-linear fits in more general settings.

Consider Figure 3.8, in which the `mpg` (gas mileage in miles per gallon) versus `horsepower` is shown for a number of cars in the `Auto` data set. The

polynomial regression

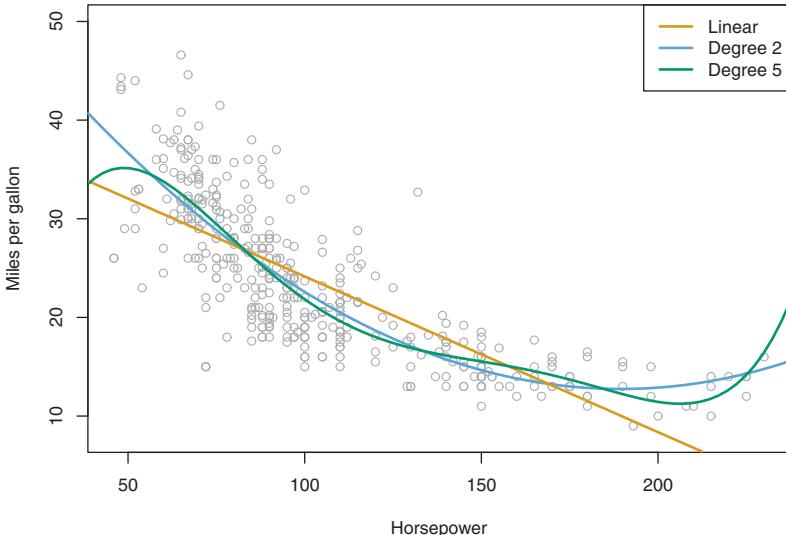


FIGURE 3.8. The `Auto` data set. For a number of cars, `mpg` and `horsepower` are shown. The linear regression fit is shown in orange. The linear regression fit for a model that includes `horsepower`² is shown as a blue curve. The linear regression fit for a model that includes all polynomials of `horsepower` up to fifth-degree is shown in green.

orange line represents the linear regression fit. There is a pronounced relationship between `mpg` and `horsepower`, but it seems clear that this relationship is in fact non-linear: the data suggest a curved relationship. A simple approach for incorporating non-linear associations in a linear model is to include transformed versions of the predictors in the model. For example, the points in Figure 3.8 seem to have a *quadratic* shape, suggesting that a quadratic model of the form

$$\text{mpg} = \beta_0 + \beta_1 \times \text{horsepower} + \beta_2 \times \text{horsepower}^2 + \epsilon \quad (3.36)$$

may provide a better fit. Equation 3.36 involves predicting `mpg` using a non-linear function of `horsepower`. *But it is still a linear model!* That is, (3.36) is simply a multiple linear regression model with $X_1 = \text{horsepower}$ and $X_2 = \text{horsepower}^2$. So we can use standard linear regression software to estimate β_0 , β_1 , and β_2 in order to produce a non-linear fit. The blue curve in Figure 3.8 shows the resulting quadratic fit to the data. The quadratic fit appears to be substantially better than the fit obtained when just the linear term is included. The R^2 of the quadratic fit is 0.688, compared to 0.606 for the linear fit, and the p-value in Table 3.10 for the quadratic term is highly significant.

If including `horsepower`² led to such a big improvement in the model, why not include `horsepower`³, `horsepower`⁴, or even `horsepower`⁵? The green curve

	Coefficient	Std. error	t-statistic	p-value
Intercept	56.9001	1.8004	31.6	< 0.0001
horsepower	-0.4662	0.0311	-15.0	< 0.0001
horsepower ²	0.0012	0.0001	10.1	< 0.0001

TABLE 3.10. For the `Auto` data set, least squares coefficient estimates associated with the regression of `mpg` onto `horsepower` and `horsepower2`.

in Figure 3.8 displays the fit that results from including all polynomials up to fifth degree in the model (3.36). The resulting fit seems unnecessarily wiggly—that is, it is unclear that including the additional terms really has led to a better fit to the data.

The approach that we have just described for extending the linear model to accommodate non-linear relationships is known as *polynomial regression*, since we have included polynomial functions of the predictors in the regression model. We further explore this approach and other non-linear extensions of the linear model in Chapter 7.

3.3.3 Potential Problems

When we fit a linear regression model to a particular data set, many problems may occur. Most common among these are the following:

1. Non-linearity of the response-predictor relationships.
2. Correlation of error terms.
3. Non-constant variance of error terms.
4. Outliers.
5. High-leverage points.
6. Collinearity.

In practice, identifying and overcoming these problems is as much an art as a science. Many pages in countless books have been written on this topic. Since the linear regression model is not our primary focus here, we will provide only a brief summary of some key points.

1. Non-linearity of the Data

The linear regression model assumes that there is a straight-line relationship between the predictors and the response. If the true relationship is far from linear, then virtually all of the conclusions that we draw from the fit are suspect. In addition, the prediction accuracy of the model can be significantly reduced.

Residual plots are a useful graphical tool for identifying non-linearity. Given a simple linear regression model, we can plot the residuals, $e_i = y_i - \hat{y}_i$, versus the predictor x_i . In the case of a multiple regression model,

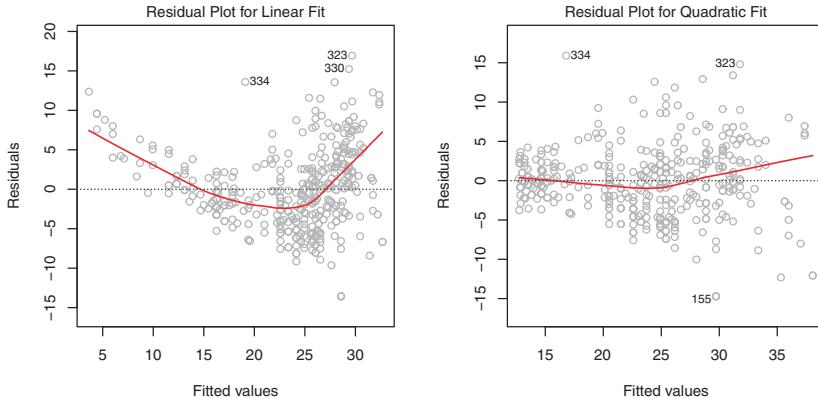


FIGURE 3.9. Plots of residuals versus predicted (or fitted) values for the `Auto` data set. In each plot, the red line is a smooth fit to the residuals, intended to make it easier to identify a trend. Left: A linear regression of `mpg` on `horsepower`. A strong pattern in the residuals indicates non-linearity in the data. Right: A linear regression of `mpg` on `horsepower` and `horsepower`². There is little pattern in the residuals.

since there are multiple predictors, we instead plot the residuals versus the predicted (or *fitted*) values \hat{y}_i . Ideally, the residual plot will show no discernible pattern. The presence of a pattern may indicate a problem with some aspect of the linear model.

The left panel of Figure 3.9 displays a residual plot from the linear regression of `mpg` onto `horsepower` on the `Auto` data set that was illustrated in Figure 3.8. The red line is a smooth fit to the residuals, which is displayed in order to make it easier to identify any trends. The residuals exhibit a clear U-shape, which provides a strong indication of non-linearity in the data. In contrast, the right-hand panel of Figure 3.9 displays the residual plot that results from the model (3.36), which contains a quadratic term. There appears to be little pattern in the residuals, suggesting that the quadratic term improves the fit to the data.

If the residual plot indicates that there are non-linear associations in the data, then a simple approach is to use non-linear transformations of the predictors, such as $\log X$, \sqrt{X} , and X^2 , in the regression model. In the later chapters of this book, we will discuss other more advanced non-linear approaches for addressing this issue.

2. Correlation of Error Terms

An important assumption of the linear regression model is that the error terms, $\epsilon_1, \epsilon_2, \dots, \epsilon_n$, are uncorrelated. What does this mean? For instance, if the errors are uncorrelated, then the fact that ϵ_i is positive provides little or no information about the sign of ϵ_{i+1} . The standard errors that are computed for the estimated regression coefficients or the fitted values

are based on the assumption of uncorrelated error terms. If in fact there is correlation among the error terms, then the estimated standard errors will tend to underestimate the true standard errors. As a result, confidence and prediction intervals will be narrower than they should be. For example, a 95 % confidence interval may in reality have a much lower probability than 0.95 of containing the true value of the parameter. In addition, p-values associated with the model will be lower than they should be; this could cause us to erroneously conclude that a parameter is statistically significant. In short, if the error terms are correlated, we may have an unwarranted sense of confidence in our model.

As an extreme example, suppose we accidentally doubled our data, leading to observations and error terms identical in pairs. If we ignored this, our standard error calculations would be as if we had a sample of size $2n$, when in fact we have only n samples. Our estimated parameters would be the same for the $2n$ samples as for the n samples, but the confidence intervals would be narrower by a factor of $\sqrt{2}$!

Why might correlations among the error terms occur? Such correlations frequently occur in the context of *time series* data, which consists of observations for which measurements are obtained at discrete points in time. In many cases, observations that are obtained at adjacent time points will have positively correlated errors. In order to determine if this is the case for a given data set, we can plot the residuals from our model as a function of time. If the errors are uncorrelated, then there should be no discernible pattern. On the other hand, if the error terms are positively correlated, then we may see *tracking* in the residuals—that is, adjacent residuals may have similar values. Figure 3.10 provides an illustration. In the top panel, we see the residuals from a linear regression fit to data generated with uncorrelated errors. There is no evidence of a time-related trend in the residuals. In contrast, the residuals in the bottom panel are from a data set in which adjacent errors had a correlation of 0.9. Now there is a clear pattern in the residuals—adjacent residuals tend to take on similar values. Finally, the center panel illustrates a more moderate case in which the residuals had a correlation of 0.5. There is still evidence of tracking, but the pattern is less clear.

time series

tracking

Many methods have been developed to properly take account of correlations in the error terms in time series data. Correlation among the error terms can also occur outside of time series data. For instance, consider a study in which individuals' heights are predicted from their weights. The assumption of uncorrelated errors could be violated if some of the individuals in the study are members of the same family, or eat the same diet, or have been exposed to the same environmental factors. In general, the assumption of uncorrelated errors is extremely important for linear regression as well as for other statistical methods, and good experimental design is crucial in order to mitigate the risk of such correlations.

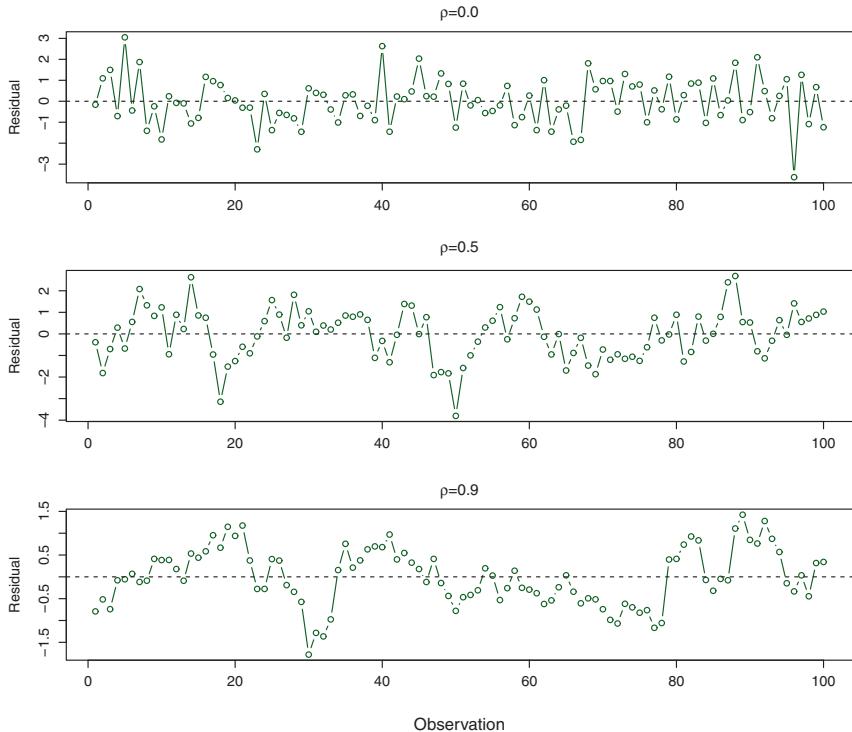


FIGURE 3.10. Plots of residuals from simulated time series data sets generated with differing levels of correlation ρ between error terms for adjacent time points.

3. Non-constant Variance of Error Terms

Another important assumption of the linear regression model is that the error terms have a constant variance, $\text{Var}(\epsilon_i) = \sigma^2$. The standard errors, confidence intervals, and hypothesis tests associated with the linear model rely upon this assumption.

Unfortunately, it is often the case that the variances of the error terms are non-constant. For instance, the variances of the error terms may increase with the value of the response. One can identify non-constant variances in the errors, or *heteroscedasticity*, from the presence of a *funnel shape* in the residual plot. An example is shown in the left-hand panel of Figure 3.11, in which the magnitude of the residuals tends to increase with the fitted values. When faced with this problem, one possible solution is to transform the response Y using a concave function such as $\log Y$ or \sqrt{Y} . Such a transformation results in a greater amount of shrinkage of the larger responses, leading to a reduction in heteroscedasticity. The right-hand panel of Figure 3.11 displays the residual plot after transforming the response

heteroscedasticity

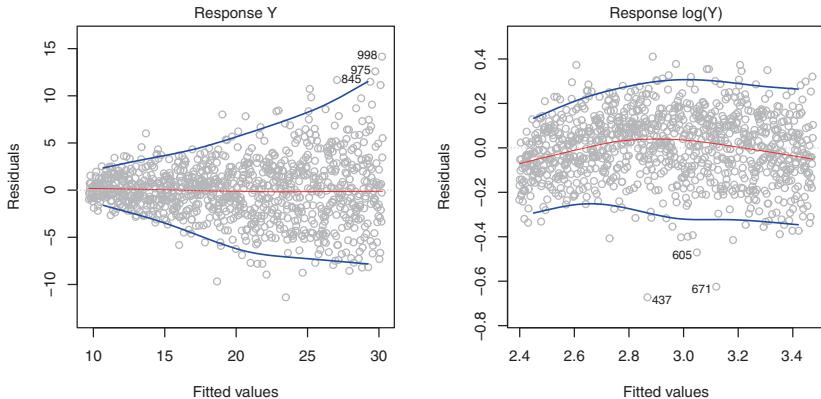


FIGURE 3.11. Residual plots. In each plot, the red line is a smooth fit to the residuals, intended to make it easier to identify a trend. The blue lines track the outer quantiles of the residuals, and emphasize patterns. Left: The funnel shape indicates heteroscedasticity. Right: The response has been log transformed, and there is now no evidence of heteroscedasticity.

using $\log Y$. The residuals now appear to have constant variance, though there is some evidence of a slight non-linear relationship in the data.

Sometimes we have a good idea of the variance of each response. For example, the i th response could be an average of n_i raw observations. If each of these raw observations is uncorrelated with variance σ^2 , then their average has variance $\sigma_i^2 = \sigma^2/n_i$. In this case a simple remedy is to fit our model by *weighted least squares*, with weights proportional to the inverse variances—i.e. $w_i = n_i$ in this case. Most linear regression software allows for observation weights.

weighted least squares

4. Outliers

An *outlier* is a point for which y_i is far from the value predicted by the model. Outliers can arise for a variety of reasons, such as incorrect recording of an observation during data collection.

The red point (observation 20) in the left-hand panel of Figure 3.11 illustrates a typical outlier. The red solid line is the least squares regression fit, while the blue dashed line is the least squares fit after removal of the outlier. In this case, removing the outlier has little effect on the least squares line: it leads to almost no change in the slope, and a minuscule reduction in the intercept. It is typical for an outlier that does not have an unusual predictor value to have little effect on the least squares fit. However, even if an outlier does not have much effect on the least squares fit, it can cause other problems. For instance, in this example, the RSE is 1.09 when the outlier is included in the regression, but it is only 0.77 when the outlier is removed. Since the RSE is used to compute all confidence intervals and

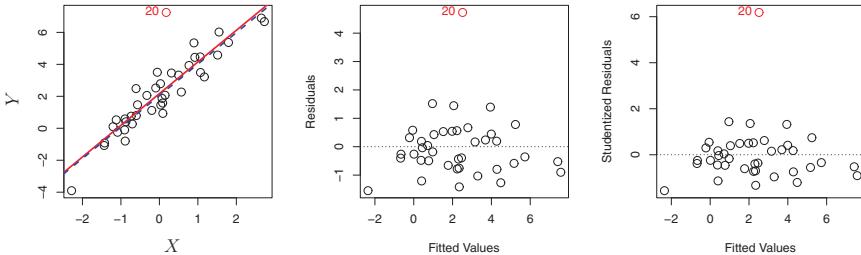


FIGURE 3.12. Left: The least squares regression line is shown in red, and the regression line after removing the outlier is shown in blue. Center: The residual plot clearly identifies the outlier. Right: The outlier has a studentized residual of 6; typically we expect values between -3 and 3 .

p-values, such a dramatic increase caused by a single data point can have implications for the interpretation of the fit. Similarly, inclusion of the outlier causes the R^2 to decline from 0.892 to 0.805.

Residual plots can be used to identify outliers. In this example, the outlier is clearly visible in the residual plot illustrated in the center panel of Figure 3.12. But in practice, it can be difficult to decide how large a residual needs to be before we consider the point to be an outlier. To address this problem, instead of plotting the residuals, we can plot the *studentized residuals*, computed by dividing each residual e_i by its estimated standard error. Observations whose studentized residuals are greater than 3 in absolute value are possible outliers. In the right-hand panel of Figure 3.12, the outlier's studentized residual exceeds 6, while all other observations have studentized residuals between -2 and 2 .

studentized residual

If we believe that an outlier has occurred due to an error in data collection or recording, then one solution is to simply remove the observation. However, care should be taken, since an outlier may instead indicate a deficiency with the model, such as a missing predictor.

5. High Leverage Points

We just saw that outliers are observations for which the response y_i is unusual given the predictor x_i . In contrast, observations with *high leverage* have an unusual value for x_i . For example, observation 41 in the left-hand panel of Figure 3.13 has high leverage, in that the predictor value for this observation is large relative to the other observations. (Note that the data displayed in Figure 3.13 are the same as the data displayed in Figure 3.12, but with the addition of a single high leverage observation.) The red solid line is the least squares fit to the data, while the blue dashed line is the fit produced when observation 41 is removed. Comparing the left-hand panels of Figures 3.12 and 3.13, we observe that removing the high leverage observation has a much more substantial impact on the least squares line.

high leverage

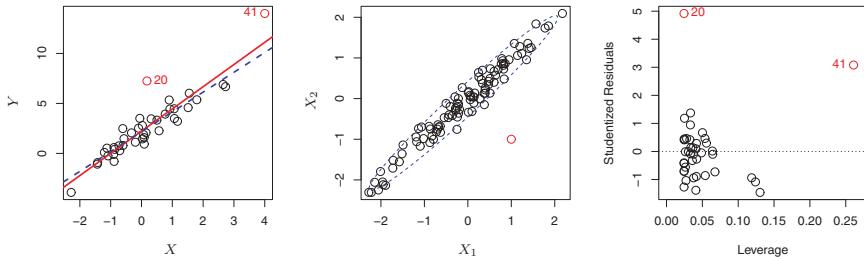


FIGURE 3.13. Left: *Observation 41 is a high leverage point, while 20 is not.* The red line is the fit to all the data, and the blue line is the fit with observation 41 removed. Center: *The red observation is not unusual in terms of its X_1 value or its X_2 value, but still falls outside the bulk of the data, and hence has high leverage.* Right: *Observation 41 has a high leverage and a high residual.*

than removing the outlier. In fact, high leverage observations tend to have a sizable impact on the estimated regression line. It is cause for concern if the least squares line is heavily affected by just a couple of observations, because any problems with these points may invalidate the entire fit. For this reason, it is important to identify high leverage observations.

In a simple linear regression, high leverage observations are fairly easy to identify, since we can simply look for observations for which the predictor value is outside of the normal range of the observations. But in a multiple linear regression with many predictors, it is possible to have an observation that is well within the range of each individual predictor's values, but that is unusual in terms of the full set of predictors. An example is shown in the center panel of Figure 3.13, for a data set with two predictors, X_1 and X_2 . Most of the observations' predictor values fall within the blue dashed ellipse, but the red observation is well outside of this range. But neither its value for X_1 nor its value for X_2 is unusual. So if we examine just X_1 or just X_2 , we will fail to notice this high leverage point. This problem is more pronounced in multiple regression settings with more than two predictors, because then there is no simple way to plot all dimensions of the data simultaneously.

In order to quantify an observation's leverage, we compute the *leverage statistic*. A large value of this statistic indicates an observation with high leverage. For a simple linear regression,

$$h_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{i'=1}^n (x_{i'} - \bar{x})^2}. \quad (3.37)$$

leverage
statistic

It is clear from this equation that h_i increases with the distance of x_i from \bar{x} . There is a simple extension of h_i to the case of multiple predictors, though we do not provide the formula here. The leverage statistic h_i is always between $1/n$ and 1, and the average leverage for all the observations is always equal to $(p+1)/n$. So if a given observation has a leverage statistic

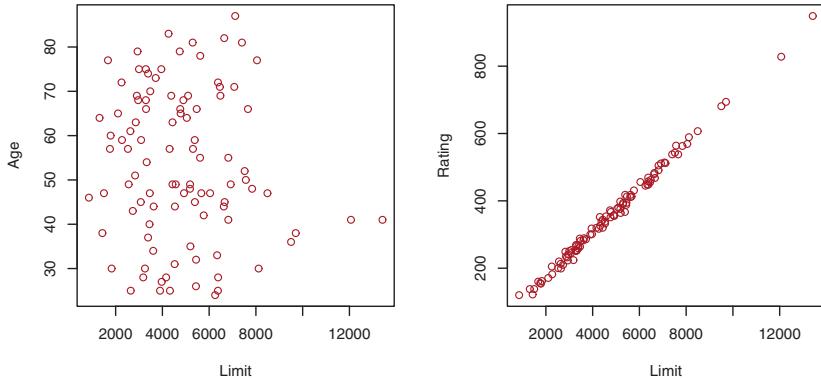


FIGURE 3.14. Scatterplots of the observations from the `Credit` data set. Left: A plot of `age` versus `limit`. These two variables are not collinear. Right: A plot of `rating` versus `limit`. There is high collinearity.

that greatly exceeds $(p+1)/n$, then we may suspect that the corresponding point has high leverage.

The right-hand panel of Figure 3.13 provides a plot of the studentized residuals versus h_i for the data in the left-hand panel of Figure 3.13. Observation 41 stands out as having a very high leverage statistic as well as a high studentized residual. In other words, it is an outlier as well as a high leverage observation. This is a particularly dangerous combination! This plot also reveals the reason that observation 20 had relatively little effect on the least squares fit in Figure 3.12: it has low leverage.

6. Collinearity

Collinearity refers to the situation in which two or more predictor variables are closely related to one another. The concept of collinearity is illustrated in Figure 3.14 using the `Credit` data set. In the left-hand panel of Figure 3.14, the two predictors `limit` and `age` appear to have no obvious relationship. In contrast, in the right-hand panel of Figure 3.14, the predictors `limit` and `rating` are very highly correlated with each other, and we say that they are *collinear*. The presence of collinearity can pose problems in the regression context, since it can be difficult to separate out the individual effects of collinear variables on the response. In other words, since `limit` and `rating` tend to increase or decrease together, it can be difficult to determine how each one separately is associated with the response, `balance`.

Figure 3.15 illustrates some of the difficulties that can result from collinearity. The left-hand panel of Figure 3.15 is a contour plot of the RSS (3.22) associated with different possible coefficient estimates for the regression of `balance` on `limit` and `age`. Each ellipse represents a set of coefficients that correspond to the same RSS, with ellipses nearest to the center taking on the lowest values of RSS. The black dots and associated dashed

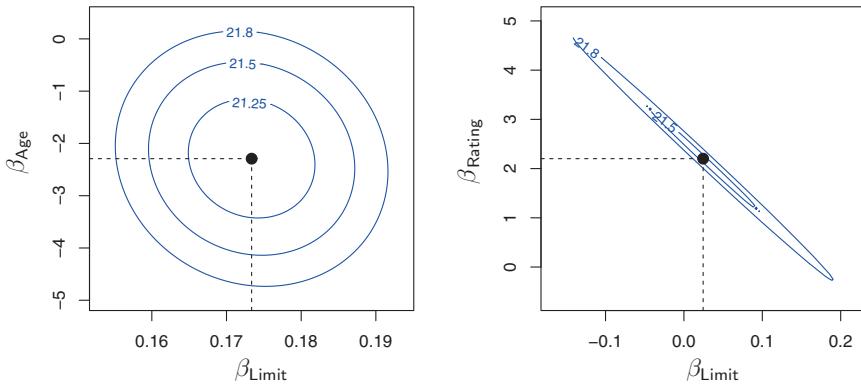


FIGURE 3.15. Contour plots for the RSS values as a function of the parameters β for various regressions involving the `Credit` data set. In each plot, the black dots represent the coefficient values corresponding to the minimum RSS. Left: A contour plot of RSS for the regression of `balance` onto `age` and `limit`. The minimum value is well defined. Right: A contour plot of RSS for the regression of `balance` onto `rating` and `limit`. Because of the collinearity, there are many pairs $(\beta_{\text{Limit}}, \beta_{\text{Rating}})$ with a similar value for RSS.

lines represent the coefficient estimates that result in the smallest possible RSS—in other words, these are the least squares estimates. The axes for `limit` and `age` have been scaled so that the plot includes possible coefficient estimates that are up to four standard errors on either side of the least squares estimates. Thus the plot includes all plausible values for the coefficients. For example, we see that the true `limit` coefficient is almost certainly somewhere between 0.15 and 0.20.

In contrast, the right-hand panel of Figure 3.15 displays contour plots of the RSS associated with possible coefficient estimates for the regression of `balance` onto `limit` and `rating`, which we know to be highly collinear. Now the contours run along a narrow valley; there is a broad range of values for the coefficient estimates that result in equal values for RSS. Hence a small change in the data could cause the pair of coefficient values that yield the smallest RSS—that is, the least squares estimates—to move anywhere along this valley. This results in a great deal of uncertainty in the coefficient estimates. Notice that the scale for the `limit` coefficient now runs from roughly -0.2 to 0.2 ; this is an eight-fold increase over the plausible range of the `limit` coefficient in the regression with `age`. Interestingly, even though the `limit` and `rating` coefficients now have much more individual uncertainty, they will almost certainly lie somewhere in this contour valley. For example, we would not expect the true value of the `limit` and `rating` coefficients to be -0.1 and 1 respectively, even though such a value is plausible for each coefficient individually.

		Coefficient	Std. error	t-statistic	p-value
Model 1	Intercept	-173.411	43.828	-3.957	< 0.0001
	age	-2.292	0.672	-3.407	0.0007
	limit	0.173	0.005	34.496	< 0.0001
Model 2	Intercept	-377.537	45.254	-8.343	< 0.0001
	rating	2.202	0.952	2.312	0.0213
	limit	0.025	0.064	0.384	0.7012

TABLE 3.11. The results for two multiple regression models involving the Credit data set are shown. Model 1 is a regression of `balance` on `age` and `limit`, and Model 2 a regression of `balance` on `rating` and `limit`. The standard error of $\hat{\beta}_{\text{limit}}$ increases 12-fold in the second regression, due to collinearity.

Since collinearity reduces the accuracy of the estimates of the regression coefficients, it causes the standard error for $\hat{\beta}_j$ to grow. Recall that the t -statistic for each predictor is calculated by dividing $\hat{\beta}_j$ by its standard error. Consequently, collinearity results in a decline in the t -statistic. As a result, in the presence of collinearity, we may fail to reject $H_0 : \beta_j = 0$. This means that the *power* of the hypothesis test—the probability of correctly detecting a *non-zero* coefficient—is reduced by collinearity.

Table 3.11 compares the coefficient estimates obtained from two separate multiple regression models. The first is a regression of `balance` on `age` and `limit`, and the second is a regression of `balance` on `rating` and `limit`. In the first regression, both `age` and `limit` are highly significant with very small p-values. In the second, the collinearity between `limit` and `rating` has caused the standard error for the `limit` coefficient estimate to increase by a factor of 12 and the p-value to increase to 0.701. In other words, the importance of the `limit` variable has been masked due to the presence of collinearity. To avoid such a situation, it is desirable to identify and address potential collinearity problems while fitting the model.

A simple way to detect collinearity is to look at the correlation matrix of the predictors. An element of this matrix that is large in absolute value indicates a pair of highly correlated variables, and therefore a collinearity problem in the data. Unfortunately, not all collinearity problems can be detected by inspection of the correlation matrix: it is possible for collinearity to exist between three or more variables even if no pair of variables has a particularly high correlation. We call this situation *multicollinearity*. Instead of inspecting the correlation matrix, a better way to assess multicollinearity is to compute the *variance inflation factor* (VIF). The VIF is the ratio of the variance of $\hat{\beta}_j$ when fitting the full model divided by the variance of $\hat{\beta}_j$ if fit on its own. The smallest possible value for VIF is 1, which indicates the complete absence of collinearity. Typically in practice there is a small amount of collinearity among the predictors. As a rule of thumb, a VIF value that exceeds 5 or 10 indicates a problematic amount of

power

multicollinearity

variance inflation factor

collinearity. The VIF for each variable can be computed using the formula

$$\text{VIF}(\hat{\beta}_j) = \frac{1}{1 - R_{X_j|X_{-j}}^2},$$

where $R_{X_j|X_{-j}}^2$ is the R^2 from a regression of X_j onto all of the other predictors. If $R_{X_j|X_{-j}}^2$ is close to one, then collinearity is present, and so the VIF will be large.

In the `Credit` data, a regression of `balance` on `age`, `rating`, and `limit` indicates that the predictors have VIF values of 1.01, 160.67, and 160.59. As we suspected, there is considerable collinearity in the data!

When faced with the problem of collinearity, there are two simple solutions. The first is to drop one of the problematic variables from the regression. This can usually be done without much compromise to the regression fit, since the presence of collinearity implies that the information that this variable provides about the response is redundant in the presence of the other variables. For instance, if we regress `balance` onto `age` and `limit`, without the `rating` predictor, then the resulting VIF values are close to the minimum possible value of 1, and the R^2 drops from 0.754 to 0.75. So dropping `rating` from the set of predictors has effectively solved the collinearity problem without compromising the fit. The second solution is to combine the collinear variables together into a single predictor. For instance, we might take the average of standardized versions of `limit` and `rating` in order to create a new variable that measures *credit worthiness*.

3.4 The Marketing Plan

We now briefly return to the seven questions about the `Advertising` data that we set out to answer at the beginning of this chapter.

1. Is there a relationship between advertising sales and budget?

This question can be answered by fitting a multiple regression model of `sales` onto `TV`, `radio`, and `newspaper`, as in (3.20), and testing the hypothesis $H_0 : \beta_{\text{TV}} = \beta_{\text{radio}} = \beta_{\text{newspaper}} = 0$. In Section 3.2.2, we showed that the F-statistic can be used to determine whether or not we should reject this null hypothesis. In this case the p-value corresponding to the F-statistic in Table 3.6 is very low, indicating clear evidence of a relationship between advertising and sales.

2. How strong is the relationship?

We discussed two measures of model accuracy in Section 3.1.3. First, the RSE estimates the standard deviation of the response from the population regression line. For the `Advertising` data, the RSE is 1,681

units while the mean value for the response is 14,022, indicating a percentage error of roughly 12%. Second, the R^2 statistic records the percentage of variability in the response that is explained by the predictors. The predictors explain almost 90% of the variance in **sales**. The RSE and R^2 statistics are displayed in Table 3.6.

3. Which media contribute to sales?

To answer this question, we can examine the p-values associated with each predictor's t-statistic (Section 3.1.2). In the multiple linear regression displayed in Table 3.4, the p-values for **TV** and **radio** are low, but the p-value for **newspaper** is not. This suggests that only **TV** and **radio** are related to **sales**. In Chapter 6 we explore this question in greater detail.

4. How large is the effect of each medium on sales?

We saw in Section 3.1.2 that the standard error of $\hat{\beta}_j$ can be used to construct confidence intervals for β_j . For the **Advertising** data, the 95% confidence intervals are as follows: (0.043, 0.049) for **TV**, (0.172, 0.206) for **radio**, and (-0.013, 0.011) for **newspaper**. The confidence intervals for **TV** and **radio** are narrow and far from zero, providing evidence that these media are related to **sales**. But the interval for **newspaper** includes zero, indicating that the variable is not statistically significant given the values of **TV** and **radio**.

We saw in Section 3.3.3 that collinearity can result in very wide standard errors. Could collinearity be the reason that the confidence interval associated with **newspaper** is so wide? The VIF scores are 1.005, 1.145, and 1.145 for **TV**, **radio**, and **newspaper**, suggesting no evidence of collinearity.

In order to assess the association of each medium individually on sales, we can perform three separate simple linear regressions. Results are shown in Tables 3.1 and 3.3. There is evidence of an extremely strong association between **TV** and **sales** and between **radio** and **sales**. There is evidence of a mild association between **newspaper** and **sales**, when the values of **TV** and **radio** are ignored.

5. How accurately can we predict future sales?

The response can be predicted using (3.21). The accuracy associated with this estimate depends on whether we wish to predict an individual response, $Y = f(X) + \epsilon$, or the average response, $f(X)$ (Section 3.2.2). If the former, we use a prediction interval, and if the latter, we use a confidence interval. Prediction intervals will always be wider than confidence intervals because they account for the uncertainty associated with ϵ , the irreducible error.

6. Is the relationship linear?

In Section 3.3.3, we saw that residual plots can be used in order to identify non-linearity. If the relationships are linear, then the residual plots should display no pattern. In the case of the **Advertising** data, we observe a non-linear effect in Figure 3.5, though this effect could also be observed in a residual plot. In Section 3.3.2, we discussed the inclusion of transformations of the predictors in the linear regression model in order to accommodate non-linear relationships.

7. Is there synergy among the advertising media?

The standard linear regression model assumes an additive relationship between the predictors and the response. An additive model is easy to interpret because the effect of each predictor on the response is unrelated to the values of the other predictors. However, the additive assumption may be unrealistic for certain data sets. In Section 3.3.2, we showed how to include an interaction term in the regression model in order to accommodate non-additive relationships. A small p-value associated with the interaction term indicates the presence of such relationships. Figure 3.5 suggested that the **Advertising** data may not be additive. Including an interaction term in the model results in a substantial increase in R^2 , from around 90% to almost 97%.

3.5 Comparison of Linear Regression with K -Nearest Neighbors

As discussed in Chapter 2, linear regression is an example of a *parametric* approach because it assumes a linear functional form for $f(X)$. Parametric methods have several advantages. They are often easy to fit, because one need estimate only a small number of coefficients. In the case of linear regression, the coefficients have simple interpretations, and tests of statistical significance can be easily performed. But parametric methods do have a disadvantage: by construction, they make strong assumptions about the form of $f(X)$. If the specified functional form is far from the truth, and prediction accuracy is our goal, then the parametric method will perform poorly. For instance, if we assume a linear relationship between X and Y but the true relationship is far from linear, then the resulting model will provide a poor fit to the data, and any conclusions drawn from it will be suspect.

In contrast, *non-parametric* methods do not explicitly assume a parametric form for $f(X)$, and thereby provide an alternative and more flexible approach for performing regression. We discuss various non-parametric methods in this book. Here we consider one of the simplest and best-known non-parametric methods, *K -nearest neighbors regression* (KNN regression).

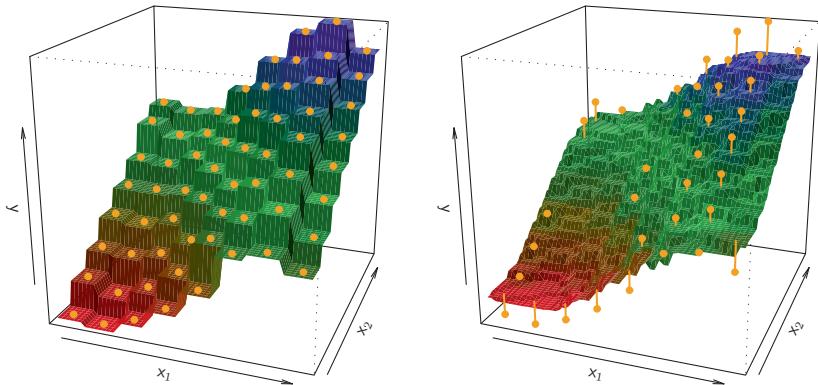


FIGURE 3.16. Plots of $\hat{f}(X)$ using KNN regression on a two-dimensional data set with 64 observations (orange dots). Left: $K = 1$ results in a rough step function fit. Right: $K = 9$ produces a much smoother fit.

The KNN regression method is closely related to the KNN classifier discussed in Chapter 2. Given a value for K and a prediction point x_0 , KNN regression first identifies the K training observations that are closest to x_0 , represented by \mathcal{N}_0 . It then estimates $f(x_0)$ using the average of all the training responses in \mathcal{N}_0 . In other words,

$$\hat{f}(x_0) = \frac{1}{K} \sum_{x_i \in \mathcal{N}_0} y_i.$$

Figure 3.16 illustrates two KNN fits on a data set with $p = 2$ predictors. The fit with $K = 1$ is shown in the left-hand panel, while the right-hand panel corresponds to $K = 9$. We see that when $K = 1$, the KNN fit perfectly interpolates the training observations, and consequently takes the form of a step function. When $K = 9$, the KNN fit still is a step function, but averaging over nine observations results in much smaller regions of constant prediction, and consequently a smoother fit. In general, the optimal value for K will depend on the *bias-variance tradeoff*, which we introduced in Chapter 2. A small value for K provides the most flexible fit, which will have low bias but high variance. This variance is due to the fact that the prediction in a given region is entirely dependent on just one observation. In contrast, larger values of K provide a smoother and less variable fit; the prediction in a region is an average of several points, and so changing one observation has a smaller effect. However, the smoothing may cause bias by masking some of the structure in $f(X)$. In Chapter 5, we introduce several approaches for estimating test error rates. These methods can be used to identify the optimal value of K in KNN regression.

In what setting will a parametric approach such as least squares linear regression outperform a non-parametric approach such as KNN regression? The answer is simple: *the parametric approach will outperform the non-parametric approach if the parametric form that has been selected is close to the true form of f* . Figure 3.17 provides an example with data generated from a one-dimensional linear regression model. The black solid lines represent $f(X)$, while the blue curves correspond to the KNN fits using $K = 1$ and $K = 9$. In this case, the $K = 1$ predictions are far too variable, while the smoother $K = 9$ fit is much closer to $f(X)$. However, since the true relationship is linear, it is hard for a non-parametric approach to compete with linear regression: a non-parametric approach incurs a cost in variance that is not offset by a reduction in bias. The blue dashed line in the left-hand panel of Figure 3.18 represents the linear regression fit to the same data. It is almost perfect. The right-hand panel of Figure 3.18 reveals that linear regression outperforms KNN for this data. The green solid line, plotted as a function of $1/K$, represents the test set mean squared error (MSE) for KNN. The KNN errors are well above the black dashed line, which is the test MSE for linear regression. When the value of K is large, then KNN performs only a little worse than least squares regression in terms of MSE. It performs far worse when K is small.

In practice, the true relationship between X and Y is rarely exactly linear. Figure 3.19 examines the relative performances of least squares regression and KNN under increasing levels of non-linearity in the relationship between X and Y . In the top row, the true relationship is nearly linear. In this case we see that the test MSE for linear regression is still superior to that of KNN for low values of K . However, for $K \geq 4$, KNN outperforms linear regression. The second row illustrates a more substantial deviation from linearity. In this situation, KNN substantially outperforms linear regression for all values of K . Note that as the extent of non-linearity increases, there is little change in the test set MSE for the non-parametric KNN method, but there is a large increase in the test set MSE of linear regression.

Figures 3.18 and 3.19 display situations in which KNN performs slightly worse than linear regression when the relationship is linear, but much better than linear regression for non-linear situations. In a real life situation in which the true relationship is unknown, one might draw the conclusion that KNN should be favored over linear regression because it will at worst be slightly inferior than linear regression if the true relationship is linear, and may give substantially better results if the true relationship is non-linear. But in reality, even when the true relationship is highly non-linear, KNN may still provide inferior results to linear regression. In particular, both Figures 3.18 and 3.19 illustrate settings with $p = 1$ predictor. But in higher dimensions, KNN often performs worse than linear regression.

Figure 3.20 considers the same strongly non-linear situation as in the second row of Figure 3.19, except that we have added additional *noise*

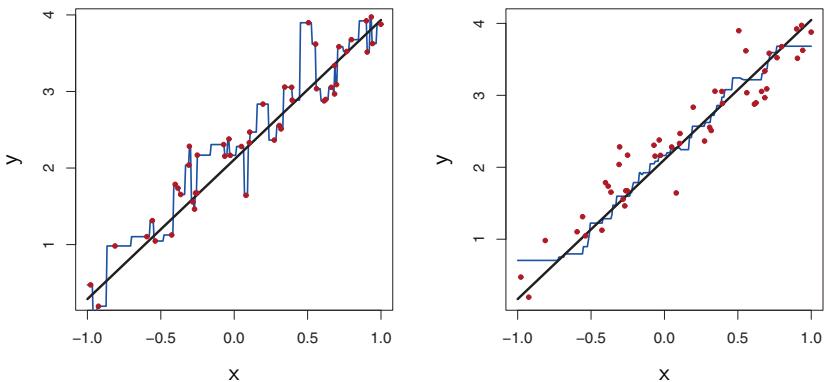


FIGURE 3.17. Plots of $\hat{f}(X)$ using KNN regression on a one-dimensional data set with 100 observations. The true relationship is given by the black solid line. Left: The blue curve corresponds to $K = 1$ and interpolates (i.e. passes directly through) the training data. Right: The blue curve corresponds to $K = 9$, and represents a smoother fit.

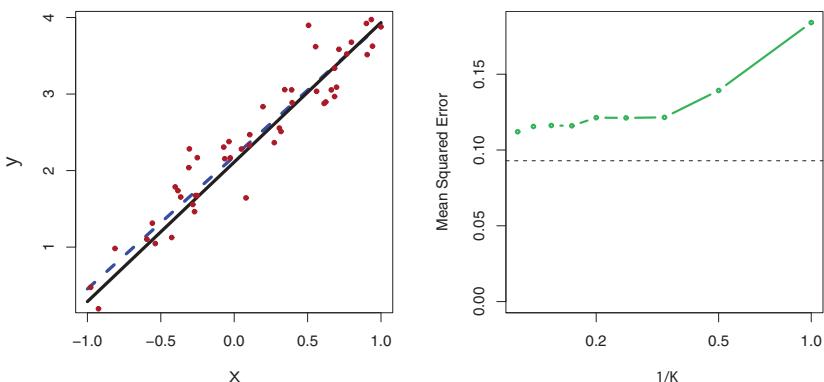


FIGURE 3.18. The same data set shown in Figure 3.17 is investigated further. Left: The blue dashed line is the least squares fit to the data. Since $f(X)$ is in fact linear (displayed as the black line), the least squares regression line provides a very good estimate of $f(X)$. Right: The dashed horizontal line represents the least squares test set MSE, while the green solid line corresponds to the MSE for KNN as a function of $1/K$ (on the log scale). Linear regression achieves a lower test MSE than does KNN regression, since $f(X)$ is in fact linear. For KNN regression, the best results occur with a very large value of K , corresponding to a small value of $1/K$.

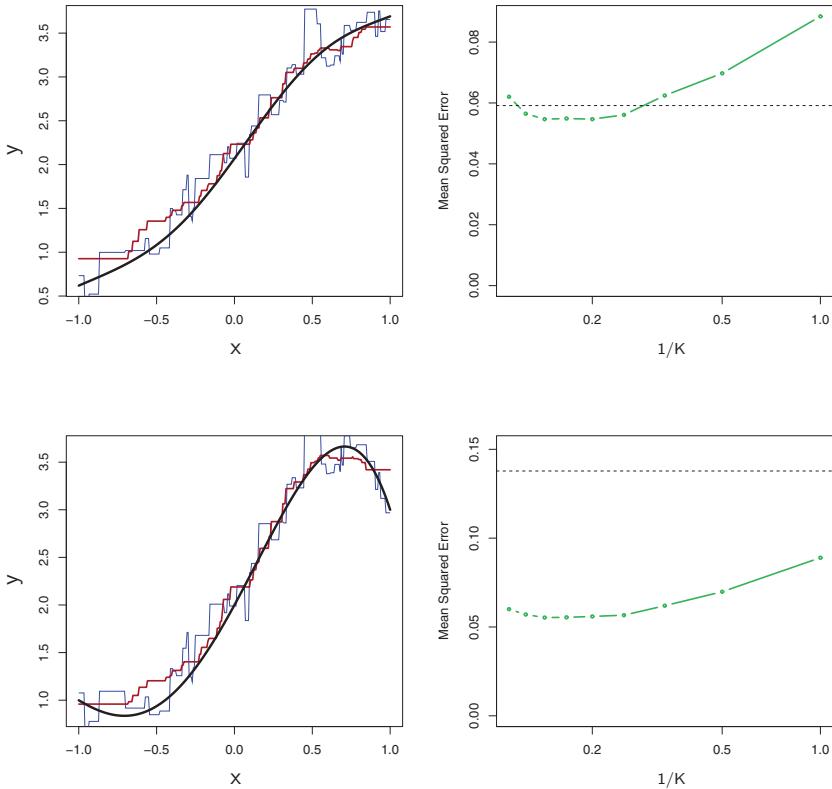


FIGURE 3.19. Top Left: In a setting with a slightly non-linear relationship between X and Y (solid black line), the KNN fits with $K = 1$ (blue) and $K = 9$ (red) are displayed. Top Right: For the slightly non-linear data, the test set MSE for least squares regression (horizontal black) and KNN with various values of $1/K$ (green) are displayed. Bottom Left and Bottom Right: As in the top panel, but with a strongly non-linear relationship between X and Y .

predictors that are not associated with the response. When $p = 1$ or $p = 2$, KNN outperforms linear regression. But for $p = 3$ the results are mixed, and for $p \geq 4$ linear regression is superior to KNN. In fact, the increase in dimension has only caused a small deterioration in the linear regression test set MSE, but it has caused more than a ten-fold increase in the MSE for KNN. This decrease in performance as the dimension increases is a common problem for KNN, and results from the fact that in higher dimensions there is effectively a reduction in sample size. In this data set there are 100 training observations; when $p = 1$, this provides enough information to accurately estimate $f(X)$. However, spreading 100 observations over $p = 20$ dimensions results in a phenomenon in which a given observation has no *nearby neighbors*—this is the so-called *curse of dimensionality*. That is, the K observations that are nearest to a given test observation x_0 may be very far away from x_0 in p -dimensional space when p is large, leading to a

curse of dimensionality

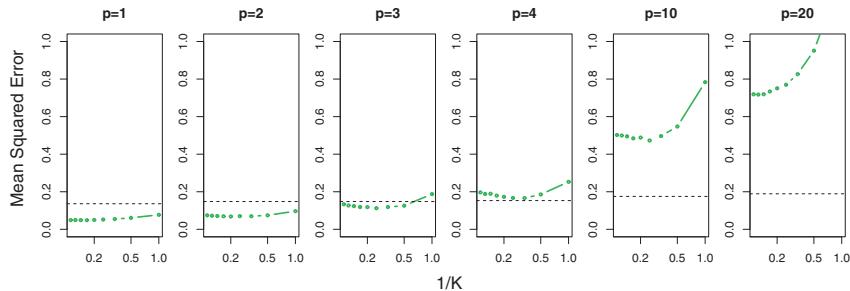


FIGURE 3.20. Test MSE for linear regression (black dashed lines) and KNN (green curves) as the number of variables p increases. The true function is non-linear in the first variable, as in the lower panel in Figure 3.19, and does not depend on the additional variables. The performance of linear regression deteriorates slowly in the presence of these additional noise variables, whereas KNN's performance degrades much more quickly as p increases.

very poor prediction of $f(x_0)$ and hence a poor KNN fit. As a general rule, parametric methods will tend to outperform non-parametric approaches when there is a small number of observations per predictor.

Even in problems in which the dimension is small, we might prefer linear regression to KNN from an interpretability standpoint. If the test MSE of KNN is only slightly lower than that of linear regression, we might be willing to forego a little bit of prediction accuracy for the sake of a simple model that can be described in terms of just a few coefficients, and for which p-values are available.

3.6 Lab: Linear Regression

3.6.1 Libraries

The `library()` function is used to load *libraries*, or groups of functions and data sets that are not included in the base `R` distribution. Basic functions that perform least squares linear regression and other simple analyses come standard with the base distribution, but more exotic functions require additional libraries. Here we load the `MASS` package, which is a very large collection of data sets and functions. We also load the `ISLR` package, which includes the data sets associated with this book.

```
> library(MASS)
> library(ISLR)
```

If you receive an error message when loading any of these libraries, it likely indicates that the corresponding library has not yet been installed on your system. Some libraries, such as `MASS`, come with `R` and do not need to be separately installed on your computer. However, other packages, such as

3.7 Exercises

Conceptual

1. Describe the null hypotheses to which the p-values given in Table 3.4 correspond. Explain what conclusions you can draw based on these p-values. Your explanation should be phrased in terms of **sales**, **TV**, **radio**, and **newspaper**, rather than in terms of the coefficients of the linear model.
2. Carefully explain the differences between the KNN classifier and KNN regression methods.
3. Suppose we have a data set with five predictors, $X_1 = \text{GPA}$, $X_2 = \text{IQ}$, $X_3 = \text{Gender}$ (1 for Female and 0 for Male), $X_4 = \text{Interaction between GPA and IQ}$, and $X_5 = \text{Interaction between GPA and Gender}$. The response is starting salary after graduation (in thousands of dollars). Suppose we use least squares to fit the model, and get $\hat{\beta}_0 = 50$, $\hat{\beta}_1 = 20$, $\hat{\beta}_2 = 0.07$, $\hat{\beta}_3 = 35$, $\hat{\beta}_4 = 0.01$, $\hat{\beta}_5 = -10$.
 - (a) Which answer is correct, and why?
 - i. For a fixed value of IQ and GPA, males earn more on average than females.
 - ii. For a fixed value of IQ and GPA, females earn more on average than males.
 - iii. For a fixed value of IQ and GPA, males earn more on average than females provided that the GPA is high enough.
 - iv. For a fixed value of IQ and GPA, females earn more on average than males provided that the GPA is high enough.
 - (b) Predict the salary of a female with IQ of 110 and a GPA of 4.0.
 - (c) True or false: Since the coefficient for the GPA/IQ interaction term is very small, there is very little evidence of an interaction effect. Justify your answer.
4. I collect a set of data ($n = 100$ observations) containing a single predictor and a quantitative response. I then fit a linear regression model to the data, as well as a separate cubic regression, i.e. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$.
 - (a) Suppose that the true relationship between X and Y is linear, i.e. $Y = \beta_0 + \beta_1 X + \epsilon$. Consider the training residual sum of squares (RSS) for the linear regression, and also the training RSS for the cubic regression. Would we expect one to be lower than the other, would we expect them to be the same, or is there not enough information to tell? Justify your answer.

- (b) Answer (a) using test rather than training RSS.
- (c) Suppose that the true relationship between X and Y is not linear, but we don't know how far it is from linear. Consider the training RSS for the linear regression, and also the training RSS for the cubic regression. Would we expect one to be lower than the other, would we expect them to be the same, or is there not enough information to tell? Justify your answer.
- (d) Answer (c) using test rather than training RSS.
5. Consider the fitted values that result from performing linear regression without an intercept. In this setting, the i th fitted value takes the form

$$\hat{y}_i = x_i \hat{\beta},$$

where

$$\hat{\beta} = \left(\sum_{i=1}^n x_i y_i \right) / \left(\sum_{i'=1}^n x_{i'}^2 \right). \quad (3.38)$$

Show that we can write

$$\hat{y}_i = \sum_{i'=1}^n a_{i'} y_{i'}.$$

What is $a_{i'}$?

Note: We interpret this result by saying that the fitted values from linear regression are linear combinations of the response values.

6. Using (3.4), argue that in the case of simple linear regression, the least squares line always passes through the point (\bar{x}, \bar{y}) .
7. It is claimed in the text that in the case of simple linear regression of Y onto X , the R^2 statistic (3.17) is equal to the square of the correlation between X and Y (3.18). Prove that this is the case. For simplicity, you may assume that $\bar{x} = \bar{y} = 0$. 

Applied

8. This question involves the use of simple linear regression on the **Auto** data set.
- (a) Use the **lm()** function to perform a simple linear regression with **mpg** as the response and **horsepower** as the predictor. Use the **summary()** function to print the results. Comment on the output. For example:

4

Classification

The linear regression model discussed in Chapter 3 assumes that the response variable Y is quantitative. But in many situations, the response variable is instead *qualitative*. For example, eye color is qualitative, taking on values blue, brown, or green. Often qualitative variables are referred to as *categorical*; we will use these terms interchangeably. In this chapter, we study approaches for predicting qualitative responses, a process that is known as *classification*. Predicting a qualitative response for an observation can be referred to as *classifying* that observation, since it involves assigning the observation to a category, or class. On the other hand, often the methods used for classification first predict the probability of each of the categories of a qualitative variable, as the basis for making the classification. In this sense they also behave like regression methods.

There are many possible classification techniques, or *classifiers*, that one might use to predict a qualitative response. We touched on some of these in Sections 2.1.5 and 2.2.3. In this chapter we discuss three of the most widely-used classifiers: *logistic regression*, *linear discriminant analysis*, and *K-nearest neighbors*. We discuss more computer-intensive methods in later chapters, such as generalized additive models (Chapter 7), trees, random forests, and boosting (Chapter 8), and support vector machines (Chapter 9).

qualitative

classification

classifier

logistic
regression

linear
discriminant
analysis

K-nearest
neighbors

4.1 An Overview of Classification

Classification problems occur often, perhaps even more so than regression problems. Some examples include:

1. A person arrives at the emergency room with a set of symptoms that could possibly be attributed to one of three medical conditions. Which of the three conditions does the individual have?
2. An online banking service must be able to determine whether or not a transaction being performed on the site is fraudulent, on the basis of the user's IP address, past transaction history, and so forth.
3. On the basis of DNA sequence data for a number of patients with and without a given disease, a biologist would like to figure out which DNA mutations are deleterious (disease-causing) and which are not.

Just as in the regression setting, in the classification setting we have a set of training observations $(x_1, y_1), \dots, (x_n, y_n)$ that we can use to build a classifier. We want our classifier to perform well not only on the training data, but also on test observations that were not used to train the classifier.

In this chapter, we will illustrate the concept of classification using the simulated **Default** data set. We are interested in predicting whether an individual will default on his or her credit card payment, on the basis of annual income and monthly credit card balance. The data set is displayed in Figure 4.1. We have plotted annual **income** and monthly credit card **balance** for a subset of 10,000 individuals. The left-hand panel of Figure 4.1 displays individuals who defaulted in a given month in orange, and those who did not in blue. (The overall default rate is about 3%, so we have plotted only a fraction of the individuals who did not default.) It appears that individuals who defaulted tended to have higher credit card balances than those who did not. In the right-hand panel of Figure 4.1, two pairs of boxplots are shown. The first shows the distribution of **balance** split by the binary **default** variable; the second is a similar plot for **income**. In this chapter, we learn how to build a model to predict **default** (Y) for any given value of **balance** (X_1) and **income** (X_2). Since Y is not quantitative, the simple linear regression model of Chapter 3 is not appropriate.

It is worth noting that Figure 4.1 displays a very pronounced relationship between the predictor **balance** and the response **default**. In most real applications, the relationship between the predictor and the response will not be nearly so strong. However, for the sake of illustrating the classification procedures discussed in this chapter, we use an example in which the relationship between the predictor and the response is somewhat exaggerated.

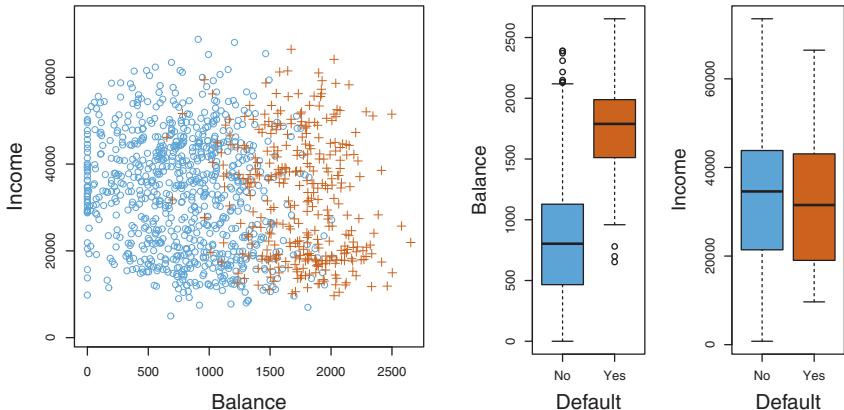


FIGURE 4.1. The `Default` data set. Left: The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue. Center: Boxplots of `balance` as a function of `default` status. Right: Boxplots of `income` as a function of `default` status.

4.2 Why Not Linear Regression?

We have stated that linear regression is not appropriate in the case of a qualitative response. Why not?

Suppose that we are trying to predict the medical condition of a patient in the emergency room on the basis of her symptoms. In this simplified example, there are three possible diagnoses: `stroke`, `drug overdose`, and `epileptic seizure`. We could consider encoding these values as a quantitative response variable, Y , as follows:

$$Y = \begin{cases} 1 & \text{if } \text{stroke}; \\ 2 & \text{if } \text{drug overdose}; \\ 3 & \text{if } \text{epileptic seizure}. \end{cases}$$

Using this coding, least squares could be used to fit a linear regression model to predict Y on the basis of a set of predictors X_1, \dots, X_p . Unfortunately, this coding implies an ordering on the outcomes, putting `drug overdose` in between `stroke` and `epileptic seizure`, and insisting that the difference between `stroke` and `drug overdose` is the same as the difference between `drug overdose` and `epileptic seizure`. In practice there is no particular reason that this needs to be the case. For instance, one could choose an equally reasonable coding,

$$Y = \begin{cases} 1 & \text{if } \text{epileptic seizure}; \\ 2 & \text{if } \text{stroke}; \\ 3 & \text{if } \text{drug overdose}. \end{cases}$$

which would imply a totally different relationship among the three conditions. Each of these codings would produce fundamentally different linear models that would ultimately lead to different sets of predictions on test observations.

If the response variable's values did take on a natural ordering, such as *mild*, *moderate*, and *severe*, and we felt the gap between mild and moderate was similar to the gap between moderate and severe, then a 1, 2, 3 coding would be reasonable. Unfortunately, in general there is no natural way to convert a qualitative response variable with more than two levels into a quantitative response that is ready for linear regression.

For a *binary* (two level) qualitative response, the situation is better. For instance, perhaps there are only two possibilities for the patient's medical condition: **stroke** and **drug overdose**. We could then potentially use the *dummy variable* approach from Section 3.3.1 to code the response as follows:

$$Y = \begin{cases} 0 & \text{if } \text{stroke}; \\ 1 & \text{if } \text{drug overdose}. \end{cases}$$

We could then fit a linear regression to this binary response, and predict **drug overdose** if $\hat{Y} > 0.5$ and **stroke** otherwise. In the binary case it is not hard to show that even if we flip the above coding, linear regression will produce the same final predictions.

For a binary response with a 0/1 coding as above, regression by least squares does make sense; it can be shown that the $X\hat{\beta}$ obtained using linear regression is in fact an estimate of $\Pr(\text{drug overdose}|X)$ in this special case. However, if we use linear regression, some of our estimates might be outside the [0, 1] interval (see Figure 4.2), making them hard to interpret as probabilities! Nevertheless, the predictions provide an ordering and can be interpreted as crude probability estimates. Curiously, it turns out that the classifications that we get if we use linear regression to predict a binary response will be the same as for the linear discriminant analysis (LDA) procedure we discuss in Section 4.4.

However, the dummy variable approach cannot be easily extended to accommodate qualitative responses with more than two levels. For these reasons, it is preferable to use a classification method that is truly suited for qualitative response values, such as the ones presented next.

4.3 Logistic Regression

Consider again the **Default** data set, where the response **default** falls into one of two categories, **Yes** or **No**. Rather than modeling this response Y directly, logistic regression models the *probability* that Y belongs to a particular category.

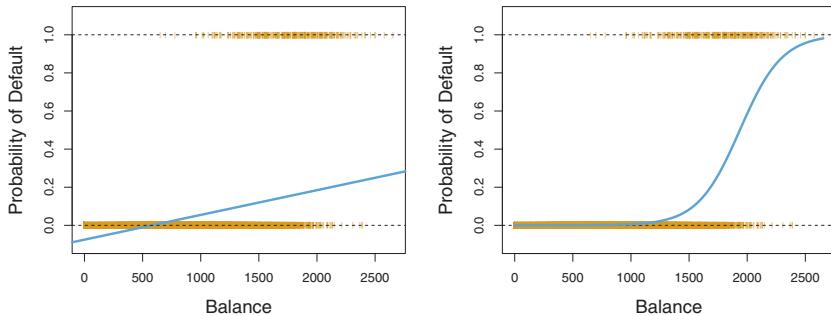


FIGURE 4.2. Classification using the `Default` data. Left: Estimated probability of `default` using linear regression. Some estimated probabilities are negative! The orange ticks indicate the 0/1 values coded for `default` (No or Yes). Right: Predicted probabilities of `default` using logistic regression. All probabilities lie between 0 and 1.

For the `Default` data, logistic regression models the probability of default. For example, the probability of default given `balance` can be written as

$$\Pr(\text{default} = \text{Yes} | \text{balance}).$$

The values of $\Pr(\text{default} = \text{Yes} | \text{balance})$, which we abbreviate $p(\text{balance})$, will range between 0 and 1. Then for any given value of `balance`, a prediction can be made for `default`. For example, one might predict `default = Yes` for any individual for whom $p(\text{balance}) > 0.5$. Alternatively, if a company wishes to be conservative in predicting individuals who are at risk for default, then they may choose to use a lower threshold, such as $p(\text{balance}) > 0.1$.

4.3.1 The Logistic Model

How should we model the relationship between $p(X) = \Pr(Y = 1|X)$ and X ? (For convenience we are using the generic 0/1 coding for the response). In Section 4.2 we talked of using a linear regression model to represent these probabilities:

$$p(X) = \beta_0 + \beta_1 X. \quad (4.1)$$

If we use this approach to predict `default=Yes` using `balance`, then we obtain the model shown in the left-hand panel of Figure 4.2. Here we see the problem with this approach: for balances close to zero we predict a negative probability of default; if we were to predict for very large balances, we would get values bigger than 1. These predictions are not sensible, since of course the true probability of default, regardless of credit card balance, must fall between 0 and 1. This problem is not unique to the credit default data. Any time a straight line is fit to a binary response that is coded as

0 or 1, in principle we can always predict $p(X) < 0$ for some values of X and $p(X) > 1$ for others (unless the range of X is limited).

To avoid this problem, we must model $p(X)$ using a function that gives outputs between 0 and 1 for all values of X . Many functions meet this description. In logistic regression, we use the *logistic function*,

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}. \quad (4.2)$$

To fit the model (4.2), we use a method called *maximum likelihood*, which we discuss in the next section. The right-hand panel of Figure 4.2 illustrates the fit of the logistic regression model to the **Default** data. Notice that for low balances we now predict the probability of default as close to, but never below, zero. Likewise, for high balances we predict a default probability close to, but never above, one. The logistic function will always produce an *S-shaped* curve of this form, and so regardless of the value of X , we will obtain a sensible prediction. We also see that the logistic model is better able to capture the range of probabilities than is the linear regression model in the left-hand plot. The average fitted probability in both cases is 0.0333 (averaged over the training data), which is the same as the overall proportion of defaulters in the data set.

logistic
functionmaximum
likelihood

After a bit of manipulation of (4.2), we find that

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X}. \quad (4.3)$$

The quantity $p(X)/[1 - p(X)]$ is called the *odds*, and can take on any value between 0 and ∞ . Values of the odds close to 0 and ∞ indicate very low and very high probabilities of default, respectively. For example, on average 1 in 5 people with an odds of 1/4 will default, since $p(X) = 0.2$ implies an odds of $\frac{0.2}{1-0.2} = 1/4$. Likewise on average nine out of every ten people with an odds of 9 will default, since $p(X) = 0.9$ implies an odds of $\frac{0.9}{1-0.9} = 9$. Odds are traditionally used instead of probabilities in horse-racing, since they relate more naturally to the correct betting strategy.

odds

By taking the logarithm of both sides of (4.3), we arrive at

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X. \quad (4.4)$$

The left-hand side is called the *log-odds* or *logit*. We see that the logistic regression model (4.2) has a logit that is linear in X .

log-odds
logit

Recall from Chapter 3 that in a linear regression model, β_1 gives the average change in Y associated with a one-unit increase in X . In contrast, in a logistic regression model, increasing X by one unit changes the log odds by β_1 (4.4), or equivalently it multiplies the odds by e^{β_1} (4.3). However, because the relationship between $p(X)$ and X in (4.2) is not a straight line,

β_1 does *not* correspond to the change in $p(X)$ associated with a one-unit increase in X . The amount that $p(X)$ changes due to a one-unit change in X will depend on the current value of X . But regardless of the value of X , if β_1 is positive then increasing X will be associated with increasing $p(X)$, and if β_1 is negative then increasing X will be associated with decreasing $p(X)$. The fact that there is not a straight-line relationship between $p(X)$ and X , and the fact that the rate of change in $p(X)$ per unit change in X depends on the current value of X , can also be seen by inspection of the right-hand panel of Figure 4.2.

4.3.2 Estimating the Regression Coefficients

The coefficients β_0 and β_1 in (4.2) are unknown, and must be estimated based on the available training data. In Chapter 3, we used the least squares approach to estimate the unknown linear regression coefficients. Although we could use (non-linear) least squares to fit the model (4.4), the more general method of *maximum likelihood* is preferred, since it has better statistical properties. The basic intuition behind using maximum likelihood to fit a logistic regression model is as follows: we seek estimates for β_0 and β_1 such that the predicted probability $\hat{p}(x_i)$ of default for each individual, using (4.2), corresponds as closely as possible to the individual's observed default status. In other words, we try to find $\hat{\beta}_0$ and $\hat{\beta}_1$ such that plugging these estimates into the model for $p(X)$, given in (4.2), yields a number close to one for all individuals who defaulted, and a number close to zero for all individuals who did not. This intuition can be formalized using a mathematical equation called a *likelihood function*:

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})). \quad (4.5)$$

likelihood
function

The estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ are chosen to *maximize* this likelihood function.

Maximum likelihood is a very general approach that is used to fit many of the non-linear models that we examine throughout this book. In the linear regression setting, the least squares approach is in fact a special case of maximum likelihood. The mathematical details of maximum likelihood are beyond the scope of this book. However, in general, logistic regression and other models can be easily fit using a statistical software package such as R, and so we do not need to concern ourselves with the details of the maximum likelihood fitting procedure.

Table 4.1 shows the coefficient estimates and related information that result from fitting a logistic regression model on the **Default** data in order to predict the probability of **default=Yes** using **balance**. We see that $\hat{\beta}_1 = 0.0055$; this indicates that an increase in **balance** is associated with an increase in the probability of **default**. To be precise, a one-unit increase in **balance** is associated with an increase in the log odds of **default** by 0.0055 units.

	Coefficient	Std. error	Z-statistic	P-value
Intercept	-10.6513	0.3612	-29.5	<0.0001
balance	0.0055	0.0002	24.9	<0.0001

TABLE 4.1. For the `Default` data, estimated coefficients of the logistic regression model that predicts the probability of `default` using `balance`. A one-unit increase in `balance` is associated with an increase in the log odds of `default` by 0.0055 units.

Many aspects of the logistic regression output shown in Table 4.1 are similar to the linear regression output of Chapter 3. For example, we can measure the accuracy of the coefficient estimates by computing their standard errors. The z -statistic in Table 4.1 plays the same role as the t -statistic in the linear regression output, for example in Table 3.1 on page 68. For instance, the z -statistic associated with β_1 is equal to $\hat{\beta}_1/SE(\hat{\beta}_1)$, and so a large (absolute) value of the z -statistic indicates evidence against the null hypothesis $H_0 : \beta_1 = 0$. This null hypothesis implies that $p(X) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$ —in other words, that the probability of `default` does not depend on `balance`. Since the p-value associated with `balance` in Table 4.1 is tiny, we can reject H_0 . In other words, we conclude that there is indeed an association between `balance` and probability of `default`. The estimated intercept in Table 4.1 is typically not of interest; its main purpose is to adjust the average fitted probabilities to the proportion of ones in the data.

4.3.3 Making Predictions

Once the coefficients have been estimated, it is a simple matter to compute the probability of `default` for any given credit card balance. For example, using the coefficient estimates given in Table 4.1, we predict that the default probability for an individual with a `balance` of \$1,000 is

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X}} = \frac{e^{-10.6513 + 0.0055 \times 1,000}}{1 + e^{-10.6513 + 0.0055 \times 1,000}} = 0.00576,$$

which is below 1%. In contrast, the predicted probability of default for an individual with a balance of \$2,000 is much higher, and equals 0.586 or 58.6%.

One can use qualitative predictors with the logistic regression model using the dummy variable approach from Section 3.3.1. As an example, the `Default` data set contains the qualitative variable `student`. To fit the model we simply create a dummy variable that takes on a value of 1 for students and 0 for non-students. The logistic regression model that results from predicting probability of default from student status can be seen in Table 4.2. The coefficient associated with the dummy variable is positive,

	Coefficient	Std. error	Z-statistic	P-value
Intercept	-3.5041	0.0707	-49.55	<0.0001
student [Yes]	0.4049	0.1150	3.52	0.0004

TABLE 4.2. For the `Default` data, estimated coefficients of the logistic regression model that predicts the probability of `default` using student status. Student status is encoded as a dummy variable, with a value of 1 for a student and a value of 0 for a non-student, and represented by the variable `student [Yes]` in the table.

and the associated p-value is statistically significant. This indicates that students tend to have higher default probabilities than non-students:

$$\widehat{\Pr}(\text{default}=\text{Yes}|\text{student}=\text{Yes}) = \frac{e^{-3.5041+0.4049\times 1}}{1+e^{-3.5041+0.4049\times 1}} = 0.0431,$$

$$\widehat{\Pr}(\text{default}=\text{Yes}|\text{student}=\text{No}) = \frac{e^{-3.5041+0.4049\times 0}}{1+e^{-3.5041+0.4049\times 0}} = 0.0292.$$

4.3.4 Multiple Logistic Regression

We now consider the problem of predicting a binary response using multiple predictors. By analogy with the extension from simple to multiple linear regression in Chapter 3, we can generalize (4.4) as follows:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p, \quad (4.6)$$

where $X = (X_1, \dots, X_p)$ are p predictors. Equation 4.6 can be rewritten as

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}. \quad (4.7)$$

Just as in Section 4.3.2, we use the maximum likelihood method to estimate $\beta_0, \beta_1, \dots, \beta_p$.

Table 4.3 shows the coefficient estimates for a logistic regression model that uses `balance`, `income` (in thousands of dollars), and `student` status to predict probability of `default`. There is a surprising result here. The p-values associated with `balance` and the dummy variable for `student` status are very small, indicating that each of these variables is associated with the probability of `default`. However, the coefficient for the dummy variable is negative, indicating that students are less likely to default than non-students. In contrast, the coefficient for the dummy variable is positive in Table 4.2. How is it possible for student status to be associated with an *increase* in probability of default in Table 4.2 and a *decrease* in probability of default in Table 4.3? The left-hand panel of Figure 4.3 provides a graphical illustration of this apparent paradox. The orange and blue solid lines show the average default rates for students and non-students, respectively,

	Coefficient	Std. error	Z-statistic	P-value
Intercept	-10.8690	0.4923	-22.08	<0.0001
balance	0.0057	0.0002	24.74	<0.0001
income	0.0030	0.0082	0.37	0.7115
student [Yes]	-0.6468	0.2362	-2.74	0.0062

TABLE 4.3. For the `Default` data, estimated coefficients of the logistic regression model that predicts the probability of `default` using `balance`, `income`, and student status. Student status is encoded as a dummy variable `student[Yes]`, with a value of 1 for a student and a value of 0 for a non-student. In fitting this model, `income` was measured in thousands of dollars.

as a function of credit card balance. The negative coefficient for `student` in the multiple logistic regression indicates that *for a fixed value of `balance` and `income`*, a student is less likely to default than a non-student. Indeed, we observe from the left-hand panel of Figure 4.3 that the student default rate is at or below that of the non-student default rate for every value of `balance`. But the horizontal broken lines near the base of the plot, which show the default rates for students and non-students averaged over all values of `balance` and `income`, suggest the opposite effect: the overall student default rate is higher than the non-student default rate. Consequently, there is a positive coefficient for `student` in the single variable logistic regression output shown in Table 4.2.

The right-hand panel of Figure 4.3 provides an explanation for this discrepancy. The variables `student` and `balance` are correlated. Students tend to hold higher levels of debt, which is in turn associated with higher probability of default. In other words, students are more likely to have large credit card balances, which, as we know from the left-hand panel of Figure 4.3, tend to be associated with high default rates. Thus, even though an individual student with a given credit card balance will tend to have a lower probability of default than a non-student with the same credit card balance, the fact that students on the whole tend to have higher credit card balances means that overall, students tend to default at a higher rate than non-students. This is an important distinction for a credit card company that is trying to determine to whom they should offer credit. A student is riskier than a non-student if no information about the student's credit card balance is available. However, that student is less risky than a non-student *with the same credit card balance!*

This simple example illustrates the dangers and subtleties associated with performing regressions involving only a single predictor when other predictors may also be relevant. As in the linear regression setting, the results obtained using one predictor may be quite different from those obtained using multiple predictors, especially when there is correlation among the predictors. In general, the phenomenon seen in Figure 4.3 is known as *confounding*.

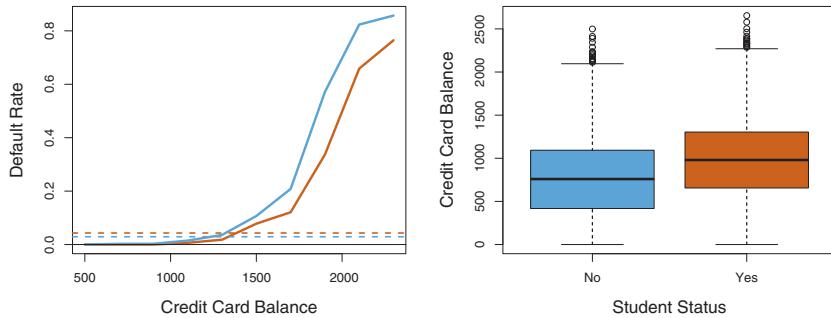


FIGURE 4.3. Confounding in the `Default` data. Left: Default rates are shown for students (orange) and non-students (blue). The solid lines display default rate as a function of `balance`, while the horizontal broken lines display the overall default rates. Right: Boxplots of `balance` for students (orange) and non-students (blue) are shown.

By substituting estimates for the regression coefficients from Table 4.3 into (4.7), we can make predictions. For example, a student with a credit card balance of \$1,500 and an income of \$40,000 has an estimated probability of default of

$$\hat{p}(X) = \frac{e^{-10.869+0.00574 \times 1,500 + 0.003 \times 40 - 0.6468 \times 1}}{1 + e^{-10.869+0.00574 \times 1,500 + 0.003 \times 40 - 0.6468 \times 1}} = 0.058. \quad (4.8)$$

A non-student with the same balance and income has an estimated probability of default of

$$\hat{p}(X) = \frac{e^{-10.869+0.00574 \times 1,500 + 0.003 \times 40 - 0.6468 \times 0}}{1 + e^{-10.869+0.00574 \times 1,500 + 0.003 \times 40 - 0.6468 \times 0}} = 0.105. \quad (4.9)$$

(Here we multiply the `income` coefficient estimate from Table 4.3 by 40, rather than by 40,000, because in that table the model was fit with `income` measured in units of \$1,000.)

4.3.5 Logistic Regression for >2 Response Classes

We sometimes wish to classify a response variable that has more than two classes. For example, in Section 4.2 we had three categories of medical condition in the emergency room: `stroke`, `drug overdose`, `epileptic seizure`. In this setting, we wish to model both $\Pr(Y = \text{stroke}|X)$ and $\Pr(Y = \text{drug overdose}|X)$, with the remaining $\Pr(Y = \text{epileptic seizure}|X) = 1 - \Pr(Y = \text{stroke}|X) - \Pr(Y = \text{drug overdose}|X)$. The two-class logistic regression models discussed in the previous sections have multiple-class extensions, but in practice they tend not to be used all that often. One of the reasons is that the method we discuss in the next section, *discriminant*

analysis, is popular for multiple-class classification. So we do not go into the details of multiple-class logistic regression here, but simply note that such an approach is possible, and that software for it is available in [R](#).

4.4 Linear Discriminant Analysis

Logistic regression involves directly modeling $\Pr(Y = k|X = x)$ using the logistic function, given by (4.7) for the case of two response classes. In statistical jargon, we model the conditional distribution of the response Y , given the predictor(s) X . We now consider an alternative and less direct approach to estimating these probabilities. In this alternative approach, we model the distribution of the predictors X separately in each of the response classes (i.e. given Y), and then use Bayes' theorem to flip these around into estimates for $\Pr(Y = k|X = x)$. When these distributions are assumed to be normal, it turns out that the model is very similar in form to logistic regression.

Why do we need another method, when we have logistic regression? There are several reasons:

- When the classes are well-separated, the parameter estimates for the logistic regression model are surprisingly unstable. Linear discriminant analysis does not suffer from this problem.
- If n is small and the distribution of the predictors X is approximately normal in each of the classes, the linear discriminant model is again more stable than the logistic regression model.
- As mentioned in Section 4.3.5, linear discriminant analysis is popular when we have more than two response classes.

4.4.1 Using Bayes' Theorem for Classification

Suppose that we wish to classify an observation into one of K classes, where $K \geq 2$. In other words, the qualitative response variable Y can take on K possible distinct and unordered values. Let π_k represent the overall or *prior* probability that a randomly chosen observation comes from the k th class; this is the probability that a given observation is associated with the k th category of the response variable Y . Let $f_k(x) \equiv \Pr(X = x|Y = k)$ denote the *density function* of X for an observation that comes from the k th class. In other words, $f_k(x)$ is relatively large if there is a high probability that an observation in the k th class has $X \approx x$, and $f_k(x)$ is small if it is very

density
function

unlikely that an observation in the k th class has $X \approx x$. Then *Bayes' theorem* states that

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}. \quad (4.10)$$

In accordance with our earlier notation, we will use the abbreviation $p_k(X) = \Pr(Y = k|X)$. This suggests that instead of directly computing $p_k(X)$ as in Section 4.3.1, we can simply plug in estimates of π_k and $f_k(X)$ into (4.10). In general, estimating π_k is easy if we have a random sample of Y s from the population: we simply compute the fraction of the training observations that belong to the k th class. However, estimating $f_k(X)$ tends to be more challenging, unless we assume some simple forms for these densities. We refer to $p_k(x)$ as the *posterior* probability that an observation $X = x$ belongs to the k th class. That is, it is the probability that the observation belongs to the k th class, *given* the predictor value for that observation.

We know from Chapter 2 that the Bayes classifier, which classifies an observation to the class for which $p_k(X)$ is largest, has the lowest possible error rate out of all classifiers. (This is of course only true if the terms in (4.10) are all correctly specified.) Therefore, if we can find a way to estimate $f_k(X)$, then we can develop a classifier that approximates the Bayes classifier. Such an approach is the topic of the following sections.

4.4.2 Linear Discriminant Analysis for $p = 1$

For now, assume that $p = 1$ —that is, we have only one predictor. We would like to obtain an estimate for $f_k(x)$ that we can plug into (4.10) in order to estimate $p_k(x)$. We will then classify an observation to the class for which $p_k(x)$ is greatest. In order to estimate $f_k(x)$, we will first make some assumptions about its form.

Suppose we assume that $f_k(x)$ is *normal* or *Gaussian*. In the one-dimensional setting, the normal density takes the form

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right), \quad (4.11)$$

where μ_k and σ_k^2 are the mean and variance parameters for the k th class. For now, let us further assume that $\sigma_1^2 = \dots = \sigma_K^2$: that is, there is a shared variance term across all K classes, which for simplicity we can denote by σ^2 . Plugging (4.11) into (4.10), we find that

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^K \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)}. \quad (4.12)$$

(Note that in (4.12), π_k denotes the prior probability that an observation belongs to the k th class, not to be confused with $\pi \approx 3.14159$, the mathematical constant.) The Bayes classifier involves assigning an observation

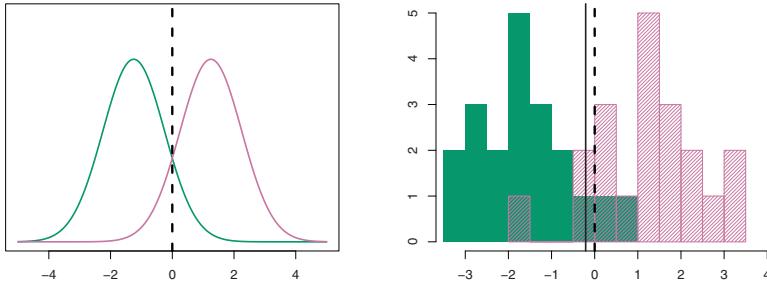


FIGURE 4.4. Left: Two one-dimensional normal density functions are shown. The dashed vertical line represents the Bayes decision boundary. Right: 20 observations were drawn from each of the two classes, and are shown as histograms. The Bayes decision boundary is again shown as a dashed vertical line. The solid vertical line represents the LDA decision boundary estimated from the training data.

$X = x$ to the class for which (4.12) is largest. Taking the log of (4.12) and rearranging the terms, it is not hard to show that this is equivalent to assigning the observation to the class for which

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \quad (4.13)$$

is largest. For instance, if $K = 2$ and $\pi_1 = \pi_2$, then the Bayes classifier assigns an observation to class 1 if $2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2$, and to class 2 otherwise. In this case, the Bayes decision boundary corresponds to the point where

$$x = \frac{\mu_1^2 - \mu_2^2}{2(\mu_1 - \mu_2)} = \frac{\mu_1 + \mu_2}{2}. \quad (4.14)$$

An example is shown in the left-hand panel of Figure 4.4. The two normal density functions that are displayed, $f_1(x)$ and $f_2(x)$, represent two distinct classes. The mean and variance parameters for the two density functions are $\mu_1 = -1.25$, $\mu_2 = 1.25$, and $\sigma_1^2 = \sigma_2^2 = 1$. The two densities overlap, and so given that $X = x$, there is some uncertainty about the class to which the observation belongs. If we assume that an observation is equally likely to come from either class—that is, $\pi_1 = \pi_2 = 0.5$ —then by inspection of (4.14), we see that the Bayes classifier assigns the observation to class 1 if $x < 0$ and class 2 otherwise. Note that in this case, we can compute the Bayes classifier because we know that X is drawn from a Gaussian distribution within each class, and we know all of the parameters involved. In a real-life situation, we are not able to calculate the Bayes classifier.

In practice, even if we are quite certain of our assumption that X is drawn from a Gaussian distribution within each class, we still have to estimate the parameters μ_1, \dots, μ_K , π_1, \dots, π_K , and σ^2 . The *linear discriminant*

analysis (LDA) method approximates the Bayes classifier by plugging estimates for π_k , μ_k , and σ^2 into (4.13). In particular, the following estimates are used:

$$\begin{aligned}\hat{\mu}_k &= \frac{1}{n_k} \sum_{i:y_i=k} x_i \\ \hat{\sigma}^2 &= \frac{1}{n-K} \sum_{k=1}^K \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2\end{aligned}\quad (4.15)$$

linear
discriminant
analysis

where n is the total number of training observations, and n_k is the number of training observations in the k th class. The estimate for μ_k is simply the average of all the training observations from the k th class, while $\hat{\sigma}^2$ can be seen as a weighted average of the sample variances for each of the K classes. Sometimes we have knowledge of the class membership probabilities π_1, \dots, π_K , which can be used directly. In the absence of any additional information, LDA estimates π_k using the proportion of the training observations that belong to the k th class. In other words,

$$\hat{\pi}_k = n_k/n. \quad (4.16)$$

The LDA classifier plugs the estimates given in (4.15) and (4.16) into (4.13), and assigns an observation $X = x$ to the class for which

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k) \quad (4.17)$$

is largest. The word *linear* in the classifier's name stems from the fact that the *discriminant functions* $\hat{\delta}_k(x)$ in (4.17) are linear functions of x (as opposed to a more complex function of x).

discriminant
function

The right-hand panel of Figure 4.4 displays a histogram of a random sample of 20 observations from each class. To implement LDA, we began by estimating π_k , μ_k , and σ^2 using (4.15) and (4.16). We then computed the decision boundary, shown as a black solid line, that results from assigning an observation to the class for which (4.17) is largest. All points to the left of this line will be assigned to the green class, while points to the right of this line are assigned to the purple class. In this case, since $n_1 = n_2 = 20$, we have $\hat{\pi}_1 = \hat{\pi}_2$. As a result, the decision boundary corresponds to the midpoint between the sample means for the two classes, $(\hat{\mu}_1 + \hat{\mu}_2)/2$. The figure indicates that the LDA decision boundary is slightly to the left of the optimal Bayes decision boundary, which instead equals $(\mu_1 + \mu_2)/2 = 0$. How well does the LDA classifier perform on this data? Since this is simulated data, we can generate a large number of test observations in order to compute the Bayes error rate and the LDA test error rate. These are 10.6% and 11.1%, respectively. In other words, the LDA classifier's error rate is only 0.5% above the smallest possible error rate! This indicates that LDA is performing pretty well on this data set.

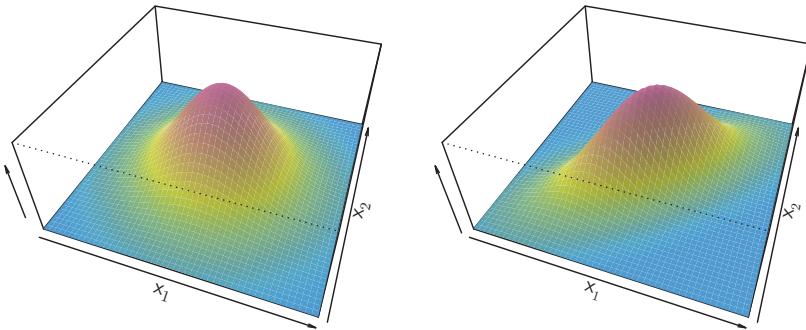


FIGURE 4.5. Two multivariate Gaussian density functions are shown, with $p = 2$. Left: The two predictors are uncorrelated. Right: The two variables have a correlation of 0.7.

To reiterate, the LDA classifier results from assuming that the observations within each class come from a normal distribution with a class-specific mean vector and a common variance σ^2 , and plugging estimates for these parameters into the Bayes classifier. In Section 4.4.4, we will consider a less stringent set of assumptions, by allowing the observations in the k th class to have a class-specific variance, σ_k^2 .

4.4.3 Linear Discriminant Analysis for $p > 1$

We now extend the LDA classifier to the case of multiple predictors. To do this, we will assume that $X = (X_1, X_2, \dots, X_p)$ is drawn from a *multivariate Gaussian* (or multivariate normal) distribution, with a class-specific mean vector and a common covariance matrix. We begin with a brief review of such a distribution.

The multivariate Gaussian distribution assumes that each individual predictor follows a one-dimensional normal distribution, as in (4.11), with some correlation between each pair of predictors. Two examples of multivariate Gaussian distributions with $p = 2$ are shown in Figure 4.5. The height of the surface at any particular point represents the probability that both X_1 and X_2 fall in a small region around that point. In either panel, if the surface is cut along the X_1 axis or along the X_2 axis, the resulting cross-section will have the shape of a one-dimensional normal distribution. The left-hand panel of Figure 4.5 illustrates an example in which $\text{Var}(X_1) = \text{Var}(X_2)$ and $\text{Cor}(X_1, X_2) = 0$; this surface has a characteristic *bell shape*. However, the bell shape will be distorted if the predictors are correlated or have unequal variances, as is illustrated in the right-hand panel of Figure 4.5. In this situation, the base of the bell will have an elliptical, rather than circular,

multivariate Gaussian

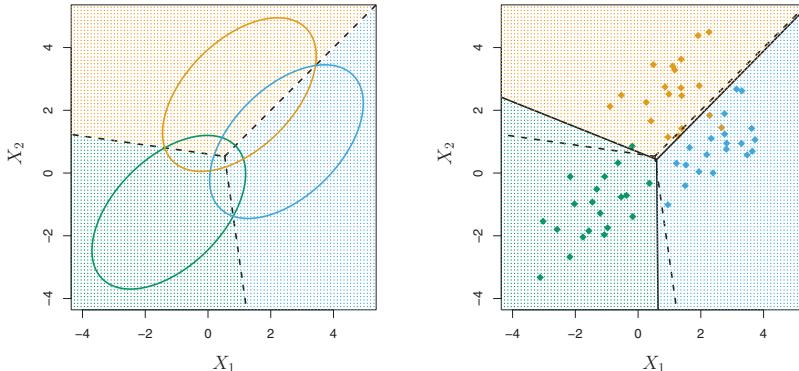


FIGURE 4.6. An example with three classes. The observations from each class are drawn from a multivariate Gaussian distribution with $p = 2$, with a class-specific mean vector and a common covariance matrix. Left: Ellipses that contain 95 % of the probability for each of the three classes are shown. The dashed lines are the Bayes decision boundaries. Right: 20 observations were generated from each class, and the corresponding LDA decision boundaries are indicated using solid black lines. The Bayes decision boundaries are once again shown as dashed lines.

shape. To indicate that a p -dimensional random variable X has a multivariate Gaussian distribution, we write $X \sim N(\mu, \Sigma)$. Here $E(X) = \mu$ is the mean of X (a vector with p components), and $\text{Cov}(X) = \Sigma$ is the $p \times p$ covariance matrix of X . Formally, the multivariate Gaussian density is defined as

$$f(x) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right). \quad (4.18)$$

In the case of $p > 1$ predictors, the LDA classifier assumes that the observations in the k th class are drawn from a multivariate Gaussian distribution $N(\mu_k, \Sigma)$, where μ_k is a class-specific mean vector, and Σ is a covariance matrix that is common to all K classes. Plugging the density function for the k th class, $f_k(X = x)$, into (4.10) and performing a little bit of algebra reveals that the Bayes classifier assigns an observation $X = x$ to the class for which

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k \quad (4.19)$$

is largest. This is the vector/matrix version of (4.13).

An example is shown in the left-hand panel of Figure 4.6. Three equally-sized Gaussian classes are shown with class-specific mean vectors and a common covariance matrix. The three ellipses represent regions that contain 95 % of the probability for each of the three classes. The dashed lines

are the Bayes decision boundaries. In other words, they represent the set of values x for which $\delta_k(x) = \delta_\ell(x)$; i.e.

$$x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k = x^T \Sigma^{-1} \mu_l - \frac{1}{2} \mu_l^T \Sigma^{-1} \mu_l \quad (4.20)$$

for $k \neq l$. (The $\log \pi_k$ term from (4.19) has disappeared because each of the three classes has the same number of training observations; i.e. π_k is the same for each class.) Note that there are three lines representing the Bayes decision boundaries because there are three *pairs of classes* among the three classes. That is, one Bayes decision boundary separates class 1 from class 2, one separates class 1 from class 3, and one separates class 2 from class 3. These three Bayes decision boundaries divide the predictor space into three regions. The Bayes classifier will classify an observation according to the region in which it is located.

Once again, we need to estimate the unknown parameters μ_1, \dots, μ_K , π_1, \dots, π_K , and Σ ; the formulas are similar to those used in the one-dimensional case, given in (4.15). To assign a new observation $X = x$, LDA plugs these estimates into (4.19) and classifies to the class for which $\hat{\delta}_k(x)$ is largest. Note that in (4.19) $\delta_k(x)$ is a linear function of x ; that is, the LDA decision rule depends on x only through a linear combination of its elements. Once again, this is the reason for the word *linear* in LDA.

In the right-hand panel of Figure 4.6, 20 observations drawn from each of the three classes are displayed, and the resulting LDA decision boundaries are shown as solid black lines. Overall, the LDA decision boundaries are pretty close to the Bayes decision boundaries, shown again as dashed lines. The test error rates for the Bayes and LDA classifiers are 0.0746 and 0.0770, respectively. This indicates that LDA is performing well on this data.

We can perform LDA on the **Default** data in order to predict whether or not an individual will default on the basis of credit card balance and student status. The LDA model fit to the 10,000 training samples results in a *training* error rate of 2.75 %. This sounds like a low error rate, but two caveats must be noted.

- First of all, training error rates will usually be lower than test error rates, which are the real quantity of interest. In other words, we might expect this classifier to perform worse if we use it to predict whether or not a new set of individuals will default. The reason is that we specifically adjust the parameters of our model to do well on the training data. The higher the ratio of parameters p to number of samples n , the more we expect this *overfitting* to play a role. For these data we don't expect this to be a problem, since $p = 2$ and $n = 10,000$.
- Second, since only 3.33 % of the individuals in the training sample defaulted, a simple but useless classifier that always predicts that

overfitting

		<i>True default status</i>		
		No	Yes	Total
<i>Predicted default status</i>	No	9,644	252	9,896
	Yes	23	81	104
	Total	9,667	333	10,000

TABLE 4.4. A confusion matrix compares the LDA predictions to the true default statuses for the 10,000 training observations in the **Default** data set. Elements on the diagonal of the matrix represent individuals whose default statuses were correctly predicted, while off-diagonal elements represent individuals that were misclassified. LDA made incorrect predictions for 23 individuals who did not default and for 252 individuals who did default.

each individual will not default, regardless of his or her credit card balance and student status, will result in an error rate of 3.33 %. In other words, the trivial *null* classifier will achieve an error rate that is only a bit higher than the LDA training set error rate.

In practice, a binary classifier such as this one can make two types of errors: it can incorrectly assign an individual who defaults to the *no default* category, or it can incorrectly assign an individual who does not default to the *default* category. It is often of interest to determine which of these two types of errors are being made. A *confusion matrix*, shown for the **Default** data in Table 4.4, is a convenient way to display this information. The table reveals that LDA predicted that a total of 104 people would default. Of these people, 81 actually defaulted and 23 did not. Hence only 23 out of 9,667 of the individuals who did not default were incorrectly labeled. This looks like a pretty low error rate! However, of the 333 individuals who defaulted, 252 (or 75.7 %) were missed by LDA. So while the overall error rate is low, the error rate among individuals who defaulted is very high. From the perspective of a credit card company that is trying to identify high-risk individuals, an error rate of $252/333 = 75.7\%$ among individuals who default may well be unacceptable.

Class-specific performance is also important in medicine and biology, where the terms *sensitivity* and *specificity* characterize the performance of a classifier or screening test. In this case the sensitivity is the percentage of true defaulters that are identified, a low 24.3 % in this case. The specificity is the percentage of non-defaulters that are correctly identified, here $(1 - 23/9,667) \times 100 = 99.8\%$.

Why does LDA do such a poor job of classifying the customers who default? In other words, why does it have such a low sensitivity? As we have seen, LDA is trying to approximate the Bayes classifier, which has the lowest *total* error rate out of all classifiers (if the Gaussian model is correct). That is, the Bayes classifier will yield the smallest possible total number of misclassified observations, irrespective of which class the errors come from. That is, some misclassifications will result from incorrectly assigning

null

confusion matrix

sensitivity
specificity

		<i>True default status</i>		Total
		No	Yes	
<i>Predicted default status</i>	No	9,432	138	9,570
	Yes	235	195	430
	Total	9,667	333	10,000

TABLE 4.5. A confusion matrix compares the LDA predictions to the true default statuses for the 10,000 training observations in the **Default** data set, using a modified threshold value that predicts default for any individuals whose posterior default probability exceeds 20 %.

a customer who does not default to the default class, and others will result from incorrectly assigning a customer who defaults to the non-default class. In contrast, a credit card company might particularly wish to avoid incorrectly classifying an individual who will default, whereas incorrectly classifying an individual who will not default, though still to be avoided, is less problematic. We will now see that it is possible to modify LDA in order to develop a classifier that better meets the credit card company's needs.

The Bayes classifier works by assigning an observation to the class for which the posterior probability $p_k(X)$ is greatest. In the two-class case, this amounts to assigning an observation to the *default* class if

$$\Pr(\text{default} = \text{Yes} | X = x) > 0.5. \quad (4.21)$$

Thus, the Bayes classifier, and by extension LDA, uses a threshold of 50 % for the posterior probability of default in order to assign an observation to the *default* class. However, if we are concerned about incorrectly predicting the default status for individuals who default, then we can consider lowering this threshold. For instance, we might label any customer with a posterior probability of default above 20 % to the *default* class. In other words, instead of assigning an observation to the *default* class if (4.21) holds, we could instead assign an observation to this class if

$$\Pr(\text{default} = \text{Yes} | X = x) > 0.2. \quad (4.22)$$

The error rates that result from taking this approach are shown in Table 4.5. Now LDA predicts that 430 individuals will default. Of the 333 individuals who default, LDA correctly predicts all but 138, or 41.4 %. This is a vast improvement over the error rate of 75.7 % that resulted from using the threshold of 50 %. However, this improvement comes at a cost: now 235 individuals who do not default are incorrectly classified. As a result, the overall error rate has increased slightly to 3.73 %. But a credit card company may consider this slight increase in the total error rate to be a small price to pay for more accurate identification of individuals who do indeed default.

Figure 4.7 illustrates the trade-off that results from modifying the threshold value for the posterior probability of default. Various error rates are

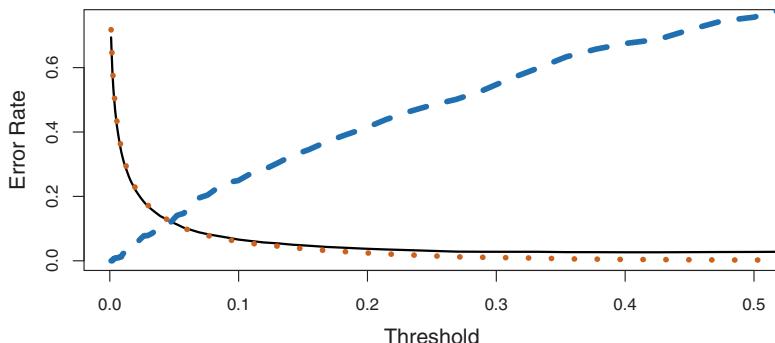


FIGURE 4.7. For the **Default** data set, error rates are shown as a function of the threshold value for the posterior probability that is used to perform the assignment. The black solid line displays the overall error rate. The blue dashed line represents the fraction of defaulting customers that are incorrectly classified, and the orange dotted line indicates the fraction of errors among the non-defaulting customers.

shown as a function of the threshold value. Using a threshold of 0.5, as in (4.21), minimizes the overall error rate, shown as a black solid line. This is to be expected, since the Bayes classifier uses a threshold of 0.5 and is known to have the lowest overall error rate. But when a threshold of 0.5 is used, the error rate among the individuals who default is quite high (blue dashed line). As the threshold is reduced, the error rate among individuals who default decreases steadily, but the error rate among the individuals who do not default increases. How can we decide which threshold value is best? Such a decision must be based on *domain knowledge*, such as detailed information about the costs associated with default.

The *ROC curve* is a popular graphic for simultaneously displaying the two types of errors for all possible thresholds. The name “ROC” is historic, and comes from communications theory. It is an acronym for *receiver operating characteristics*. Figure 4.8 displays the ROC curve for the LDA classifier on the training data. The overall performance of a classifier, summarized over all possible thresholds, is given by the *area under the (ROC) curve* (AUC). An ideal ROC curve will hug the top left corner, so the larger the AUC the better the classifier. For this data the AUC is 0.95, which is close to the maximum of one so would be considered very good. We expect a classifier that performs no better than chance to have an AUC of 0.5 (when evaluated on an independent test set not used in model training). ROC curves are useful for comparing different classifiers, since they take into account all possible thresholds. It turns out that the ROC curve for the logistic regression model of Section 4.3.4 fit to these data is virtually indistinguishable from this one for the LDA model, so we do not display it here.

As we have seen above, varying the classifier threshold changes its true positive and false positive rate. These are also called the *sensitivity* and one

ROC curve

area under
the (ROC)
curve

sensitivity

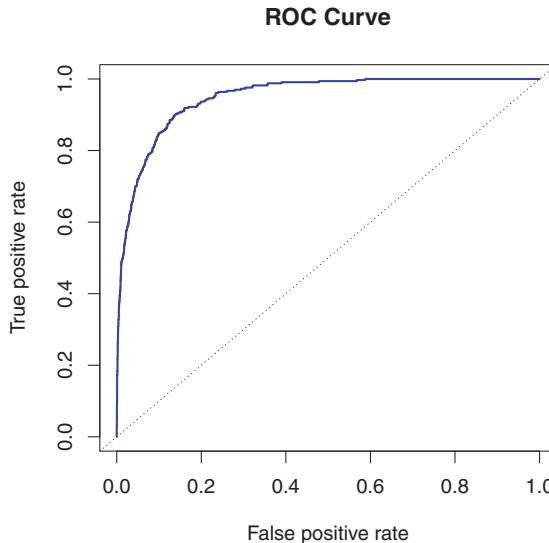


FIGURE 4.8. A ROC curve for the LDA classifier on the **Default** data. It traces out two types of error as we vary the threshold value for the posterior probability of default. The actual thresholds are not shown. The true positive rate is the sensitivity: the fraction of defaulters that are correctly identified, using a given threshold value. The false positive rate is 1-specificity: the fraction of non-defaulters that we classify incorrectly as defaulters, using that same threshold value. The ideal ROC curve hugs the top left corner, indicating a high true positive rate and a low false positive rate. The dotted line represents the “no information” classifier; this is what we would expect if student status and credit card balance are not associated with probability of default.

		Predicted class			
		– or Null	+ or Non-null	Total	
True class	– or Null	True Neg. (TN)	False Pos. (FP)	N	
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P	
Total		N*	P*		

TABLE 4.6. Possible results when applying a classifier or diagnostic test to a population.

minus the *specificity* of our classifier. Since there is an almost bewildering array of terms used in this context, we now give a summary. Table 4.6 shows the possible results when applying a classifier (or diagnostic test) to a population. To make the connection with the epidemiology literature, we think of “+” as the “disease” that we are trying to detect, and “−” as the “non-disease” state. To make the connection to the classical hypothesis testing literature, we think of “−” as the null hypothesis and “+” as the alternative (non-null) hypothesis. In the context of the **Default** data, “+” indicates an individual who defaults, and “−” indicates one who does not.

Name	Definition	Synonyms
False Pos. rate	FP/N	Type I error, 1–Specificity
True Pos. rate	TP/P	1–Type II error, power, sensitivity, recall
Pos. Pred. value	TP/P*	Precision, 1–false discovery proportion
Neg. Pred. value	TN/N*	

TABLE 4.7. Important measures for classification and diagnostic testing, derived from quantities in Table 4.6.

Table 4.7 lists many of the popular performance measures that are used in this context. The denominators for the false positive and true positive rates are the actual population counts in each class. In contrast, the denominators for the positive predictive value and the negative predictive value are the total predicted counts for each class.

4.4.4 Quadratic Discriminant Analysis

As we have discussed, LDA assumes that the observations within each class are drawn from a multivariate Gaussian distribution with a class-specific mean vector and a covariance matrix that is common to all K classes. *Quadratic discriminant analysis* (QDA) provides an alternative approach. Like LDA, the QDA classifier results from assuming that the observations from each class are drawn from a Gaussian distribution, and plugging estimates for the parameters into Bayes' theorem in order to perform prediction. However, unlike LDA, QDA assumes that each class has its own covariance matrix. That is, it assumes that an observation from the k th class is of the form $X \sim N(\mu_k, \Sigma_k)$, where Σ_k is a covariance matrix for the k th class. Under this assumption, the Bayes classifier assigns an observation $X = x$ to the class for which

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) - \frac{1}{2} \log |\Sigma_k| + \log \pi_k \\ &= -\frac{1}{2}x^T \Sigma_k^{-1} x + x^T \Sigma_k^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma_k^{-1} \mu_k - \frac{1}{2} \log |\Sigma_k| + \log \pi_k\end{aligned}\tag{4.23}$$

quadratic
discriminant
analysis

is largest. So the QDA classifier involves plugging estimates for Σ_k , μ_k , and π_k into (4.23), and then assigning an observation $X = x$ to the class for which this quantity is largest. Unlike in (4.19), the quantity x appears as a *quadratic* function in (4.23). This is where QDA gets its name.

Why does it matter whether or not we assume that the K classes share a common covariance matrix? In other words, why would one prefer LDA to QDA, or vice-versa? The answer lies in the bias-variance trade-off. When there are p predictors, then estimating a covariance matrix requires estimating $p(p+1)/2$ parameters. QDA estimates a separate covariance matrix for each class, for a total of $Kp(p+1)/2$ parameters. With 50 predictors this

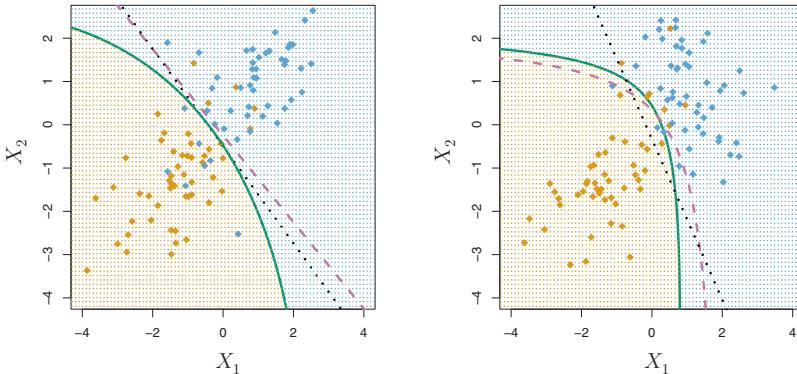


FIGURE 4.9. Left: The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with $\Sigma_1 = \Sigma_2$. The shading indicates the QDA decision rule. Since the Bayes decision boundary is linear, it is more accurately approximated by LDA than by QDA. Right: Details are as given in the left-hand panel, except that $\Sigma_1 \neq \Sigma_2$. Since the Bayes decision boundary is non-linear, it is more accurately approximated by QDA than by LDA.

is some multiple of 1,275, which is a lot of parameters. By instead assuming that the K classes share a common covariance matrix, the LDA model becomes linear in x , which means there are Kp linear coefficients to estimate. Consequently, LDA is a much less flexible classifier than QDA, and so has substantially lower variance. This can potentially lead to improved prediction performance. But there is a trade-off: if LDA's assumption that the K classes share a common covariance matrix is badly off, then LDA can suffer from high bias. Roughly speaking, LDA tends to be a better bet than QDA if there are relatively few training observations and so reducing variance is crucial. In contrast, QDA is recommended if the training set is very large, so that the variance of the classifier is not a major concern, or if the assumption of a common covariance matrix for the K classes is clearly untenable.

Figure 4.9 illustrates the performances of LDA and QDA in two scenarios. In the left-hand panel, the two Gaussian classes have a common correlation of 0.7 between X_1 and X_2 . As a result, the Bayes decision boundary is linear and is accurately approximated by the LDA decision boundary. The QDA decision boundary is inferior, because it suffers from higher variance without a corresponding decrease in bias. In contrast, the right-hand panel displays a situation in which the orange class has a correlation of 0.7 between the variables and the blue class has a correlation of -0.7. Now the Bayes decision boundary is quadratic, and so QDA more accurately approximates this boundary than does LDA.

4.5 A Comparison of Classification Methods

In this chapter, we have considered three different classification approaches: logistic regression, LDA, and QDA. In Chapter 2, we also discussed the K -nearest neighbors (KNN) method. We now consider the types of scenarios in which one approach might dominate the others.

Though their motivations differ, the logistic regression and LDA methods are closely connected. Consider the two-class setting with $p = 1$ predictor, and let $p_1(x)$ and $p_2(x) = 1 - p_1(x)$ be the probabilities that the observation $X = x$ belongs to class 1 and class 2, respectively. In the LDA framework, we can see from (4.12) to (4.13) (and a bit of simple algebra) that the log odds is given by

$$\log \left(\frac{p_1(x)}{1 - p_1(x)} \right) = \log \left(\frac{p_1(x)}{p_2(x)} \right) = c_0 + c_1 x, \quad (4.24)$$

where c_0 and c_1 are functions of μ_1, μ_2 , and σ^2 . From (4.4), we know that in logistic regression,

$$\log \left(\frac{p_1}{1 - p_1} \right) = \beta_0 + \beta_1 x. \quad (4.25)$$

Both (4.24) and (4.25) are linear functions of x . Hence, both logistic regression and LDA produce linear decision boundaries. The only difference between the two approaches lies in the fact that β_0 and β_1 are estimated using maximum likelihood, whereas c_0 and c_1 are computed using the estimated mean and variance from a normal distribution. This same connection between LDA and logistic regression also holds for multidimensional data with $p > 1$.

Since logistic regression and LDA differ only in their fitting procedures, one might expect the two approaches to give similar results. This is often, but not always, the case. LDA assumes that the observations are drawn from a Gaussian distribution with a common covariance matrix in each class, and so can provide some improvements over logistic regression when this assumption approximately holds. Conversely, logistic regression can outperform LDA if these Gaussian assumptions are not met.

Recall from Chapter 2 that KNN takes a completely different approach from the classifiers seen in this chapter. In order to make a prediction for an observation $X = x$, the K training observations that are closest to x are identified. Then X is assigned to the class to which the plurality of these observations belong. Hence KNN is a completely non-parametric approach: no assumptions are made about the shape of the decision boundary. Therefore, we can expect this approach to dominate LDA and logistic regression when the decision boundary is highly non-linear. On the other hand, KNN does not tell us which predictors are important; we don't get a table of coefficients as in Table 4.3.

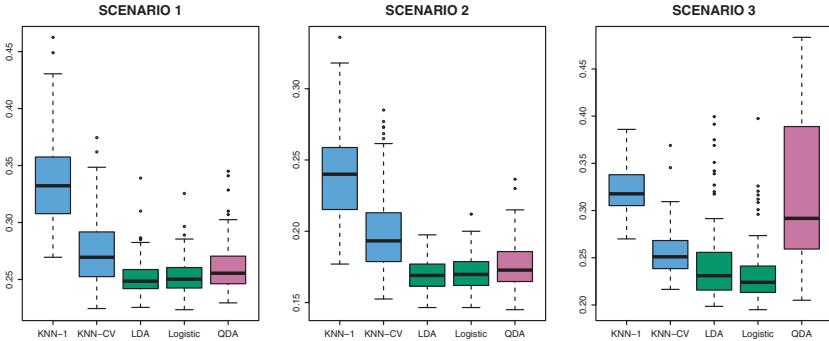


FIGURE 4.10. Boxplots of the test error rates for each of the linear scenarios described in the main text.

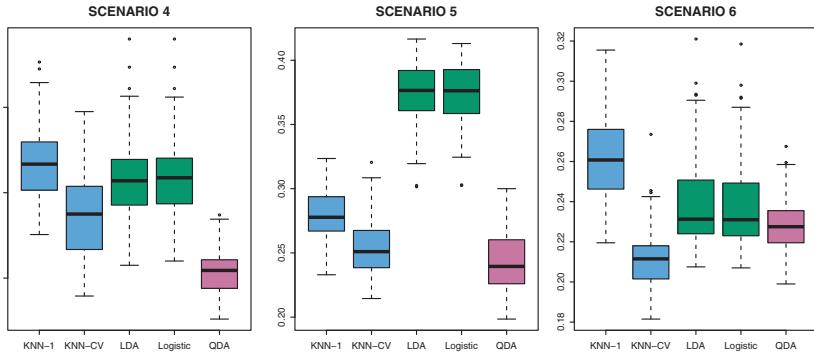


FIGURE 4.11. Boxplots of the test error rates for each of the non-linear scenarios described in the main text.

Finally, QDA serves as a compromise between the non-parametric KNN method and the linear LDA and logistic regression approaches. Since QDA assumes a quadratic decision boundary, it can accurately model a wider range of problems than can the linear methods. Though not as flexible as KNN, QDA can perform better in the presence of a limited number of training observations because it does make some assumptions about the form of the decision boundary.

To illustrate the performances of these four classification approaches, we generated data from six different scenarios. In three of the scenarios, the Bayes decision boundary is linear, and in the remaining scenarios it is non-linear. For each scenario, we produced 100 random training data sets. On each of these training sets, we fit each method to the data and computed the resulting test error rate on a large test set. Results for the linear scenarios are shown in Figure 4.10, and the results for the non-linear scenarios are in Figure 4.11. The KNN method requires selection of K , the number of neighbors. We performed KNN with two values of K : $K = 1$,

and a value of K that was chosen automatically using an approach called *cross-validation*, which we discuss further in Chapter 5.

In each of the six scenarios, there were $p = 2$ predictors. The scenarios were as follows:

Scenario 1: There were 20 training observations in each of two classes. The observations within each class were uncorrelated random normal variables with a different mean in each class. The left-hand panel of Figure 4.10 shows that LDA performed well in this setting, as one would expect since this is the model assumed by LDA. KNN performed poorly because it paid a price in terms of variance that was not offset by a reduction in bias. QDA also performed worse than LDA, since it fit a more flexible classifier than necessary. Since logistic regression assumes a linear decision boundary, its results were only slightly inferior to those of LDA.

Scenario 2: Details are as in Scenario 1, except that within each class, the two predictors had a correlation of -0.5 . The center panel of Figure 4.10 indicates little change in the relative performances of the methods as compared to the previous scenario.

Scenario 3: We generated X_1 and X_2 from the *t-distribution*, with 50 observations per class. The *t-distribution* has a similar shape to the normal distribution, but it has a tendency to yield more extreme points—that is, more points that are far from the mean. In this setting, the decision boundary was still linear, and so fit into the logistic regression framework. The set-up violated the assumptions of LDA, since the observations were not drawn from a normal distribution. The right-hand panel of Figure 4.10 shows that logistic regression outperformed LDA, though both methods were superior to the other approaches. In particular, the QDA results deteriorated considerably as a consequence of non-normality.

Scenario 4: The data were generated from a normal distribution, with a correlation of 0.5 between the predictors in the first class, and correlation of -0.5 between the predictors in the second class. This setup corresponded to the QDA assumption, and resulted in quadratic decision boundaries. The left-hand panel of Figure 4.11 shows that QDA outperformed all of the other approaches.

Scenario 5: Within each class, the observations were generated from a normal distribution with uncorrelated predictors. However, the responses were sampled from the logistic function using X_1^2 , X_2^2 , and $X_1 \times X_2$ as predictors. Consequently, there is a quadratic decision boundary. The center panel of Figure 4.11 indicates that QDA once again performed best, followed closely by KNN-CV. The linear methods had poor performance.

Scenario 6: Details are as in the previous scenario, but the responses were sampled from a more complicated non-linear function. As a result, even the quadratic decision boundaries of QDA could not adequately model the data. The right-hand panel of Figure 4.11 shows that QDA gave slightly better results than the linear methods, while the much more flexible KNN-CV method gave the best results. But KNN with $K = 1$ gave the worst results out of all methods. This highlights the fact that even when the data exhibits a complex non-linear relationship, a non-parametric method such as KNN can still give poor results if the level of smoothness is not chosen correctly.

These six examples illustrate that no one method will dominate the others in every situation. When the true decision boundaries are linear, then the LDA and logistic regression approaches will tend to perform well. When the boundaries are moderately non-linear, QDA may give better results. Finally, for much more complicated decision boundaries, a non-parametric approach such as KNN can be superior. But the level of smoothness for a non-parametric approach must be chosen carefully. In the next chapter we examine a number of approaches for choosing the correct level of smoothness and, in general, for selecting the best overall method.

Finally, recall from Chapter 3 that in the regression setting we can accommodate a non-linear relationship between the predictors and the response by performing regression using transformations of the predictors. A similar approach could be taken in the classification setting. For instance, we could create a more flexible version of logistic regression by including X^2 , X^3 , and even X^4 as predictors. This may or may not improve logistic regression's performance, depending on whether the increase in variance due to the added flexibility is offset by a sufficiently large reduction in bias. We could do the same for LDA. If we added all possible quadratic terms and cross-products to LDA, the form of the model would be the same as the QDA model, although the parameter estimates would be different. This device allows us to move somewhere between an LDA and a QDA model.

4.6 Lab: Logistic Regression, LDA, QDA, and KNN

4.6.1 The Stock Market Data

We will begin by examining some numerical and graphical summaries of the `Smarket` data, which is part of the `ISLR` library. This data set consists of percentage returns for the S&P 500 stock index over 1,250 days, from the beginning of 2001 until the end of 2005. For each date, we have recorded the percentage returns for each of the five previous trading days, `Lag1` through `Lag5`. We have also recorded `Volume` (the number of shares traded

4.7 Exercises

Conceptual

- Using a little bit of algebra, prove that (4.2) is equivalent to (4.3). In other words, the logistic function representation and logit representation for the logistic regression model are equivalent.
- It was stated in the text that classifying an observation to the class for which (4.12) is largest is equivalent to classifying an observation to the class for which (4.13) is largest. Prove that this is the case. In other words, under the assumption that the observations in the k th class are drawn from a $N(\mu_k, \sigma^2)$ distribution, the Bayes' classifier assigns an observation to the class for which the discriminant function is maximized.
- This problem relates to the QDA model, in which the observations within each class are drawn from a normal distribution with a class-specific mean vector and a class specific covariance matrix. We consider the simple case where $p = 1$; i.e. there is only one feature.

Suppose that we have K classes, and that if an observation belongs to the k th class then X comes from a one-dimensional normal distribution, $X \sim N(\mu_k, \sigma_k^2)$. Recall that the density function for the one-dimensional normal distribution is given in (4.11). Prove that in this case, the Bayes' classifier is *not* linear. Argue that it is in fact quadratic.

Hint: For this problem, you should follow the arguments laid out in Section 4.4.2, but without making the assumption that $\sigma_1^2 = \dots = \sigma_K^2$.

- When the number of features p is large, there tends to be a deterioration in the performance of KNN and other *local* approaches that perform prediction using only observations that are *near* the test observation for which a prediction must be made. This phenomenon is known as the *curse of dimensionality*, and it ties into the fact that non-parametric approaches often perform poorly when p is large. We will now investigate this curse.



curse of dimensionality

- Suppose that we have a set of observations, each with measurements on $p = 1$ feature, X . We assume that X is uniformly (evenly) distributed on $[0, 1]$. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10% of the range of X closest to that test observation. For instance, in order to predict the response for a test observation with $X = 0.6$,

we will use observations in the range $[0.55, 0.65]$. On average, what fraction of the available observations will we use to make the prediction?

- (b) Now suppose that we have a set of observations, each with measurements on $p = 2$ features, X_1 and X_2 . We assume that (X_1, X_2) are uniformly distributed on $[0, 1] \times [0, 1]$. We wish to predict a test observation's response using only observations that are within 10 % of the range of X_1 and within 10 % of the range of X_2 closest to that test observation. For instance, in order to predict the response for a test observation with $X_1 = 0.6$ and $X_2 = 0.35$, we will use observations in the range $[0.55, 0.65]$ for X_1 and in the range $[0.3, 0.4]$ for X_2 . On average, what fraction of the available observations will we use to make the prediction?
- (c) Now suppose that we have a set of observations on $p = 100$ features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10 % of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make the prediction?
- (d) Using your answers to parts (a)–(c), argue that a drawback of KNN when p is large is that there are very few training observations “near” any given test observation.
- (e) Now suppose that we wish to make a prediction for a test observation by creating a p -dimensional hypercube centered around the test observation that contains, on average, 10 % of the training observations. For $p = 1, 2$, and 100, what is the length of each side of the hypercube? Comment on your answer.

Note: A hypercube is a generalization of a cube to an arbitrary number of dimensions. When $p = 1$, a hypercube is simply a line segment, when $p = 2$ it is a square, and when $p = 100$ it is a 100-dimensional cube.

5. We now examine the differences between LDA and QDA.
 - (a) If the Bayes decision boundary is linear, do we expect LDA or QDA to perform better on the training set? On the test set?
 - (b) If the Bayes decision boundary is non-linear, do we expect LDA or QDA to perform better on the training set? On the test set?
 - (c) In general, as the sample size n increases, do we expect the test prediction accuracy of QDA relative to LDA to improve, decline, or be unchanged? Why?

- (d) True or False: Even if the Bayes decision boundary for a given problem is linear, we will probably achieve a superior test error rate using QDA rather than LDA because QDA is flexible enough to model a linear decision boundary. Justify your answer.
6. Suppose we collect data for a group of students in a statistics class with variables $X_1 = \text{hours studied}$, $X_2 = \text{undergrad GPA}$, and $Y = \text{receive an A}$. We fit a logistic regression and produce estimated coefficient, $\hat{\beta}_0 = -6$, $\hat{\beta}_1 = 0.05$, $\hat{\beta}_2 = 1$.
- Estimate the probability that a student who studies for 40 h and has an undergrad GPA of 3.5 gets an A in the class.
 - How many hours would the student in part (a) need to study to have a 50% chance of getting an A in the class?
7. Suppose that we wish to predict whether a given stock will issue a dividend this year (“Yes” or “No”) based on X , last year’s percent profit. We examine a large number of companies and discover that the mean value of X for companies that issued a dividend was $\bar{X} = 10$, while the mean for those that didn’t was $\bar{X} = 0$. In addition, the variance of X for these two sets of companies was $\hat{\sigma}^2 = 36$. Finally, 80% of companies issued dividends. Assuming that X follows a normal distribution, predict the probability that a company will issue a dividend this year given that its percentage profit was $X = 4$ last year.

Hint: Recall that the density function for a normal random variable is $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$. You will need to use Bayes’ theorem.

8. Suppose that we take a data set, divide it into equally-sized training and test sets, and then try out two different classification procedures. First we use logistic regression and get an error rate of 20% on the training data and 30% on the test data. Next we use 1-nearest neighbors (i.e. $K = 1$) and get an average error rate (averaged over both test and training data sets) of 18%. Based on these results, which method should we prefer to use for classification of new observations? Why?
9. This problem has to do with *odds*.
- On average, what fraction of people with an odds of 0.37 of defaulting on their credit card payment will in fact default?
 - Suppose that an individual has a 16% chance of defaulting on her credit card payment. What are the odds that she will default?

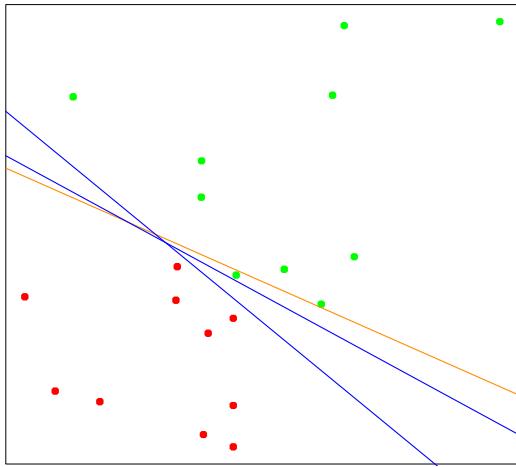


FIGURE 4.14. A toy example with two classes separable by a hyperplane. The orange line is the least squares solution, which misclassifies one of the training points. Also shown are two blue separating hyperplanes found by the perceptron learning algorithm with different random starts.

4.5 Separating Hyperplanes

We have seen that linear discriminant analysis and logistic regression both estimate linear decision boundaries in similar but slightly different ways. For the rest of this chapter we describe separating hyperplane classifiers. These procedures construct linear decision boundaries that explicitly try to separate the data into different classes as well as possible. They provide the basis for support vector classifiers, discussed in Chapter 12. The mathematical level of this section is somewhat higher than that of the previous sections.

Figure 4.14 shows 20 data points in two classes in \mathbb{R}^2 . These data can be separated by a linear boundary. Included in the figure (blue lines) are two of the infinitely many possible *separating hyperplanes*. The orange line is the least squares solution to the problem, obtained by regressing the $-1/1$ response Y on X (with intercept); the line is given by

$$\{x : \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0\}. \quad (4.39)$$

This least squares solution does not do a perfect job in separating the points, and makes one error. This is the same boundary found by LDA, in light of its equivalence with linear regression in the two-class case (Section 4.3 and Exercise 4.2).

Classifiers such as (4.39), that compute a linear combination of the input features and return the sign, were called *perceptrons* in the engineering liter-

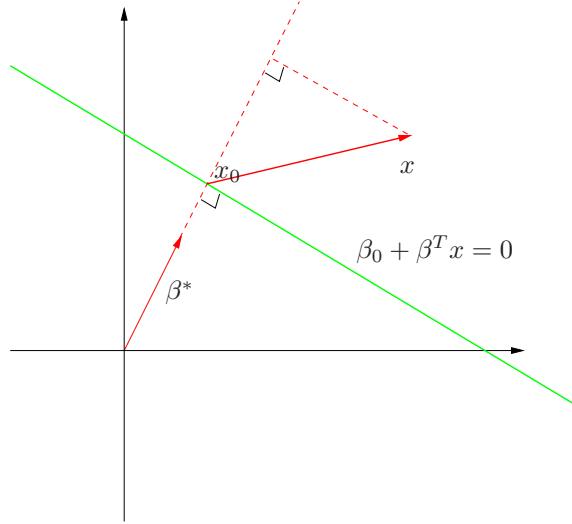


FIGURE 4.15. The linear algebra of a hyperplane (affine set).

ature in the late 1950s (Rosenblatt, 1958). Perceptrons set the foundations for the neural network models of the 1980s and 1990s.

Before we continue, let us digress slightly and review some vector algebra. Figure 4.15 depicts a hyperplane or *affine set* L defined by the equation $f(x) = \beta_0 + \beta^T x = 0$; since we are in \mathbb{R}^2 this is a line.

Here we list some properties:

1. For any two points x_1 and x_2 lying in L , $\beta^T(x_1 - x_2) = 0$, and hence $\beta^* = \beta/\|\beta\|$ is the vector normal to the surface of L .
2. For any point x_0 in L , $\beta^T x_0 = -\beta_0$.
3. The signed distance of any point x to L is given by

$$\begin{aligned}\beta^{*T}(x - x_0) &= \frac{1}{\|\beta\|}(\beta^T x + \beta_0) \\ &= \frac{1}{\|f'(x)\|} f(x).\end{aligned}\tag{4.40}$$

Hence $f(x)$ is proportional to the signed distance from x to the hyperplane defined by $f(x) = 0$.

4.5.1 Rosenblatt's Perceptron Learning Algorithm

The *perceptron learning algorithm* tries to find a separating hyperplane by minimizing the distance of misclassified points to the decision boundary. If

a response $y_i = 1$ is misclassified, then $x_i^T \beta + \beta_0 < 0$, and the opposite for a misclassified response with $y_i = -1$. The goal is to minimize

$$D(\beta, \beta_0) = - \sum_{i \in \mathcal{M}} y_i (x_i^T \beta + \beta_0), \quad (4.41)$$

where \mathcal{M} indexes the set of misclassified points. The quantity is non-negative and proportional to the distance of the misclassified points to the decision boundary defined by $\beta^T x + \beta_0 = 0$. The gradient (assuming \mathcal{M} is fixed) is given by

$$\partial \frac{D(\beta, \beta_0)}{\partial \beta} = - \sum_{i \in \mathcal{M}} y_i x_i, \quad (4.42)$$

$$\partial \frac{D(\beta, \beta_0)}{\partial \beta_0} = - \sum_{i \in \mathcal{M}} y_i. \quad (4.43)$$

The algorithm in fact uses *stochastic gradient descent* to minimize this piecewise linear criterion. This means that rather than computing the sum of the gradient contributions of each observation followed by a step in the negative gradient direction, a step is taken after each observation is visited. Hence the misclassified observations are visited in some sequence, and the parameters β are updated via

$$\begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} \leftarrow \begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} + \rho \begin{pmatrix} y_i x_i \\ y_i \end{pmatrix}. \quad (4.44)$$

Here ρ is the learning rate, which in this case can be taken to be 1 without loss in generality. If the classes are linearly separable, it can be shown that the algorithm converges to a separating hyperplane in a finite number of steps (Exercise 4.6). Figure 4.14 shows two solutions to a toy problem, each started at a different random guess.

There are a number of problems with this algorithm, summarized in Ripley (1996):

- When the data are separable, there are many solutions, and which one is found depends on the starting values.
- The “finite” number of steps can be very large. The smaller the gap, the longer the time to find it.
- When the data are not separable, the algorithm will not converge, and cycles develop. The cycles can be long and therefore hard to detect.

The second problem can often be eliminated by seeking a hyperplane not in the original space, but in a much enlarged space obtained by creating

many basis-function transformations of the original variables. This is analogous to driving the residuals in a polynomial regression problem down to zero by making the degree sufficiently large. Perfect separation cannot always be achieved: for example, if observations from two different classes share the same input. It may not be desirable either, since the resulting model is likely to be overfit and will not generalize well. We return to this point at the end of the next section.

A rather elegant solution to the first problem is to add additional constraints to the separating hyperplane.

4.5.2 Optimal Separating Hyperplanes



The *optimal separating hyperplane* separates the two classes and maximizes the distance to the closest point from either class (Vapnik, 1996). Not only does this provide a unique solution to the separating hyperplane problem, but by maximizing the margin between the two classes on the training data, this leads to better classification performance on test data.

We need to generalize criterion (4.41). Consider the optimization problem

$$\begin{aligned} & \max_{\beta, \beta_0, \|\beta\|=1} M \\ & \text{subject to } y_i(x_i^T \beta + \beta_0) \geq M, \quad i = 1, \dots, N. \end{aligned} \tag{4.45}$$

The set of conditions ensure that all the points are at least a signed distance M from the decision boundary defined by β and β_0 , and we seek the largest such M and associated parameters. We can get rid of the $\|\beta\| = 1$ constraint by replacing the conditions with

$$\frac{1}{\|\beta\|} y_i(x_i^T \beta + \beta_0) \geq M, \tag{4.46}$$

(which redefines β_0) or equivalently

$$y_i(x_i^T \beta + \beta_0) \geq M \|\beta\|. \tag{4.47}$$

Since for any β and β_0 satisfying these inequalities, any positively scaled multiple satisfies them too, we can arbitrarily set $\|\beta\| = 1/M$. Thus (4.45) is equivalent to

$$\begin{aligned} & \min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2 \\ & \text{subject to } y_i(x_i^T \beta + \beta_0) \geq 1, \quad i = 1, \dots, N. \end{aligned} \tag{4.48}$$

In light of (4.40), the constraints define an empty slab or margin around the linear decision boundary of thickness $1/\|\beta\|$. Hence we choose β and β_0 to maximize its thickness. This is a convex optimization problem (quadratic

criterion with linear inequality constraints). The Lagrange (primal) function, to be minimized w.r.t. β and β_0 , is

$$L_P = \frac{1}{2} \|\beta\|^2 - \sum_{i=1}^N \alpha_i [y_i(x_i^T \beta + \beta_0) - 1]. \quad (4.49)$$

Setting the derivatives to zero, we obtain:

$$\beta = \sum_{i=1}^N \alpha_i y_i x_i, \quad (4.50)$$

$$0 = \sum_{i=1}^N \alpha_i y_i, \quad (4.51)$$

and substituting these in (4.49) we obtain the so-called Wolfe dual

$$L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \alpha_i \alpha_k y_i y_k x_i^T x_k \quad \text{subject to } \alpha_i \geq 0. \quad (4.52)$$

The solution is obtained by maximizing L_D in the positive orthant, a simpler convex optimization problem, for which standard software can be used. In addition the solution must satisfy the Karush–Kuhn–Tucker conditions, which include (4.50), (4.51), (4.52) and

$$\alpha_i [y_i(x_i^T \beta + \beta_0) - 1] = 0 \quad \forall i. \quad (4.53)$$

From these we can see that

- if $\alpha_i > 0$, then $y_i(x_i^T \beta + \beta_0) = 1$, or in other words, x_i is on the boundary of the slab;
- if $y_i(x_i^T \beta + \beta_0) > 1$, x_i is not on the boundary of the slab, and $\alpha_i = 0$.

From (4.50) we see that the solution vector β is defined in terms of a linear combination of the *support points* x_i —those points defined to be on the boundary of the slab via $\alpha_i > 0$. Figure 4.16 shows the optimal separating hyperplane for our toy example; there are three support points. Likewise, β_0 is obtained by solving (4.53) for any of the support points.

The optimal separating hyperplane produces a function $\hat{f}(x) = x^T \hat{\beta} + \hat{\beta}_0$ for classifying new observations:

$$\hat{G}(x) = \text{sign}(\hat{f}(x)). \quad (4.54)$$

Although none of the training observations fall in the margin (by construction), this will not necessarily be the case for test observations. The

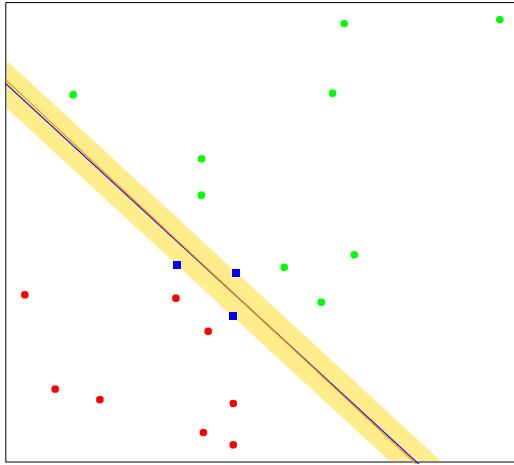


FIGURE 4.16. The same data as in Figure 4.14. The shaded region delineates the maximum margin separating the two classes. There are three support points indicated, which lie on the boundary of the margin, and the optimal separating hyperplane (blue line) bisects the slab. Included in the figure is the boundary found using logistic regression (red line), which is very close to the optimal separating hyperplane (see Section 12.3.3).

intuition is that a large margin on the training data will lead to good separation on the test data.

The description of the solution in terms of support points seems to suggest that the optimal hyperplane focuses more on the points that count, and is more robust to model misspecification. The LDA solution, on the other hand, depends on all of the data, even points far away from the decision boundary. Note, however, that the identification of these support points required the use of all the data. Of course, if the classes are really Gaussian, then LDA is optimal, and separating hyperplanes will pay a price for focusing on the (noisier) data at the boundaries of the classes.

Included in Figure 4.16 is the logistic regression solution to this problem, fit by maximum likelihood. Both solutions are similar in this case. When a separating hyperplane exists, logistic regression will always find it, since the log-likelihood can be driven to 0 in this case (Exercise 4.5). The logistic regression solution shares some other qualitative features with the separating hyperplane solution. The coefficient vector is defined by a weighted least squares fit of a zero-mean linearized response on the input features, and the weights are larger for points near the decision boundary than for those further away.

When the data are not separable, there will be no feasible solution to this problem, and an alternative formulation is needed. Again one can enlarge the space using basis transformations, but this can lead to artificial

separation through over-fitting. In Chapter 12 we discuss a more attractive alternative known as the *support vector machine*, which allows for overlap, but minimizes a measure of the extent of this overlap.

Bibliographic Notes

Good general texts on classification include Duda et al. (2000), Hand (1981), McLachlan (1992) and Ripley (1996). Mardia et al. (1979) have a concise discussion of linear discriminant analysis. Michie et al. (1994) compare a large number of popular classifiers on benchmark datasets. Linear separating hyperplanes are discussed in Vapnik (1996). Our account of the perceptron learning algorithm follows Ripley (1996).

Exercises

Ex. 4.1 Show how to solve the generalized eigenvalue problem $\max a^T \mathbf{B}a$ subject to $a^T \mathbf{W}a = 1$ by transforming to a standard eigenvalue problem.

Ex. 4.2 Suppose we have features $x \in \mathbb{R}^p$, a two-class response, with class sizes N_1, N_2 , and the target coded as $-N/N_1, N/N_2$.

(a) Show that the LDA rule classifies to class 2 if

$$x^T \hat{\Sigma}^{-1} (\hat{\mu}_2 - \hat{\mu}_1) > \frac{1}{2} \hat{\mu}_2^T \hat{\Sigma}^{-1} \hat{\mu}_2 - \frac{1}{2} \hat{\mu}_1^T \hat{\Sigma}^{-1} \hat{\mu}_1 + \log\left(\frac{N_1}{N}\right) - \log\left(\frac{N_2}{N}\right),$$

and class 1 otherwise.

(b) Consider minimization of the least squares criterion

$$\sum_{i=1}^N (y_i - \beta_0 - \beta^T x_i)^2. \quad (4.55)$$

Show that the solution $\hat{\beta}$ satisfies

$$\left[(N-2)\hat{\Sigma} + \frac{N_1 N_2}{N} \hat{\Sigma}_B \right] \beta = N(\hat{\mu}_2 - \hat{\mu}_1) \quad (4.56)$$

(after simplification), where $\hat{\Sigma}_B = (\hat{\mu}_2 - \hat{\mu}_1)(\hat{\mu}_2 - \hat{\mu}_1)^T$.

(c) Hence show that $\hat{\Sigma}_B \beta$ is in the direction $(\hat{\mu}_2 - \hat{\mu}_1)$ and thus

$$\hat{\beta} \propto \hat{\Sigma}^{-1} (\hat{\mu}_2 - \hat{\mu}_1). \quad (4.57)$$

Therefore the least squares regression coefficient is identical to the LDA coefficient, up to a scalar multiple.

6

Linear Model Selection and Regularization

In the regression setting, the standard linear model

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon \quad (6.1)$$

is commonly used to describe the relationship between a response Y and a set of variables X_1, X_2, \dots, X_p . We have seen in Chapter 3 that one typically fits this model using least squares.

In the chapters that follow, we consider some approaches for extending the linear model framework. In Chapter 7 we generalize (6.1) in order to accommodate non-linear, but still additive, relationships, while in Chapter 8 we consider even more general non-linear models. However, the linear model has distinct advantages in terms of inference and, on real-world problems, is often surprisingly competitive in relation to non-linear methods. Hence, before moving to the non-linear world, we discuss in this chapter some ways in which the simple linear model can be improved, by replacing plain least squares fitting with some alternative fitting procedures.

Why might we want to use another fitting procedure instead of least squares? As we will see, alternative fitting procedures can yield better *prediction accuracy* and *model interpretability*.

- *Prediction Accuracy:* Provided that the true relationship between the response and the predictors is approximately linear, the least squares estimates will have low bias. If $n \gg p$ —that is, if n , the number of observations, is much larger than p , the number of variables—then the least squares estimates tend to also have low variance, and hence will perform well on test observations. However, if n is not much larger

than p , then there can be a lot of variability in the least squares fit, resulting in overfitting and consequently poor predictions on future observations not used in model training. And if $p > n$, then there is no longer a unique least squares coefficient estimate: the variance is *infinite* so the method cannot be used at all. By *constraining* or *shrinking* the estimated coefficients, we can often substantially reduce the variance at the cost of a negligible increase in bias. This can lead to substantial improvements in the accuracy with which we can predict the response for observations not used in model training.

- *Model Interpretability:* It is often the case that some or many of the variables used in a multiple regression model are in fact not associated with the response. Including such *irrelevant* variables leads to unnecessary complexity in the resulting model. By removing these variables—that is, by setting the corresponding coefficient estimates to zero—we can obtain a model that is more easily interpreted. Now least squares is extremely unlikely to yield any coefficient estimates that are exactly zero. In this chapter, we see some approaches for automatically performing *feature selection* or *variable selection*—that is, for excluding irrelevant variables from a multiple regression model.

There are many alternatives, both classical and modern, to using least squares to fit (6.1). In this chapter, we discuss three important classes of methods.

feature
selection
variable
selection

- *Subset Selection.* This approach involves identifying a subset of the p predictors that we believe to be related to the response. We then fit a model using least squares on the reduced set of variables.
- *Shrinkage.* This approach involves fitting a model involving all p predictors. However, the estimated coefficients are shrunk towards zero relative to the least squares estimates. This shrinkage (also known as *regularization*) has the effect of reducing variance. Depending on what type of shrinkage is performed, some of the coefficients may be estimated to be exactly zero. Hence, shrinkage methods can also perform variable selection.
- *Dimension Reduction.* This approach involves *projecting* the p predictors into a M -dimensional subspace, where $M < p$. This is achieved by computing M different *linear combinations*, or *projections*, of the variables. Then these M projections are used as predictors to fit a linear regression model by least squares.

In the following sections we describe each of these approaches in greater detail, along with their advantages and disadvantages. Although this chapter describes extensions and modifications to the linear model for regression seen in Chapter 3, the same concepts apply to other methods, such as the classification models seen in Chapter 4.

6.1 Subset Selection

In this section we consider some methods for selecting subsets of predictors. These include best subset and stepwise model selection procedures.

6.1.1 Best Subset Selection

To perform *best subset selection*, we fit a separate least squares regression for each possible combination of the p predictors. That is, we fit all p models that contain exactly one predictor, all $\binom{p}{2} = p(p-1)/2$ models that contain exactly two predictors, and so forth. We then look at all of the resulting models, with the goal of identifying the one that is *best*.

best subset selection

The problem of selecting the *best model* from among the 2^p possibilities considered by best subset selection is not trivial. This is usually broken up into two stages, as described in Algorithm 6.1.

Algorithm 6.1 Best subset selection

1. Let \mathcal{M}_0 denote the *null model*, which contains no predictors. This model simply predicts the sample mean for each observation.
 2. For $k = 1, 2, \dots, p$:
 - (a) Fit all $\binom{p}{k}$ models that contain exactly k predictors.
 - (b) Pick the best among these $\binom{p}{k}$ models, and call it \mathcal{M}_k . Here *best* is defined as having the smallest RSS, or equivalently largest R^2 .
 3. Select a single best model from among $\mathcal{M}_0, \dots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2 .
-

In Algorithm 6.1, Step 2 identifies the best model (on the training data) for each subset size, in order to reduce the problem from one of 2^p possible models to one of $p + 1$ possible models. In Figure 6.1, these models form the lower frontier depicted in red.

Now in order to select a single best model, we must simply choose among these $p + 1$ options. This task must be performed with care, because the RSS of these $p + 1$ models decreases monotonically, and the R^2 increases monotonically, as the number of features included in the models increases. Therefore, if we use these statistics to select the best model, then we will always end up with a model involving all of the variables. The problem is that a low RSS or a high R^2 indicates a model with a low *training* error, whereas we wish to choose a model that has a low *test* error. (As shown in Chapter 2 in Figures 2.9–2.11, training error tends to be quite a bit smaller than test error, and a low training error by no means guarantees a low test error.) Therefore, in Step 3, we use cross-validated prediction

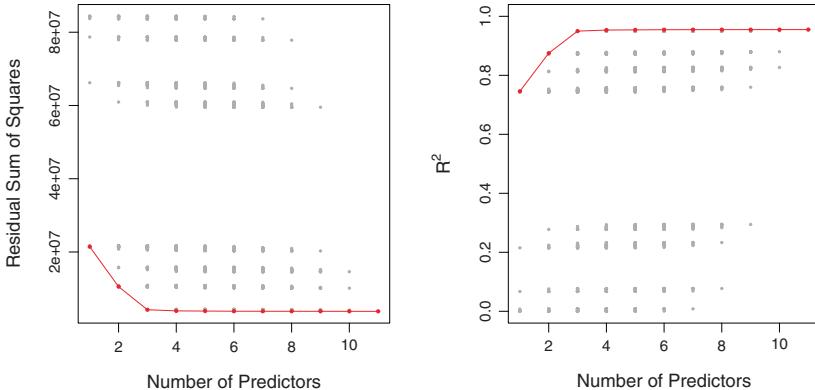


FIGURE 6.1. For each possible model containing a subset of the ten predictors in the `Credit` data set, the RSS and R^2 are displayed. The red frontier tracks the best model for a given number of predictors, according to RSS and R^2 . Though the data set contains only ten predictors, the x-axis ranges from 1 to 11, since one of the variables is categorical and takes on three values, leading to the creation of two dummy variables.

error, C_p , BIC, or adjusted R^2 in order to select among $\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_p$. These approaches are discussed in Section 6.1.3.

An application of best subset selection is shown in Figure 6.1. Each plotted point corresponds to a least squares regression model fit using a different subset of the 11 predictors in the `Credit` data set, discussed in Chapter 3. Here the variable `ethnicity` is a three-level qualitative variable, and so is represented by two dummy variables, which are selected separately in this case. We have plotted the RSS and R^2 statistics for each model, as a function of the number of variables. The red curves connect the best models for each model size, according to RSS or R^2 . The figure shows that, as expected, these quantities improve as the number of variables increases; however, from the three-variable model on, there is little improvement in RSS and R^2 as a result of including additional predictors.

Although we have presented best subset selection here for least squares regression, the same ideas apply to other types of models, such as logistic regression. In the case of logistic regression, instead of ordering models by RSS in Step 2 of Algorithm 6.1, we instead use the *deviance*, a measure that plays the role of RSS for a broader class of models. The deviance is negative two times the maximized log-likelihood; the smaller the deviance, the better the fit.

While best subset selection is a simple and conceptually appealing approach, it suffers from computational limitations. The number of possible models that must be considered grows rapidly as p increases. In general, there are 2^p models that involve subsets of p predictors. So if $p = 10$, then there are approximately 1,000 possible models to be considered, and if

$p = 20$, then there are over one million possibilities! Consequently, best subset selection becomes computationally infeasible for values of p greater than around 40, even with extremely fast modern computers. There are computational shortcuts—so called branch-and-bound techniques—for eliminating some choices, but these have their limitations as p gets large. They also only work for least squares linear regression. We present computationally efficient alternatives to best subset selection next.

6.1.2 Stepwise Selection

For computational reasons, best subset selection cannot be applied with very large p . Best subset selection may also suffer from statistical problems when p is large. The larger the search space, the higher the chance of finding models that look good on the training data, even though they might not have any predictive power on future data. Thus an enormous search space can lead to overfitting and high variance of the coefficient estimates.

For both of these reasons, *stepwise* methods, which explore a far more restricted set of models, are attractive alternatives to best subset selection.

Forward Stepwise Selection

Forward stepwise selection is a computationally efficient alternative to best subset selection. While the best subset selection procedure considers all 2^p possible models containing subsets of the p predictors, forward stepwise considers a much smaller set of models. Forward stepwise selection begins with a model containing no predictors, and then adds predictors to the model, one-at-a-time, until all of the predictors are in the model. In particular, at each step the variable that gives the greatest *additional* improvement to the fit is added to the model. More formally, the forward stepwise selection procedure is given in Algorithm 6.2.

forward
stepwise
selection

Algorithm 6.2 Forward stepwise selection

1. Let \mathcal{M}_0 denote the *null* model, which contains no predictors.
 2. For $k = 0, \dots, p - 1$:
 - (a) Consider all $p - k$ models that augment the predictors in \mathcal{M}_k with one additional predictor.
 - (b) Choose the *best* among these $p - k$ models, and call it \mathcal{M}_{k+1} . Here *best* is defined as having smallest RSS or highest R^2 .
 3. Select a single best model from among $\mathcal{M}_0, \dots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2 .
-

Unlike best subset selection, which involved fitting 2^p models, forward stepwise selection involves fitting one null model, along with $p - k$ models in the k th iteration, for $k = 0, \dots, p - 1$. This amounts to a total of $1 + \sum_{k=0}^{p-1} (p - k) = 1 + p(p+1)/2$ models. This is a substantial difference: when $p = 20$, best subset selection requires fitting 1,048,576 models, whereas forward stepwise selection requires fitting only 211 models.¹

In Step 2(b) of Algorithm 6.2, we must identify the *best* model from among those $p - k$ that augment \mathcal{M}_k with one additional predictor. We can do this by simply choosing the model with the lowest RSS or the highest R^2 . However, in Step 3, we must identify the best model among a set of models with different numbers of variables. This is more challenging, and is discussed in Section 6.1.3.

Forward stepwise selection's computational advantage over best subset selection is clear. Though forward stepwise tends to do well in practice, it is not guaranteed to find the best possible model out of all 2^p models containing subsets of the p predictors. For instance, suppose that in a given data set with $p = 3$ predictors, the best possible one-variable model contains X_1 , and the best possible two-variable model instead contains X_2 and X_3 . Then forward stepwise selection will fail to select the best possible two-variable model, because \mathcal{M}_1 will contain X_1 , so \mathcal{M}_2 must also contain X_1 together with one additional variable.

Table 6.1, which shows the first four selected models for best subset and forward stepwise selection on the `Credit` data set, illustrates this phenomenon. Both best subset selection and forward stepwise selection choose `rating` for the best one-variable model and then include `income` and `student` for the two- and three-variable models. However, best subset selection replaces `rating` by `cards` in the four-variable model, while forward stepwise selection must maintain `rating` in its four-variable model. In this example, Figure 6.1 indicates that there is not much difference between the three- and four-variable models in terms of RSS, so either of the four-variable models will likely be adequate.

Forward stepwise selection can be applied even in the high-dimensional setting where $n < p$; however, in this case, it is possible to construct submodels $\mathcal{M}_0, \dots, \mathcal{M}_{n-1}$ only, since each submodel is fit using least squares, which will not yield a unique solution if $p \geq n$.

Backward Stepwise Selection

Like forward stepwise selection, *backward stepwise selection* provides an efficient alternative to best subset selection. However, unlike forward

backward
stepwise
selection

¹Though forward stepwise selection considers $p(p + 1)/2 + 1$ models, it performs a *guided* search over model space, and so the *effective* model space considered contains substantially more than $p(p + 1)/2 + 1$ models.

# Variables	Best subset	Forward stepwise
One	<code>rating</code>	<code>rating</code>
Two	<code>rating, income</code>	<code>rating, income</code>
Three	<code>rating, income, student</code>	<code>rating, income, student</code>
Four	<code>cards, income, student, limit</code>	<code>rating, income, student, limit</code>

TABLE 6.1. The first four selected models for best subset selection and forward stepwise selection on the `Credit` data set. The first three models are identical but the fourth models differ.

stepwise selection, it begins with the full least squares model containing all p predictors, and then iteratively removes the least useful predictor, one-at-a-time. Details are given in Algorithm 6.3.

Algorithm 6.3 Backward stepwise selection

1. Let \mathcal{M}_p denote the *full* model, which contains all p predictors.
 2. For $k = p, p - 1, \dots, 1$:
 - (a) Consider all k models that contain all but one of the predictors in \mathcal{M}_k , for a total of $k - 1$ predictors.
 - (b) Choose the *best* among these k models, and call it \mathcal{M}_{k-1} . Here *best* is defined as having smallest RSS or highest R^2 .
 3. Select a single best model from among $\mathcal{M}_0, \dots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2 .
-

Like forward stepwise selection, the backward selection approach searches through only $1 + p(p+1)/2$ models, and so can be applied in settings where p is too large to apply best subset selection.² Also like forward stepwise selection, backward stepwise selection is not guaranteed to yield the *best* model containing a subset of the p predictors.

Backward selection requires that the number of samples n is larger than the number of variables p (so that the full model can be fit). In contrast, forward stepwise can be used even when $n < p$, and so is the only viable subset method when p is very large.

²Like forward stepwise selection, backward stepwise selection performs a *guided* search over model space, and so effectively considers substantially more than $1 + p(p+1)/2$ models.

Hybrid Approaches

The best subset, forward stepwise, and backward stepwise selection approaches generally give similar but not identical models. As another alternative, hybrid versions of forward and backward stepwise selection are available, in which variables are added to the model sequentially, in analogy to forward selection. However, after adding each new variable, the method may also remove any variables that no longer provide an improvement in the model fit. Such an approach attempts to more closely mimic best subset selection while retaining the computational advantages of forward and backward stepwise selection.

6.1.3 Choosing the Optimal Model

Best subset selection, forward selection, and backward selection result in the creation of a set of models, each of which contains a subset of the p predictors. In order to implement these methods, we need a way to determine which of these models is *best*. As we discussed in Section 6.1.1, the model containing all of the predictors will always have the smallest RSS and the largest R^2 , since these quantities are related to the training error. Instead, we wish to choose a model with a low test error. As is evident here, and as we show in Chapter 2, the training error can be a poor estimate of the test error. Therefore, RSS and R^2 are not suitable for selecting the best model among a collection of models with different numbers of predictors.

In order to select the best model with respect to test error, we need to estimate this test error. There are two common approaches:

1. We can indirectly estimate test error by making an *adjustment* to the training error to account for the bias due to overfitting.
2. We can *directly* estimate the test error, using either a validation set approach or a cross-validation approach, as discussed in Chapter 5.

We consider both of these approaches below.

C_p , AIC, BIC, and Adjusted R^2

We show in Chapter 2 that the training set MSE is generally an underestimate of the test MSE. (Recall that $\text{MSE} = \text{RSS}/n$.) This is because when we fit a model to the training data using least squares, we specifically estimate the regression coefficients such that the training RSS (but not the test RSS) is as small as possible. In particular, the training error will decrease as more variables are included in the model, but the test error may not. Therefore, training set RSS and training set R^2 cannot be used to select from among a set of models with different numbers of variables.

However, a number of techniques for *adjusting* the training error for the model size are available. These approaches can be used to select among a set

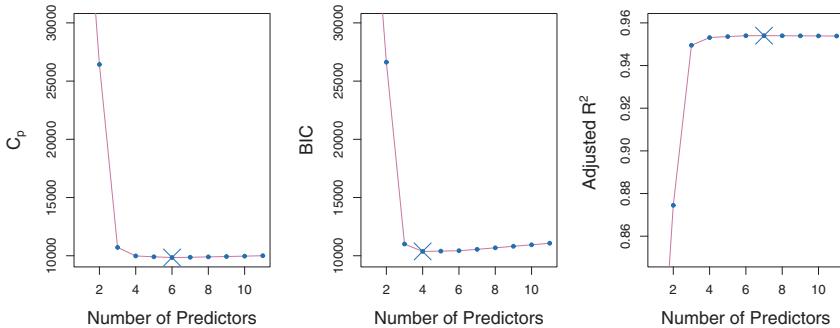


FIGURE 6.2. C_p , BIC, and adjusted R^2 are shown for the best models of each size for the **Credit** data set (the lower frontier in Figure 6.1). C_p and BIC are estimates of test MSE. In the middle plot we see that the BIC estimate of test error shows an increase after four variables are selected. The other two plots are rather flat after four variables are included.

of models with different numbers of variables. We now consider four such approaches: C_p , Akaike information criterion (AIC), Bayesian information criterion (BIC), and adjusted R^2 . Figure 6.2 displays C_p , BIC, and adjusted R^2 for the best model of each size produced by best subset selection on the **Credit** data set.

For a fitted least squares model containing d predictors, the C_p estimate of test MSE is computed using the equation

$$C_p = \frac{1}{n} (\text{RSS} + 2d\hat{\sigma}^2), \quad (6.2)$$

where $\hat{\sigma}^2$ is an estimate of the variance of the error ϵ associated with each response measurement in (6.1).³ Essentially, the C_p statistic adds a penalty of $2d\hat{\sigma}^2$ to the training RSS in order to adjust for the fact that the training error tends to underestimate the test error. Clearly, the penalty increases as the number of predictors in the model increases; this is intended to adjust for the corresponding decrease in training RSS. Though it is beyond the scope of this book, one can show that if $\hat{\sigma}^2$ is an unbiased estimate of σ^2 in (6.2), then C_p is an unbiased estimate of test MSE. As a consequence, the C_p statistic tends to take on a small value for models with a low test error, so when determining which of a set of models is best, we choose the model with the lowest C_p value. In Figure 6.2, C_p selects the six-variable model containing the predictors **income**, **limit**, **rating**, **cards**, **age** and **student**.

C_p
 Akaike
 information
 criterion
 Bayesian
 information
 criterion
 adjusted R^2

³Mallow's C_p is sometimes defined as $C'_p = \text{RSS}/\hat{\sigma}^2 + 2d - n$. This is equivalent to the definition given above in the sense that $C_p = \frac{1}{n}\hat{\sigma}^2(C'_p + n)$, and so the model with smallest C_p also has smallest C'_p .

The AIC criterion is defined for a large class of models fit by maximum likelihood. In the case of the model (6.1) with Gaussian errors, maximum likelihood and least squares are the same thing. In this case AIC is given by

$$\text{AIC} = \frac{1}{n\hat{\sigma}^2} (\text{RSS} + 2d\hat{\sigma}^2),$$

where, for simplicity, we have omitted an additive constant. Hence for least squares models, C_p and AIC are proportional to each other, and so only C_p is displayed in Figure 6.2.

BIC is derived from a Bayesian point of view, but ends up looking similar to C_p (and AIC) as well. For the least squares model with d predictors, the BIC is, up to irrelevant constants, given by

$$\text{BIC} = \frac{1}{n} (\text{RSS} + \log(n)d\hat{\sigma}^2). \quad (6.3)$$

Like C_p , the BIC will tend to take on a small value for a model with a low test error, and so generally we select the model that has the lowest BIC value. Notice that BIC replaces the $2d\hat{\sigma}^2$ used by C_p with a $\log(n)d\hat{\sigma}^2$ term, where n is the number of observations. Since $\log n > 2$ for any $n > 7$, the BIC statistic generally places a heavier penalty on models with many variables, and hence results in the selection of smaller models than C_p . In Figure 6.2, we see that this is indeed the case for the **Credit** data set; BIC chooses a model that contains only the four predictors `income`, `limit`, `cards`, and `student`. In this case the curves are very flat and so there does not appear to be much difference in accuracy between the four-variable and six-variable models.

The adjusted R^2 statistic is another popular approach for selecting among a set of models that contain different numbers of variables. Recall from Chapter 3 that the usual R^2 is defined as $1 - \text{RSS}/\text{TSS}$, where $\text{TSS} = \sum(y_i - \bar{y})^2$ is the *total sum of squares* for the response. Since RSS always decreases as more variables are added to the model, the R^2 always increases as more variables are added. For a least squares model with d variables, the adjusted R^2 statistic is calculated as

$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n - d - 1)}{\text{TSS}/(n - 1)}. \quad (6.4)$$

Unlike C_p , AIC, and BIC, for which a *small* value indicates a model with a low test error, a *large* value of adjusted R^2 indicates a model with a small test error. Maximizing the adjusted R^2 is equivalent to minimizing $\frac{\text{RSS}}{n-d-1}$. While RSS always decreases as the number of variables in the model increases, $\frac{\text{RSS}}{n-d-1}$ may increase or decrease, due to the presence of d in the denominator.

The intuition behind the adjusted R^2 is that once all of the correct variables have been included in the model, adding additional *noise* variables

will lead to only a very small decrease in RSS. Since adding noise variables leads to an increase in d , such variables will lead to an increase in $\frac{\text{RSS}}{n-d-1}$, and consequently a decrease in the adjusted R^2 . Therefore, in theory, the model with the largest adjusted R^2 will have only correct variables and no noise variables. Unlike the R^2 statistic, the adjusted R^2 statistic *pays a price* for the inclusion of unnecessary variables in the model. Figure 6.2 displays the adjusted R^2 for the **Credit** data set. Using this statistic results in the selection of a model that contains seven variables, adding **gender** to the model selected by C_p and AIC.

C_p , AIC, and BIC all have rigorous theoretical justifications that are beyond the scope of this book. These justifications rely on asymptotic arguments (scenarios where the sample size n is very large). Despite its popularity, and even though it is quite intuitive, the adjusted R^2 is not as well motivated in statistical theory as AIC, BIC, and C_p . All of these measures are simple to use and compute. Here we have presented the formulas for AIC, BIC, and C_p in the case of a linear model fit using least squares; however, these quantities can also be defined for more general types of models.

Validation and Cross-Validation

As an alternative to the approaches just discussed, we can directly estimate the test error using the validation set and cross-validation methods discussed in Chapter 5. We can compute the validation set error or the cross-validation error for each model under consideration, and then select the model for which the resulting estimated test error is smallest. This procedure has an advantage relative to AIC, BIC, C_p , and adjusted R^2 , in that it provides a direct estimate of the test error, and makes fewer assumptions about the true underlying model. It can also be used in a wider range of model selection tasks, even in cases where it is hard to pinpoint the model degrees of freedom (e.g. the number of predictors in the model) or hard to estimate the error variance σ^2 .

In the past, performing cross-validation was computationally prohibitive for many problems with large p and/or large n , and so AIC, BIC, C_p , and adjusted R^2 were more attractive approaches for choosing among a set of models. However, nowadays with fast computers, the computations required to perform cross-validation are hardly ever an issue. Thus, cross-validation is a very attractive approach for selecting from among a number of models under consideration.

Figure 6.3 displays, as a function of d , the BIC, validation set errors, and cross-validation errors on the **Credit** data, for the best d -variable model. The validation errors were calculated by randomly selecting three-quarters of the observations as the training set, and the remainder as the validation set. The cross-validation errors were computed using $k = 10$ folds. In this case, the validation and cross-validation methods both result in a

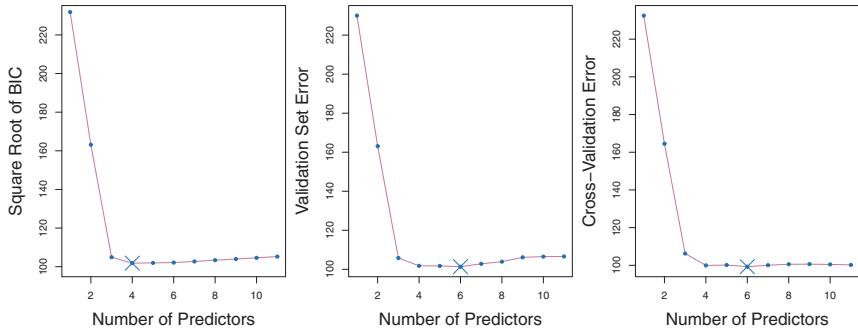


FIGURE 6.3. For the `Credit` data set, three quantities are displayed for the best model containing d predictors, for d ranging from 1 to 11. The overall best model, based on each of these quantities, is shown as a blue cross. Left: Square root of BIC. Center: Validation set errors. Right: Cross-validation errors.

six-variable model. However, all three approaches suggest that the four-, five-, and six-variable models are roughly equivalent in terms of their test errors.

In fact, the estimated test error curves displayed in the center and right-hand panels of Figure 6.3 are quite flat. While a three-variable model clearly has lower estimated test error than a two-variable model, the estimated test errors of the 3- to 11-variable models are quite similar. Furthermore, if we repeated the validation set approach using a different split of the data into a training set and a validation set, or if we repeated cross-validation using a different set of cross-validation folds, then the precise model with the lowest estimated test error would surely change. In this setting, we can select a model using the *one-standard-error rule*. We first calculate the standard error of the estimated test MSE for each model size, and then select the smallest model for which the estimated test error is within one standard error of the lowest point on the curve. The rationale here is that if a set of models appear to be more or less equally good, then we might as well choose the simplest model—that is, the model with the smallest number of predictors. In this case, applying the one-standard-error rule to the validation set or cross-validation approach leads to selection of the three-variable model.

one-standard-error rule

6.2 Shrinkage Methods

The subset selection methods described in Section 6.1 involve using least squares to fit a linear model that contains a subset of the predictors. As an alternative, we can fit a model containing all p predictors using a technique that *constraints* or *regularizes* the coefficient estimates, or equivalently, that *shrinks* the coefficient estimates towards zero. It may not be immediately

obvious why such a constraint should improve the fit, but it turns out that shrinking the coefficient estimates can significantly reduce their variance. The two best-known techniques for shrinking the regression coefficients towards zero are *ridge regression* and the *lasso*.

6.2.1 Ridge Regression

Recall from Chapter 3 that the least squares fitting procedure estimates $\beta_0, \beta_1, \dots, \beta_p$ using the values that minimize

$$\text{RSS} = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2.$$

Ridge regression is very similar to least squares, except that the coefficients are estimated by minimizing a slightly different quantity. In particular, the ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^p \beta_j^2, \quad (6.5)$$

where $\lambda \geq 0$ is a *tuning parameter*, to be determined separately. Equation 6.5 trades off two different criteria. As with least squares, ridge regression seeks coefficient estimates that fit the data well, by making the RSS small. However, the second term, $\lambda \sum_j \beta_j^2$, called a *shrinkage penalty*, is small when β_1, \dots, β_p are close to zero, and so it has the effect of *shrinking* the estimates of β_j towards zero. The tuning parameter λ serves to control the relative impact of these two terms on the regression coefficient estimates. When $\lambda = 0$, the penalty term has no effect, and ridge regression will produce the least squares estimates. However, as $\lambda \rightarrow \infty$, the impact of the shrinkage penalty grows, and the ridge regression coefficient estimates will approach zero. Unlike least squares, which generates only one set of coefficient estimates, ridge regression will produce a different set of coefficient estimates, $\hat{\beta}_\lambda^R$, for each value of λ . Selecting a good value for λ is critical; we defer this discussion to Section 6.2.3, where we use cross-validation.

Note that in (6.5), the shrinkage penalty is applied to β_1, \dots, β_p , but not to the intercept β_0 . We want to shrink the estimated association of each variable with the response; however, we do not want to shrink the intercept, which is simply a measure of the mean value of the response when $x_{i1} = x_{i2} = \dots = x_{ip} = 0$. If we assume that the variables—that is, the columns of the data matrix \mathbf{X} —have been centered to have mean zero before ridge regression is performed, then the estimated intercept will take the form $\hat{\beta}_0 = \bar{y} = \sum_{i=1}^n y_i/n$.

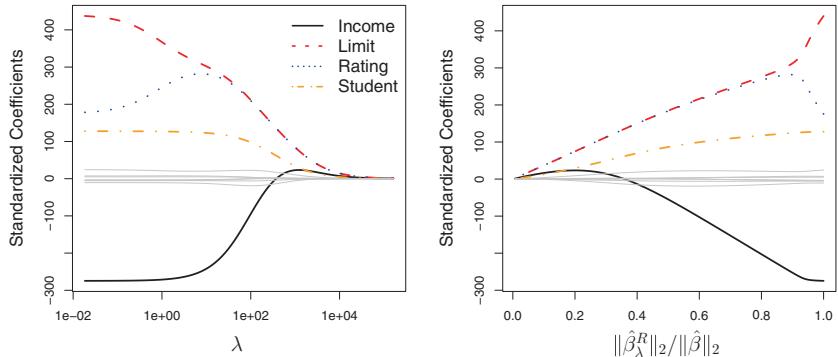


FIGURE 6.4. The standardized ridge regression coefficients are displayed for the `Credit` data set, as a function of λ and $\|\hat{\beta}_\lambda^R\|_2 / \|\hat{\beta}\|_2$.

An Application to the Credit Data

In Figure 6.4, the ridge regression coefficient estimates for the `Credit` data set are displayed. In the left-hand panel, each curve corresponds to the ridge regression coefficient estimate for one of the ten variables, plotted as a function of λ . For example, the black solid line represents the ridge regression estimate for the `income` coefficient, as λ is varied. At the extreme left-hand side of the plot, λ is essentially zero, and so the corresponding ridge coefficient estimates are the same as the usual least squares estimates. But as λ increases, the ridge coefficient estimates shrink towards zero. When λ is extremely large, then all of the ridge coefficient estimates are basically zero; this corresponds to the *null model* that contains no predictors. In this plot, the `income`, `limit`, `rating`, and `student` variables are displayed in distinct colors, since these variables tend to have by far the largest coefficient estimates. While the ridge coefficient estimates tend to decrease in aggregate as λ increases, individual coefficients, such as `rating` and `income`, may occasionally increase as λ increases.

The right-hand panel of Figure 6.4 displays the same ridge coefficient estimates as the left-hand panel, but instead of displaying λ on the x -axis, we now display $\|\hat{\beta}_\lambda^R\|_2 / \|\hat{\beta}\|_2$, where $\hat{\beta}$ denotes the vector of least squares coefficient estimates. The notation $\|\beta\|_2$ denotes the ℓ_2 norm (pronounced “ell 2”) of a vector, and is defined as $\|\beta\|_2 = \sqrt{\sum_{j=1}^p \beta_j^2}$. It measures the distance of β from zero. As λ increases, the ℓ_2 norm of $\hat{\beta}_\lambda^R$ will always decrease, and so will $\|\hat{\beta}_\lambda^R\|_2 / \|\hat{\beta}\|_2$. The latter quantity ranges from 1 (when $\lambda = 0$, in which case the ridge regression coefficient estimate is the same as the least squares estimate, and so their ℓ_2 norms are the same) to 0 (when $\lambda = \infty$, in which case the ridge regression coefficient estimate is a vector of zeros, with ℓ_2 norm equal to zero). Therefore, we can think of the x -axis in the right-hand panel of Figure 6.4 as the amount that the ridge

regression coefficient estimates have been shrunk towards zero; a small value indicates that they have been shrunk very close to zero.

The standard least squares coefficient estimates discussed in Chapter 3 are *scale equivariant*: multiplying X_j by a constant c simply leads to a scaling of the least squares coefficient estimates by a factor of $1/c$. In other words, regardless of how the j th predictor is scaled, $X_j \hat{\beta}_j$ will remain the same. In contrast, the ridge regression coefficient estimates can change *substantially* when multiplying a given predictor by a constant. For instance, consider the `income` variable, which is measured in dollars. One could reasonably have measured income in thousands of dollars, which would result in a reduction in the observed values of `income` by a factor of 1,000. Now due to the sum of squared coefficients term in the ridge regression formulation (6.5), such a change in scale will not simply cause the ridge regression coefficient estimate for `income` to change by a factor of 1,000. In other words, $X_j \hat{\beta}_{j,\lambda}^R$ will depend not only on the value of λ , but also on the scaling of the j th predictor. In fact, the value of $X_j \hat{\beta}_{j,\lambda}^R$ may even depend on the scaling of the *other* predictors! Therefore, it is best to apply ridge regression after *standardizing the predictors*, using the formula

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n} \sum_{i=1}^n (x_{ij} - \bar{x}_j)^2}}, \quad (6.6)$$

so that they are all on the same scale. In (6.6), the denominator is the estimated standard deviation of the j th predictor. Consequently, all of the standardized predictors will have a standard deviation of one. As a result the final fit will not depend on the scale on which the predictors are measured. In Figure 6.4, the y -axis displays the standardized ridge regression coefficient estimates—that is, the coefficient estimates that result from performing ridge regression using standardized predictors.

Why Does Ridge Regression Improve Over Least Squares?

Ridge regression's advantage over least squares is rooted in the *bias-variance trade-off*. As λ increases, the flexibility of the ridge regression fit decreases, leading to decreased variance but increased bias. This is illustrated in the left-hand panel of Figure 6.5, using a simulated data set containing $p = 45$ predictors and $n = 50$ observations. The green curve in the left-hand panel of Figure 6.5 displays the variance of the ridge regression predictions as a function of λ . At the least squares coefficient estimates, which correspond to ridge regression with $\lambda = 0$, the variance is high but there is no bias. But as λ increases, the shrinkage of the ridge coefficient estimates leads to a substantial reduction in the variance of the predictions, at the expense of a slight increase in bias. Recall that the test mean squared error (MSE), plotted in purple, is a function of the variance plus the squared bias. For values

scale
equivariant

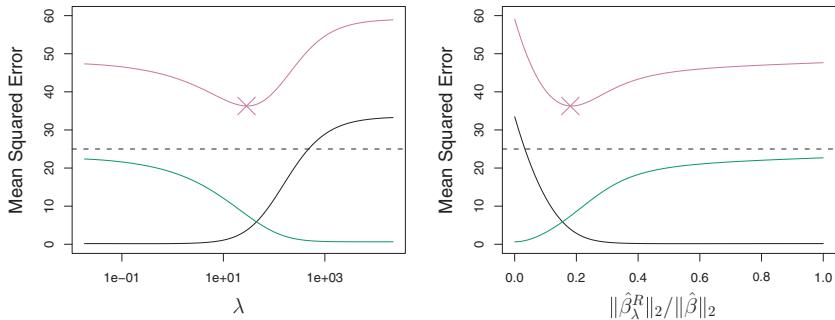


FIGURE 6.5. Squared bias (black), variance (green), and test mean squared error (purple) for the ridge regression predictions on a simulated data set, as a function of λ and $\|\hat{\beta}_\lambda^R\|_2 / \|\hat{\beta}\|_2$. The horizontal dashed lines indicate the minimum possible MSE. The purple crosses indicate the ridge regression models for which the MSE is smallest.

of λ up to about 10, the variance decreases rapidly, with very little increase in bias, plotted in black. Consequently, the MSE drops considerably as λ increases from 0 to 10. Beyond this point, the decrease in variance due to increasing λ slows, and the shrinkage on the coefficients causes them to be significantly underestimated, resulting in a large increase in the bias. The minimum MSE is achieved at approximately $\lambda = 30$. Interestingly, because of its high variance, the MSE associated with the least squares fit, when $\lambda = 0$, is almost as high as that of the null model for which all coefficient estimates are zero, when $\lambda = \infty$. However, for an intermediate value of λ , the MSE is considerably lower.

The right-hand panel of Figure 6.5 displays the same curves as the left-hand panel, this time plotted against the ℓ_2 norm of the ridge regression coefficient estimates divided by the ℓ_2 norm of the least squares estimates. Now as we move from left to right, the fits become more flexible, and so the bias decreases and the variance increases.

In general, in situations where the relationship between the response and the predictors is close to linear, the least squares estimates will have low bias but may have high variance. This means that a small change in the training data can cause a large change in the least squares coefficient estimates. In particular, when the number of variables p is almost as large as the number of observations n , as in the example in Figure 6.5, the least squares estimates will be extremely variable. And if $p > n$, then the least squares estimates do not even have a unique solution, whereas ridge regression can still perform well by trading off a small increase in bias for a large decrease in variance. Hence, ridge regression works best in situations where the least squares estimates have high variance.

Ridge regression also has substantial computational advantages over best subset selection, which requires searching through 2^p models. As we

discussed previously, even for moderate values of p , such a search can be computationally infeasible. In contrast, for any fixed value of λ , ridge regression only fits a single model, and the model-fitting procedure can be performed quite quickly. In fact, one can show that the computations required to solve (6.5), *simultaneously for all values of λ* , are almost identical to those for fitting a model using least squares.

6.2.2 The Lasso

Ridge regression does have one obvious disadvantage. Unlike best subset, forward stepwise, and backward stepwise selection, which will generally select models that involve just a subset of the variables, ridge regression will include all p predictors in the final model. The penalty $\lambda \sum \beta_j^2$ in (6.5) will shrink all of the coefficients towards zero, but it will not set any of them exactly to zero (unless $\lambda = \infty$). This may not be a problem for prediction accuracy, but it can create a challenge in model interpretation in settings in which the number of variables p is quite large. For example, in the `Credit` data set, it appears that the most important variables are `income`, `limit`, `rating`, and `student`. So we might wish to build a model including just these predictors. However, ridge regression will always generate a model involving all ten predictors. Increasing the value of λ will tend to reduce the magnitudes of the coefficients, but will not result in exclusion of any of the variables.

The *lasso* is a relatively recent alternative to ridge regression that overcomes this disadvantage. The lasso coefficients, $\hat{\beta}_\lambda^L$, minimize the quantity

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| = \text{RSS} + \lambda \sum_{j=1}^p |\beta_j|. \quad (6.7)$$

Comparing (6.7) to (6.5), we see that the lasso and ridge regression have similar formulations. The only difference is that the β_j^2 term in the ridge regression penalty (6.5) has been replaced by $|\beta_j|$ in the lasso penalty (6.7). In statistical parlance, the lasso uses an ℓ_1 (pronounced “ell 1”) penalty instead of an ℓ_2 penalty. The ℓ_1 norm of a coefficient vector β is given by $\|\beta\|_1 = \sum |\beta_j|$.

As with ridge regression, the lasso shrinks the coefficient estimates towards zero. However, in the case of the lasso, the ℓ_1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large. Hence, much like best subset selection, the lasso performs *variable selection*. As a result, models generated from the lasso are generally much easier to interpret than those produced by ridge regression. We say that the lasso yields *sparse* models—that is, models that involve only a subset of the variables. As in ridge regression, selecting a good value of λ for the lasso is critical; we defer this discussion to Section 6.2.3, where we use cross-validation.

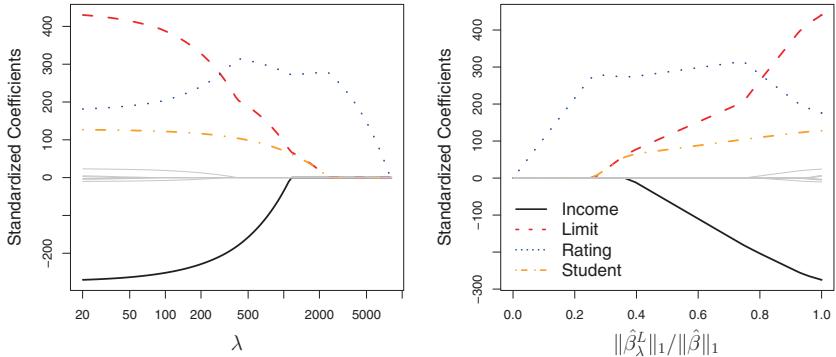


FIGURE 6.6. The standardized lasso coefficients on the `Credit` data set are shown as a function of λ and $\|\hat{\beta}_\lambda^L\|_1/\|\hat{\beta}\|_1$.

As an example, consider the coefficient plots in Figure 6.6, which are generated from applying the lasso to the `Credit` data set. When $\lambda = 0$, then the lasso simply gives the least squares fit, and when λ becomes sufficiently large, the lasso gives the null model in which all coefficient estimates equal zero. However, in between these two extremes, the ridge regression and lasso models are quite different from each other. Moving from left to right in the right-hand panel of Figure 6.6, we observe that at first the lasso results in a model that contains only the `rating` predictor. Then `student` and `limit` enter the model almost simultaneously, shortly followed by `income`. Eventually, the remaining variables enter the model. Hence, depending on the value of λ , the lasso can produce a model involving any number of variables. In contrast, ridge regression will always include all of the variables in the model, although the magnitude of the coefficient estimates will depend on λ .

Another Formulation for Ridge Regression and the Lasso

One can show that the lasso and ridge regression coefficient estimates solve the problems

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^p |\beta_j| \leq s \quad (6.8)$$

and

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^p \beta_j^2 \leq s, \quad (6.9)$$

respectively. In other words, for every value of λ , there is some s such that the Equations (6.7) and (6.8) will give the same lasso coefficient estimates. Similarly, for every value of λ there is a corresponding s such that Equations (6.5) and (6.9) will give the same ridge regression coefficient estimates. When $p = 2$, then (6.8) indicates that the lasso coefficient estimates have the smallest RSS out of all points that lie within the diamond defined by $|\beta_1| + |\beta_2| \leq s$. Similarly, the ridge regression estimates have the smallest RSS out of all points that lie within the circle defined by $\beta_1^2 + \beta_2^2 \leq s$.

We can think of (6.8) as follows. When we perform the lasso we are trying to find the set of coefficient estimates that lead to the smallest RSS, subject to the constraint that there is a *budget* s for how large $\sum_{j=1}^p |\beta_j|$ can be. When s is extremely large, then this budget is not very restrictive, and so the coefficient estimates can be large. In fact, if s is large enough that the least squares solution falls within the budget, then (6.8) will simply yield the least squares solution. In contrast, if s is small, then $\sum_{j=1}^p |\beta_j|$ must be small in order to avoid violating the budget. Similarly, (6.9) indicates that when we perform ridge regression, we seek a set of coefficient estimates such that the RSS is as small as possible, subject to the requirement that $\sum_{j=1}^p \beta_j^2$ not exceed the budget s .

The formulations (6.8) and (6.9) reveal a close connection between the lasso, ridge regression, and best subset selection. Consider the problem

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^p I(\beta_j \neq 0) \leq s. \quad (6.10)$$

Here $I(\beta_j \neq 0)$ is an indicator variable: it takes on a value of 1 if $\beta_j \neq 0$, and equals zero otherwise. Then (6.10) amounts to finding a set of coefficient estimates such that RSS is as small as possible, subject to the constraint that no more than s coefficients can be nonzero. The problem (6.10) is equivalent to best subset selection. Unfortunately, solving (6.10) is computationally infeasible when p is large, since it requires considering all $\binom{p}{s}$ models containing s predictors. Therefore, we can interpret ridge regression and the lasso as computationally feasible alternatives to best subset selection that replace the intractable form of the budget in (6.10) with forms that are much easier to solve. Of course, the lasso is much more closely related to best subset selection, since only the lasso performs feature selection for s sufficiently small in (6.8).

The Variable Selection Property of the Lasso

Why is it that the lasso, unlike ridge regression, results in coefficient estimates that are exactly equal to zero? The formulations (6.8) and (6.9) can be used to shed light on the issue. Figure 6.7 illustrates the situation. The least squares solution is marked as $\hat{\beta}$, while the blue diamond and

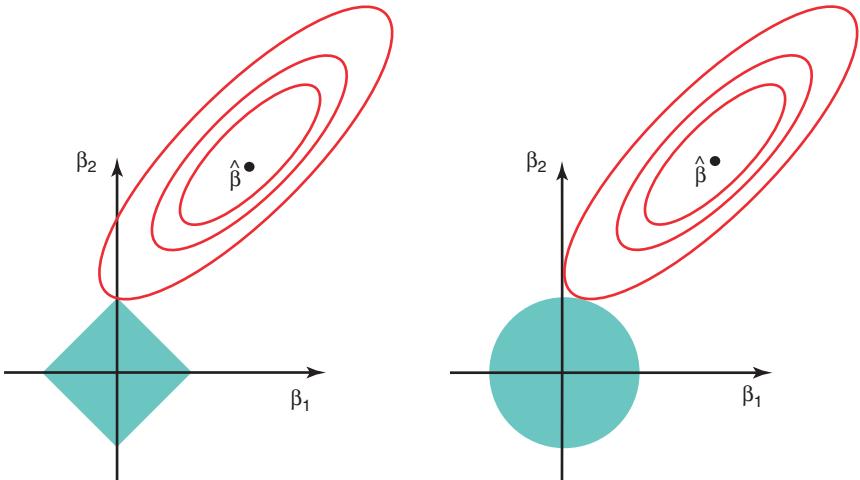


FIGURE 6.7. Contours of the error and constraint functions for the lasso (left) and ridge regression (right). The solid blue areas are the constraint regions, $|\beta_1| + |\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$, while the red ellipses are the contours of the RSS.

circle represent the lasso and ridge regression constraints in (6.8) and (6.9), respectively. If s is sufficiently large, then the constraint regions will contain $\hat{\beta}$, and so the ridge regression and lasso estimates will be the same as the least squares estimates. (Such a large value of s corresponds to $\lambda = 0$ in (6.5) and (6.7).) However, in Figure 6.7 the least squares estimates lie outside of the diamond and the circle, and so the least squares estimates are not the same as the lasso and ridge regression estimates.

The ellipses that are centered around $\hat{\beta}$ represent regions of constant RSS. In other words, all of the points on a given ellipse share a common value of the RSS. As the ellipses expand away from the least squares coefficient estimates, the RSS increases. Equations (6.8) and (6.9) indicate that the lasso and ridge regression coefficient estimates are given by the first point at which an ellipse contacts the constraint region. Since ridge regression has a circular constraint with no sharp points, this intersection will not generally occur on an axis, and so the ridge regression coefficient estimates will be exclusively non-zero. However, the lasso constraint has corners at each of the axes, and so the ellipse will often intersect the constraint region at an axis. When this occurs, one of the coefficients will equal zero. In higher dimensions, many of the coefficient estimates may equal zero simultaneously. In Figure 6.7, the intersection occurs at $\beta_1 = 0$, and so the resulting model will only include β_2 .

In Figure 6.7, we considered the simple case of $p = 2$. When $p = 3$, then the constraint region for ridge regression becomes a sphere, and the constraint region for the lasso becomes a polyhedron. When $p > 3$, the

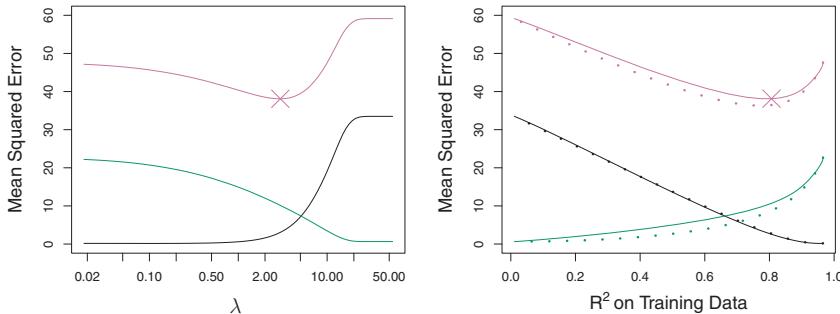


FIGURE 6.8. Left: Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso on a simulated data set. Right: Comparison of squared bias, variance and test MSE between lasso (solid) and ridge (dotted). Both are plotted against their R^2 on the training data, as a common form of indexing. The crosses in both plots indicate the lasso model for which the MSE is smallest.

constraint for ridge regression becomes a hypersphere, and the constraint for the lasso becomes a polytope. However, the key ideas depicted in Figure 6.7 still hold. In particular, the lasso leads to feature selection when $p > 2$ due to the sharp corners of the polyhedron or polytope.

Comparing the Lasso and Ridge Regression

It is clear that the lasso has a major advantage over ridge regression, in that it produces simpler and more interpretable models that involve only a subset of the predictors. However, which method leads to better prediction accuracy? Figure 6.8 displays the variance, squared bias, and test MSE of the lasso applied to the same simulated data as in Figure 6.5. Clearly the lasso leads to qualitatively similar behavior to ridge regression, in that as λ increases, the variance decreases and the bias increases. In the right-hand panel of Figure 6.8, the dotted lines represent the ridge regression fits. Here we plot both against their R^2 on the training data. This is another useful way to index models, and can be used to compare models with different types of regularization, as is the case here. In this example, the lasso and ridge regression result in almost identical biases. However, the variance of ridge regression is slightly lower than the variance of the lasso. Consequently, the minimum MSE of ridge regression is slightly smaller than that of the lasso.

However, the data in Figure 6.8 were generated in such a way that all 45 predictors were related to the response—that is, none of the true coefficients $\beta_1, \dots, \beta_{45}$ equaled zero. The lasso implicitly assumes that a number of the coefficients truly equal zero. Consequently, it is not surprising that ridge regression outperforms the lasso in terms of prediction error in this setting. Figure 6.9 illustrates a similar situation, except that now the response is a

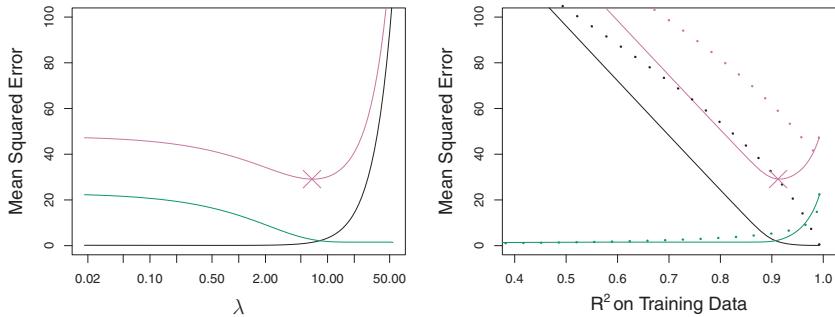


FIGURE 6.9. Left: Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso. The simulated data is similar to that in Figure 6.8, except that now only two predictors are related to the response. Right: Comparison of squared bias, variance and test MSE between lasso (solid) and ridge (dotted). Both are plotted against their R^2 on the training data, as a common form of indexing. The crosses in both plots indicate the lasso model for which the MSE is smallest.

function of only 2 out of 45 predictors. Now the lasso tends to outperform ridge regression in terms of bias, variance, and MSE.

These two examples illustrate that neither ridge regression nor the lasso will universally dominate the other. In general, one might expect the lasso to perform better in a setting where a relatively small number of predictors have substantial coefficients, and the remaining predictors have coefficients that are very small or that equal zero. Ridge regression will perform better when the response is a function of many predictors, all with coefficients of roughly equal size. However, the number of predictors that is related to the response is never known *a priori* for real data sets. A technique such as cross-validation can be used in order to determine which approach is better on a particular data set.

As with ridge regression, when the least squares estimates have excessively high variance, the lasso solution can yield a reduction in variance at the expense of a small increase in bias, and consequently can generate more accurate predictions. Unlike ridge regression, the lasso performs variable selection, and hence results in models that are easier to interpret.

There are very efficient algorithms for fitting both ridge and lasso models; in both cases the entire coefficient paths can be computed with about the same amount of work as a single least squares fit. We will explore this further in the lab at the end of this chapter.

A Simple Special Case for Ridge Regression and the Lasso

In order to obtain a better intuition about the behavior of ridge regression and the lasso, consider a simple special case with $n = p$, and \mathbf{X} a diagonal matrix with 1's on the diagonal and 0's in all off-diagonal elements. To simplify the problem further, assume also that we are performing regres-

sion without an intercept. With these assumptions, the usual least squares problem simplifies to finding β_1, \dots, β_p that minimize

$$\sum_{j=1}^p (y_j - \beta_j)^2. \quad (6.11)$$

In this case, the least squares solution is given by

$$\hat{\beta}_j = y_j.$$

And in this setting, ridge regression amounts to finding β_1, \dots, β_p such that

$$\sum_{j=1}^p (y_j - \beta_j)^2 + \lambda \sum_{j=1}^p \beta_j^2 \quad (6.12)$$

is minimized, and the lasso amounts to finding the coefficients such that

$$\sum_{j=1}^p (y_j - \beta_j)^2 + \lambda \sum_{j=1}^p |\beta_j| \quad (6.13)$$

is minimized. One can show that in this setting, the ridge regression estimates take the form

$$\hat{\beta}_j^R = y_j / (1 + \lambda), \quad (6.14)$$

and the lasso estimates take the form

$$\hat{\beta}_j^L = \begin{cases} y_j - \lambda/2 & \text{if } y_j > \lambda/2; \\ y_j + \lambda/2 & \text{if } y_j < -\lambda/2; \\ 0 & \text{if } |y_j| \leq \lambda/2. \end{cases} \quad (6.15)$$

Figure 6.10 displays the situation. We can see that ridge regression and the lasso perform two very different types of shrinkage. In ridge regression, each least squares coefficient estimate is shrunk by the same proportion. In contrast, the lasso shrinks each least squares coefficient towards zero by a constant amount, $\lambda/2$; the least squares coefficients that are less than $\lambda/2$ in absolute value are shrunk entirely to zero. The type of shrinkage performed by the lasso in this simple setting (6.15) is known as *soft-thresholding*. The fact that some lasso coefficients are shrunk entirely to zero explains why the lasso performs feature selection.

In the case of a more general data matrix \mathbf{X} , the story is a little more complicated than what is depicted in Figure 6.10, but the main ideas still hold approximately: ridge regression more or less shrinks every dimension of the data by the same proportion, whereas the lasso more or less shrinks all coefficients toward zero by a similar amount, and sufficiently small coefficients are shrunk all the way to zero.

soft-
thresholding

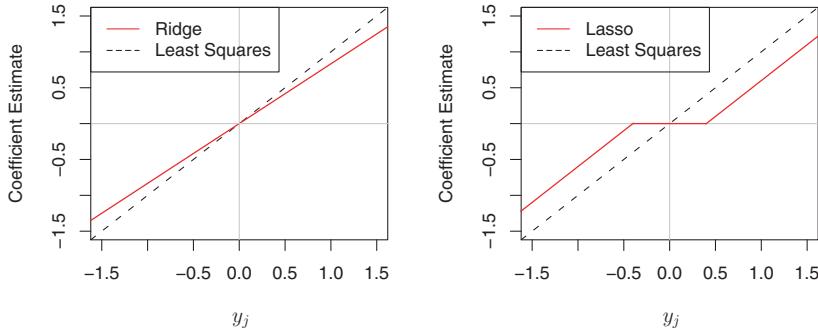


FIGURE 6.10. The ridge regression and lasso coefficient estimates for a simple setting with $n = p$ and \mathbf{X} a diagonal matrix with 1's on the diagonal. Left: The ridge regression coefficient estimates are shrunken proportionally towards zero, relative to the least squares estimates. Right: The lasso coefficient estimates are soft-thresholded towards zero.

Bayesian Interpretation for Ridge Regression and the Lasso



We now show that one can view ridge regression and the lasso through a Bayesian lens. A Bayesian viewpoint for regression assumes that the coefficient vector β has some *prior* distribution, say $p(\beta)$, where $\beta = (\beta_0, \beta_1, \dots, \beta_p)^T$. The likelihood of the data can be written as $f(Y|X, \beta)$, where $X = (X_1, \dots, X_p)$. Multiplying the prior distribution by the likelihood gives us (up to a proportionality constant) the *posterior distribution*, which takes the form

posterior distribution

$$p(\beta|X, Y) \propto f(Y|X, \beta)p(\beta|X) = f(Y|X, \beta)p(\beta),$$

where the proportionality above follows from Bayes' theorem, and the equality above follows from the assumption that X is fixed.

We assume the usual linear model,

$$Y = \beta_0 + X_1\beta_1 + \dots + X_p\beta_p + \epsilon,$$

and suppose that the errors are independent and drawn from a normal distribution. Furthermore, assume that $p(\beta) = \prod_{j=1}^p g(\beta_j)$, for some density function g . It turns out that ridge regression and the lasso follow naturally from two special cases of g :

- If g is a Gaussian distribution with mean zero and standard deviation a function of λ , then it follows that the *posterior mode* for β —that is, the most likely value for β , given the data—is given by the ridge regression solution. (In fact, the ridge regression solution is also the posterior mean.)

posterior mode

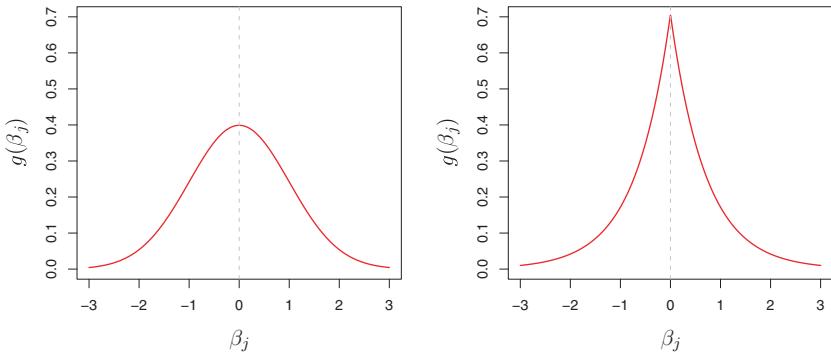


FIGURE 6.11. Left: Ridge regression is the posterior mode for β under a Gaussian prior. Right: The lasso is the posterior mode for β under a double-exponential prior.

- If g is a double-exponential (Laplace) distribution with mean zero and scale parameter a function of λ , then it follows that the posterior mode for β is the lasso solution. (However, the lasso solution is *not* the posterior mean, and in fact, the posterior mean does not yield a sparse coefficient vector.)

The Gaussian and double-exponential priors are displayed in Figure 6.11. Therefore, from a Bayesian viewpoint, ridge regression and the lasso follow directly from assuming the usual linear model with normal errors, together with a simple prior distribution for β . Notice that the lasso prior is steeply peaked at zero, while the Gaussian is flatter and fatter at zero. Hence, the lasso expects a priori that many of the coefficients are (exactly) zero, while ridge assumes the coefficients are randomly distributed about zero.

6.2.3 Selecting the Tuning Parameter

Just as the subset selection approaches considered in Section 6.1 require a method to determine which of the models under consideration is best, implementing ridge regression and the lasso requires a method for selecting a value for the tuning parameter λ in (6.5) and (6.7), or equivalently, the value of the constraint s in (6.9) and (6.8). Cross-validation provides a simple way to tackle this problem. We choose a grid of λ values, and compute the cross-validation error for each value of λ , as described in Chapter 5. We then select the tuning parameter value for which the cross-validation error is smallest. Finally, the model is re-fit using all of the available observations and the selected value of the tuning parameter.

Figure 6.12 displays the choice of λ that results from performing leave-one-out cross-validation on the ridge regression fits from the **Credit** data set. The dashed vertical lines indicate the selected value of λ . In this case the value is relatively small, indicating that the optimal fit only involves a

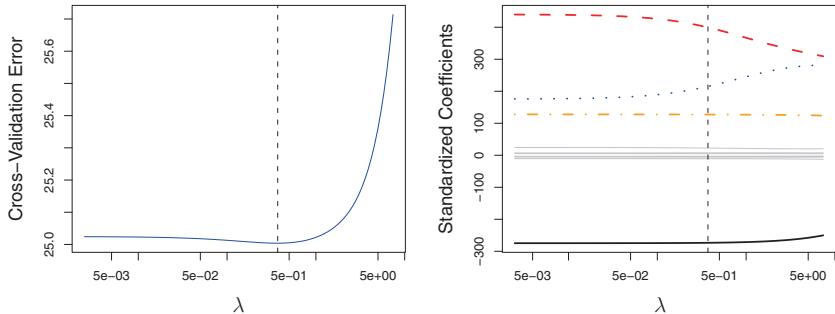


FIGURE 6.12. Left: Cross-validation errors that result from applying ridge regression to the `Credit` data set with various value of λ . Right: The coefficient estimates as a function of λ . The vertical dashed lines indicate the value of λ selected by cross-validation.

small amount of shrinkage relative to the least squares solution. In addition, the dip is not very pronounced, so there is rather a wide range of values that would give very similar error. In a case like this we might simply use the least squares solution.

Figure 6.13 provides an illustration of ten-fold cross-validation applied to the lasso fits on the sparse simulated data from Figure 6.9. The left-hand panel of Figure 6.13 displays the cross-validation error, while the right-hand panel displays the coefficient estimates. The vertical dashed lines indicate the point at which the cross-validation error is smallest. The two colored lines in the right-hand panel of Figure 6.13 represent the two predictors that are related to the response, while the grey lines represent the unrelated predictors; these are often referred to as *signal* and *noise* variables, respectively. Not only has the lasso correctly given much larger coefficient estimates to the two signal predictors, but also the minimum cross-validation error corresponds to a set of coefficient estimates for which only the signal variables are non-zero. Hence cross-validation together with the lasso has correctly identified the two signal variables in the model, even though this is a challenging setting, with $p = 45$ variables and only $n = 50$ observations. In contrast, the least squares solution—displayed on the far right of the right-hand panel of Figure 6.13—assigns a large coefficient estimate to only one of the two signal variables.

signal

6.3 Dimension Reduction Methods

The methods that we have discussed so far in this chapter have controlled variance in two different ways, either by using a subset of the original variables, or by shrinking their coefficients toward zero. All of these methods

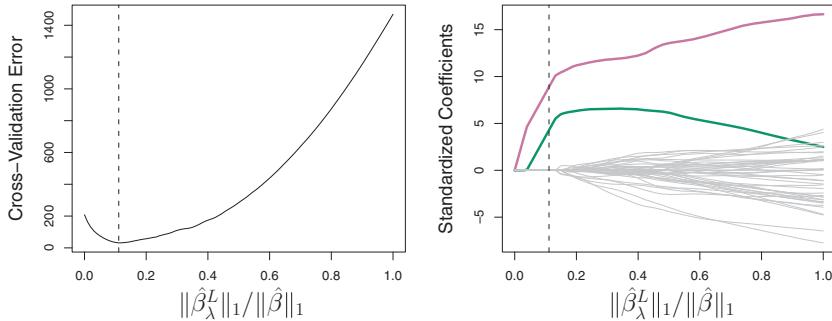


FIGURE 6.13. Left: Ten-fold cross-validation MSE for the lasso, applied to the sparse simulated data set from Figure 6.9. Right: The corresponding lasso coefficient estimates are displayed. The vertical dashed lines indicate the lasso fit for which the cross-validation error is smallest.

are defined using the original predictors, X_1, X_2, \dots, X_p . We now explore a class of approaches that *transform* the predictors and then fit a least squares model using the transformed variables. We will refer to these techniques as *dimension reduction* methods.

Let Z_1, Z_2, \dots, Z_M represent $M < p$ linear combinations of our original p predictors. That is,

$$Z_m = \sum_{j=1}^p \phi_{jm} X_j \quad (6.16)$$

dimension
reduction
linear
combination

for some constants $\phi_{1m}, \phi_{2m}, \dots, \phi_{pm}$, $m = 1, \dots, M$. We can then fit the linear regression model

$$y_i = \theta_0 + \sum_{m=1}^M \theta_m z_{im} + \epsilon_i, \quad i = 1, \dots, n, \quad (6.17)$$

using least squares. Note that in (6.17), the regression coefficients are given by $\theta_0, \theta_1, \dots, \theta_M$. If the constants $\phi_{1m}, \phi_{2m}, \dots, \phi_{pm}$ are chosen wisely, then such dimension reduction approaches can often outperform least squares regression. In other words, fitting (6.17) using least squares can lead to better results than fitting (6.1) using least squares.

The term *dimension reduction* comes from the fact that this approach reduces the problem of estimating the $p+1$ coefficients $\beta_0, \beta_1, \dots, \beta_p$ to the simpler problem of estimating the $M+1$ coefficients $\theta_0, \theta_1, \dots, \theta_M$, where $M < p$. In other words, the dimension of the problem has been reduced from $p+1$ to $M+1$.

Notice that from (6.16),

$$\sum_{m=1}^M \theta_m z_{im} = \sum_{m=1}^M \theta_m \sum_{j=1}^p \phi_{jm} x_{ij} = \sum_{j=1}^p \sum_{m=1}^M \theta_m \phi_{jm} x_{ij} = \sum_{j=1}^p \beta_j x_{ij},$$

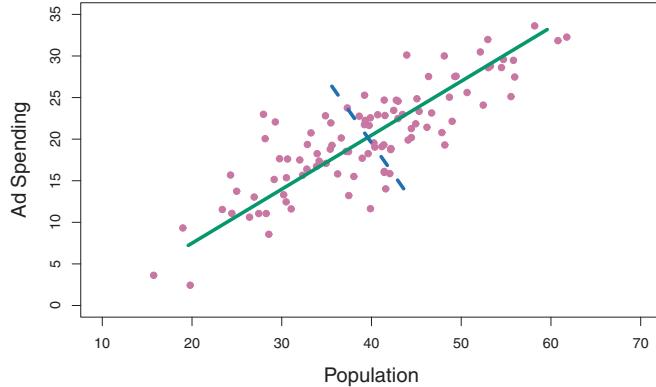


FIGURE 6.14. The population size (`pop`) and ad spending (`ad`) for 100 different cities are shown as purple circles. The green solid line indicates the first principal component, and the blue dashed line indicates the second principal component.

where

$$\beta_j = \sum_{m=1}^M \theta_m \phi_{jm}. \quad (6.18)$$

Hence (6.17) can be thought of as a special case of the original linear regression model given by (6.1). Dimension reduction serves to constrain the estimated β_j coefficients, since now they must take the form (6.18). This constraint on the form of the coefficients has the potential to bias the coefficient estimates. However, in situations where p is large relative to n , selecting a value of $M \ll p$ can significantly reduce the variance of the fitted coefficients. If $M = p$, and all the Z_m are linearly independent, then (6.18) poses no constraints. In this case, no dimension reduction occurs, and so fitting (6.17) is equivalent to performing least squares on the original p predictors.

All dimension reduction methods work in two steps. First, the transformed predictors Z_1, Z_2, \dots, Z_M are obtained. Second, the model is fit using these M predictors. However, the choice of Z_1, Z_2, \dots, Z_M , or equivalently, the selection of the ϕ_{jm} 's, can be achieved in different ways. In this chapter, we will consider two approaches for this task: *principal components* and *partial least squares*.

6.3.1 Principal Components Regression

Principal components analysis (PCA) is a popular approach for deriving a low-dimensional set of features from a large set of variables. PCA is discussed in greater detail as a tool for *unsupervised learning* in Chapter 10. Here we describe its use as a dimension reduction technique for regression.

principal
components
analysis

An Overview of Principal Components Analysis

PCA is a technique for reducing the dimension of a $n \times p$ data matrix \mathbf{X} . The *first principal component* direction of the data is that along which the observations *vary the most*. For instance, consider Figure 6.14, which shows population size (pop) in tens of thousands of people, and ad spending for a particular company (ad) in thousands of dollars, for 100 cities. The green solid line represents the first principal component direction of the data. We can see by eye that this is the direction along which there is the greatest variability in the data. That is, if we *projected* the 100 observations onto this line (as shown in the left-hand panel of Figure 6.15), then the resulting projected observations would have the largest possible variance; projecting the observations onto any other line would yield projected observations with lower variance. Projecting a point onto a line simply involves finding the location on the line which is closest to the point.

The first principal component is displayed graphically in Figure 6.14, but how can it be summarized mathematically? It is given by the formula

$$Z_1 = 0.839 \times (\text{pop} - \overline{\text{pop}}) + 0.544 \times (\text{ad} - \overline{\text{ad}}). \quad (6.19)$$

Here $\phi_{11} = 0.839$ and $\phi_{21} = 0.544$ are the principal component loadings, which define the direction referred to above. In (6.19), $\overline{\text{pop}}$ indicates the mean of all pop values in this data set, and $\overline{\text{ad}}$ indicates the mean of all advertising spending. The idea is that out of every possible *linear combination* of pop and ad such that $\phi_{11}^2 + \phi_{21}^2 = 1$, this particular linear combination yields the highest variance: i.e. this is the linear combination for which $\text{Var}(\phi_{11} \times (\text{pop} - \overline{\text{pop}}) + \phi_{21} \times (\text{ad} - \overline{\text{ad}}))$ is maximized. It is necessary to consider only linear combinations of the form $\phi_{11}^2 + \phi_{21}^2 = 1$, since otherwise we could increase ϕ_{11} and ϕ_{21} arbitrarily in order to blow up the variance. In (6.19), the two loadings are both positive and have similar size, and so Z_1 is almost an *average* of the two variables.

Since $n = 100$, pop and ad are vectors of length 100, and so is Z_1 in (6.19). For instance,

$$z_{i1} = 0.839 \times (\text{pop}_i - \overline{\text{pop}}) + 0.544 \times (\text{ad}_i - \overline{\text{ad}}). \quad (6.20)$$

The values of z_{11}, \dots, z_{n1} are known as the *principal component scores*, and can be seen in the right-hand panel of Figure 6.15.

There is also another interpretation for PCA: the first principal component vector defines the line that is *as close as possible* to the data. For instance, in Figure 6.14, the first principal component line minimizes the sum of the squared perpendicular distances between each point and the line. These distances are plotted as dashed line segments in the left-hand panel of Figure 6.15, in which the crosses represent the *projection* of each point onto the first principal component line. The first principal component has been chosen so that the projected observations are *as close as possible* to the original observations.

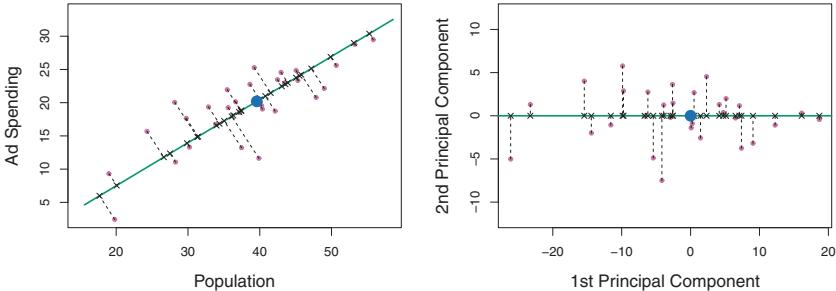


FIGURE 6.15. A subset of the advertising data. The mean pop and ad budgets are indicated with a blue circle. Left: The first principal component direction is shown in green. It is the dimension along which the data vary the most, and it also defines the line that is closest to all n of the observations. The distances from each observation to the principal component are represented using the black dashed line segments. The blue dot represents $(\bar{\text{pop}}, \bar{\text{ad}})$. Right: The left-hand panel has been rotated so that the first principal component direction coincides with the x -axis.

In the right-hand panel of Figure 6.15, the left-hand panel has been rotated so that the first principal component direction coincides with the x -axis. It is possible to show that the *first principal component score* for the i th observation, given in (6.20), is the distance in the x -direction of the i th cross from zero. So for example, the point in the bottom-left corner of the left-hand panel of Figure 6.15 has a large negative principal component score, $z_{i1} = -26.1$, while the point in the top-right corner has a large positive score, $z_{i1} = 18.7$. These scores can be computed directly using (6.20).

We can think of the values of the principal component Z_1 as single-number summaries of the joint pop and ad budgets for each location. In this example, if $z_{i1} = 0.839 \times (\text{pop}_i - \bar{\text{pop}}) + 0.544 \times (\text{ad}_i - \bar{\text{ad}}) < 0$, then this indicates a city with below-average population size and below-average ad spending. A positive score suggests the opposite. How well can a single number represent both pop and ad ? In this case, Figure 6.14 indicates that pop and ad have approximately a linear relationship, and so we might expect that a single-number summary will work well. Figure 6.16 displays z_{i1} versus both pop and ad . The plots show a strong relationship between the first principal component and the two features. In other words, the first principal component appears to capture most of the information contained in the pop and ad predictors.

So far we have concentrated on the first principal component. In general, one can construct up to p distinct principal components. The second principal component Z_2 is a linear combination of the variables that is uncorrelated with Z_1 , and has largest variance subject to this constraint. The second principal component direction is illustrated as a dashed blue line in Figure 6.14. It turns out that the zero correlation condition of Z_1 with Z_2

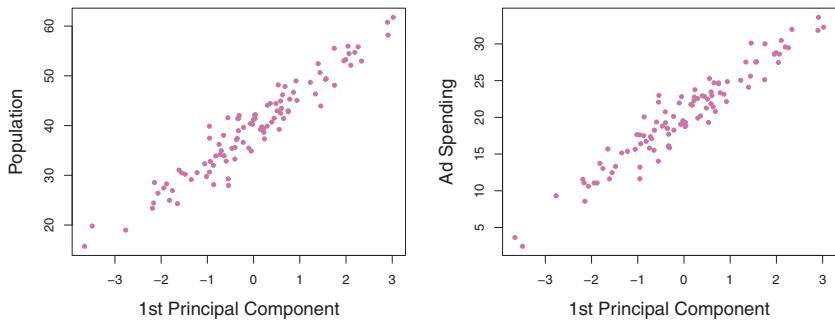


FIGURE 6.16. Plots of the first principal component scores z_{i1} versus pop and ad . The relationships are strong.

is equivalent to the condition that the direction must be *perpendicular*, or *orthogonal*, to the first principal component direction. The second principal component is given by the formula

$$Z_2 = 0.544 \times (\text{pop} - \bar{\text{pop}}) - 0.839 \times (\text{ad} - \bar{\text{ad}}).$$

Since the advertising data has two predictors, the first two principal components contain all of the information that is in pop and ad . However, by construction, the first component will contain the most information. Consider, for example, the much larger variability of z_{i1} (the x -axis) versus z_{i2} (the y -axis) in the right-hand panel of Figure 6.15. The fact that the second principal component scores are much closer to zero indicates that this component captures far less information. As another illustration, Figure 6.17 displays z_{i2} versus pop and ad . There is little relationship between the second principal component and these two predictors, again suggesting that in this case, one only needs the first principal component in order to accurately represent the pop and ad budgets.

With two-dimensional data, such as in our advertising example, we can construct at most two principal components. However, if we had other predictors, such as population age, income level, education, and so forth, then additional components could be constructed. They would successively maximize variance, subject to the constraint of being uncorrelated with the preceding components.

The Principal Components Regression Approach

The *principal components regression* (PCR) approach involves constructing the first M principal components, Z_1, \dots, Z_M , and then using these components as the predictors in a linear regression model that is fit using least squares. The key idea is that often a small number of principal components suffice to explain most of the variability in the data, as well as the relationship with the response. In other words, we assume that *the directions in which X_1, \dots, X_p show the most variation are the directions that are associated with Y* . While this assumption is not guaranteed

perpendicular
orthogonal

principal
components
regression

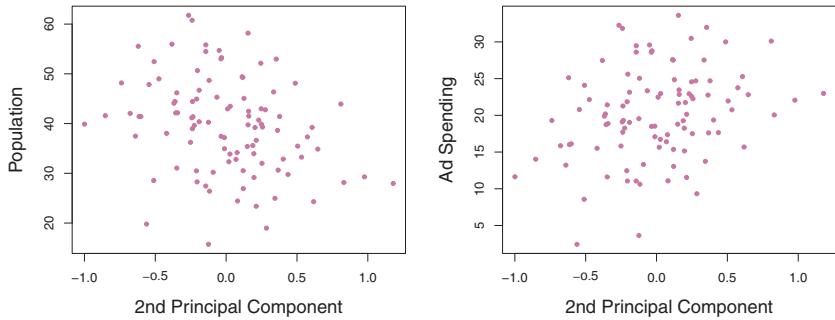


FIGURE 6.17. Plots of the second principal component scores z_{i2} versus `pop` and `ad`. The relationships are weak.

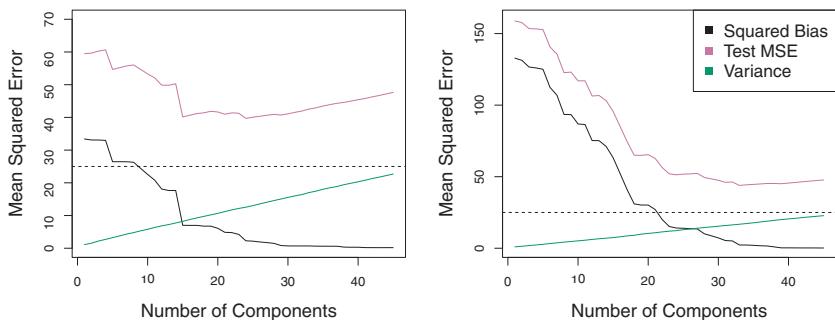


FIGURE 6.18. PCR was applied to two simulated data sets. Left: Simulated data from Figure 6.8. Right: Simulated data from Figure 6.9.

to be true, it often turns out to be a reasonable enough approximation to give good results.

If the assumption underlying PCR holds, then fitting a least squares model to Z_1, \dots, Z_M will lead to better results than fitting a least squares model to X_1, \dots, X_p , since most or all of the information in the data that relates to the response is contained in Z_1, \dots, Z_M , and by estimating only $M \ll p$ coefficients we can mitigate overfitting. In the advertising data, the first principal component explains most of the variance in both `pop` and `ad`, so a principal component regression that uses this single variable to predict some response of interest, such as `sales`, will likely perform quite well.

Figure 6.18 displays the PCR fits on the simulated data sets from Figures 6.8 and 6.9. Recall that both data sets were generated using $n = 50$ observations and $p = 45$ predictors. However, while the response in the first data set was a function of all the predictors, the response in the second data set was generated using only two of the predictors. The curves are plotted as a function of M , the number of principal components used as predictors in the regression model. As more principal components are used in

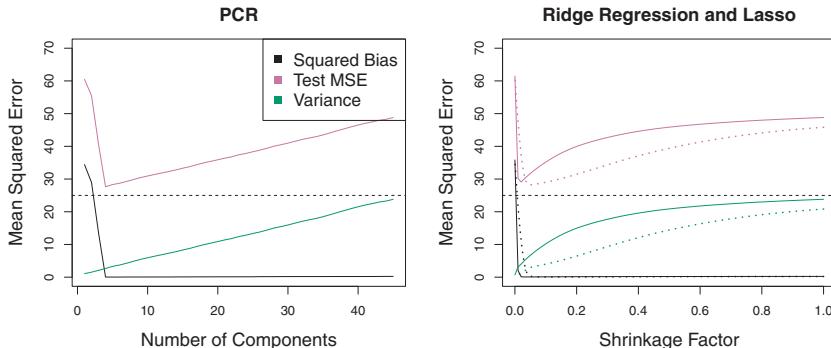


FIGURE 6.19. PCR, ridge regression, and the lasso were applied to a simulated data set in which the first five principal components of X contain all the information about the response Y . In each panel, the irreducible error $\text{Var}(\epsilon)$ is shown as a horizontal dashed line. Left: Results for PCR. Right: Results for lasso (solid) and ridge regression (dotted). The x -axis displays the shrinkage factor of the coefficient estimates, defined as the ℓ_2 norm of the shrunken coefficient estimates divided by the ℓ_2 norm of the least squares estimate.

the regression model, the bias decreases, but the variance increases. This results in a typical U-shape for the mean squared error. When $M = p = 45$, then PCR amounts simply to a least squares fit using all of the original predictors. The figure indicates that performing PCR with an appropriate choice of M can result in a substantial improvement over least squares, especially in the left-hand panel. However, by examining the ridge regression and lasso results in Figures 6.5, 6.8, and 6.9, we see that PCR does not perform as well as the two shrinkage methods in this example.

The relatively worse performance of PCR in Figure 6.18 is a consequence of the fact that the data were generated in such a way that many principal components are required in order to adequately model the response. In contrast, PCR will tend to do well in cases when the first few principal components are sufficient to capture most of the variation in the predictors as well as the relationship with the response. The left-hand panel of Figure 6.19 illustrates the results from another simulated data set designed to be more favorable to PCR. Here the response was generated in such a way that it depends exclusively on the first five principal components. Now the bias drops to zero rapidly as M , the number of principal components used in PCR, increases. The mean squared error displays a clear minimum at $M = 5$. The right-hand panel of Figure 6.19 displays the results on these data using ridge regression and the lasso. All three methods offer a significant improvement over least squares. However, PCR and ridge regression slightly outperform the lasso.

We note that even though PCR provides a simple way to perform regression using $M < p$ predictors, it is *not* a feature selection method. This is because each of the M principal components used in the regression

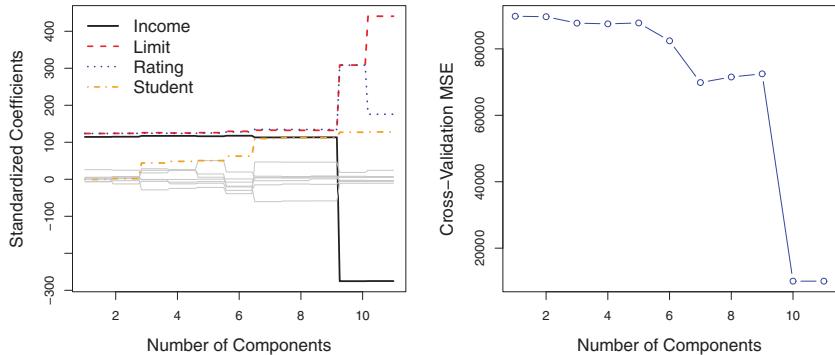


FIGURE 6.20. Left: PCR standardized coefficient estimates on the `Credit` data set for different values of M . Right: The ten-fold cross validation MSE obtained using PCR, as a function of M .

is a linear combination of all p of the *original* features. For instance, in (6.19), Z_1 was a linear combination of both `pop` and `ad`. Therefore, while PCR often performs quite well in many practical settings, it does not result in the development of a model that relies upon a small set of the original features. In this sense, PCR is more closely related to ridge regression than to the lasso. In fact, one can show that PCR and ridge regression are very closely related. One can even think of ridge regression as a continuous version of PCR!⁴

In PCR, the number of principal components, M , is typically chosen by cross-validation. The results of applying PCR to the `Credit` data set are shown in Figure 6.20; the right-hand panel displays the cross-validation errors obtained, as a function of M . On these data, the lowest cross-validation error occurs when there are $M = 10$ components; this corresponds to almost no dimension reduction at all, since PCR with $M = 11$ is equivalent to simply performing least squares.

When performing PCR, we generally recommend *standardizing* each predictor, using (6.6), prior to generating the principal components. This standardization ensures that all variables are on the same scale. In the absence of standardization, the high-variance variables will tend to play a larger role in the principal components obtained, and the scale on which the variables are measured will ultimately have an effect on the final PCR model. However, if the variables are all measured in the same units (say, kilograms, or inches), then one might choose not to standardize them.

⁴More details can be found in Section 3.5 of *Elements of Statistical Learning* by Hastie, Tibshirani, and Friedman.

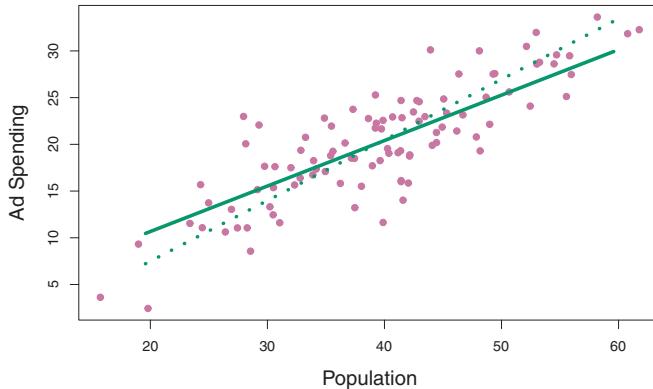


FIGURE 6.21. For the advertising data, the first PLS direction (solid line) and first PCR direction (dotted line) are shown.

6.3.2 Partial Least Squares

The PCR approach that we just described involves identifying linear combinations, or *directions*, that best represent the predictors X_1, \dots, X_p . These directions are identified in an *unsupervised* way, since the response Y is not used to help determine the principal component directions. That is, the response does not *supervise* the identification of the principal components. Consequently, PCR suffers from a drawback: there is no guarantee that the directions that best explain the predictors will also be the best directions to use for predicting the response. Unsupervised methods are discussed further in Chapter 10.

We now present *partial least squares* (PLS), a *supervised* alternative to PCR. Like PCR, PLS is a dimension reduction method, which first identifies a new set of features Z_1, \dots, Z_M that are linear combinations of the original features, and then fits a linear model via least squares using these M new features. But unlike PCR, PLS identifies these new features in a supervised way—that is, it makes use of the response Y in order to identify new features that not only approximate the old features well, but also that *are related to the response*. Roughly speaking, the PLS approach attempts to find directions that help explain both the response and the predictors.

partial least
squares

We now describe how the first PLS direction is computed. After standardizing the p predictors, PLS computes the first direction Z_1 by setting each ϕ_{j1} in (6.16) equal to the coefficient from the simple linear regression of Y onto X_j . One can show that this coefficient is proportional to the correlation between Y and X_j . Hence, in computing $Z_1 = \sum_{j=1}^p \phi_{j1} X_j$, PLS places the highest weight on the variables that are most strongly related to the response.

Figure 6.21 displays an example of PLS on the advertising data. The solid green line indicates the first PLS direction, while the dotted line shows the first principal component direction. PLS has chosen a direction that has less change in the `ad` dimension per unit change in the `pop` dimension, relative

to PCA. This suggests that `pop` is more highly correlated with the response than is `ad`. The PLS direction does not fit the predictors as closely as does PCA, but it does a better job explaining the response.

To identify the second PLS direction we first *adjust* each of the variables for Z_1 , by regressing each variable on Z_1 and taking *residuals*. These residuals can be interpreted as the remaining information that has not been explained by the first PLS direction. We then compute Z_2 using this *orthogonalized* data in exactly the same fashion as Z_1 was computed based on the original data. This iterative approach can be repeated M times to identify multiple PLS components Z_1, \dots, Z_M . Finally, at the end of this procedure, we use least squares to fit a linear model to predict Y using Z_1, \dots, Z_M in exactly the same fashion as for PCR.

As with PCR, the number M of partial least squares directions used in PLS is a tuning parameter that is typically chosen by cross-validation. We generally standardize the predictors and response before performing PLS.

PLS is popular in the field of chemometrics, where many variables arise from digitized spectrometry signals. In practice it often performs no better than ridge regression or PCR. While the supervised dimension reduction of PLS can reduce bias, it also has the potential to increase variance, so that the overall benefit of PLS relative to PCR is a wash.

6.4 Considerations in High Dimensions

6.4.1 High-Dimensional Data

Most traditional statistical techniques for regression and classification are intended for the *low-dimensional* setting in which n , the number of observations, is much greater than p , the number of features. This is due in part to the fact that throughout most of the field's history, the bulk of scientific problems requiring the use of statistics have been low-dimensional. For instance, consider the task of developing a model to predict a patient's blood pressure on the basis of his or her age, gender, and body mass index (BMI). There are three predictors, or four if an intercept is included in the model, and perhaps several thousand patients for whom blood pressure and age, gender, and BMI are available. Hence $n \gg p$, and so the problem is low-dimensional. (By dimension here we are referring to the size of p .)

low-dimensional

In the past 20 years, new technologies have changed the way that data are collected in fields as diverse as finance, marketing, and medicine. It is now commonplace to collect an almost unlimited number of feature measurements (p very large). While p can be extremely large, the number of observations n is often limited due to cost, sample availability, or other considerations. Two examples are as follows:

1. Rather than predicting blood pressure on the basis of just age, gender, and BMI, one might also collect measurements for half a million

single nucleotide polymorphisms (SNPs; these are individual DNA mutations that are relatively common in the population) for inclusion in the predictive model. Then $n \approx 200$ and $p \approx 500,000$.

2. A marketing analyst interested in understanding people's online shopping patterns could treat as features all of the search terms entered by users of a search engine. This is sometimes known as the "bag-of-words" model. The same researcher might have access to the search histories of only a few hundred or a few thousand search engine users who have consented to share their information with the researcher. For a given user, each of the p search terms is scored present (0) or absent (1), creating a large binary feature vector. Then $n \approx 1,000$ and p is much larger.

Data sets containing more features than observations are often referred to as *high-dimensional*. Classical approaches such as least squares linear regression are not appropriate in this setting. Many of the issues that arise in the analysis of high-dimensional data were discussed earlier in this book, since they apply also when $n > p$: these include the role of the bias-variance trade-off and the danger of overfitting. Though these issues are always relevant, they can become particularly important when the number of features is very large relative to the number of observations.

high-dimensional

We have defined the *high-dimensional setting* as the case where the number of features p is larger than the number of observations n . But the considerations that we will now discuss certainly also apply if p is slightly smaller than n , and are best always kept in mind when performing supervised learning.

6.4.2 What Goes Wrong in High Dimensions?

In order to illustrate the need for extra care and specialized techniques for regression and classification when $p > n$, we begin by examining what can go wrong if we apply a statistical technique not intended for the high-dimensional setting. For this purpose, we examine least squares regression. But the same concepts apply to logistic regression, linear discriminant analysis, and other classical statistical approaches.

When the number of features p is as large as, or larger than, the number of observations n , least squares as described in Chapter 3 cannot (or rather, *should not*) be performed. The reason is simple: regardless of whether or not there truly is a relationship between the features and the response, least squares will yield a set of coefficient estimates that result in a perfect fit to the data, such that the residuals are zero.

An example is shown in Figure 6.22 with $p = 1$ feature (plus an intercept) in two cases: when there are 20 observations, and when there are only two observations. When there are 20 observations, $n > p$ and the least

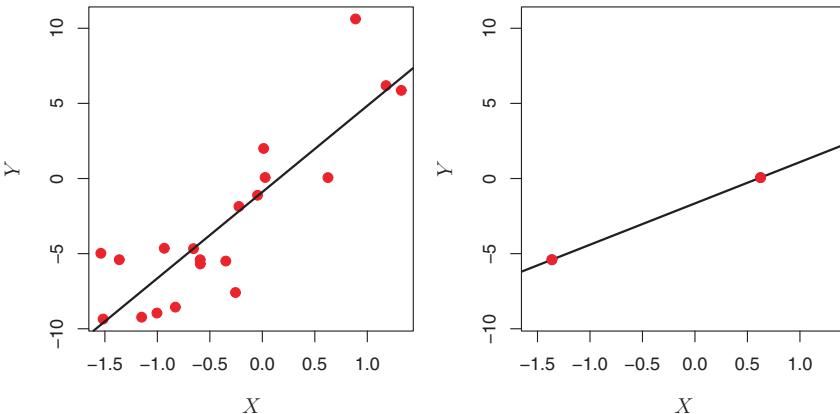


FIGURE 6.22. Left: Least squares regression in the low-dimensional setting. Right: Least squares regression with $n = 2$ observations and two parameters to be estimated (an intercept and a coefficient).

squares regression line does not perfectly fit the data; instead, the regression line seeks to approximate the 20 observations as well as possible. On the other hand, when there are only two observations, then regardless of the values of those observations, the regression line will fit the data exactly. This is problematic because this perfect fit will almost certainly lead to overfitting of the data. In other words, though it is possible to perfectly fit the training data in the high-dimensional setting, the resulting linear model will perform extremely poorly on an independent test set, and therefore does not constitute a useful model. In fact, we can see that this happened in Figure 6.22: the least squares line obtained in the right-hand panel will perform very poorly on a test set comprised of the observations in the left-hand panel. The problem is simple: when $p > n$ or $p \approx n$, a simple least squares regression line is too *flexible* and hence overfits the data.

Figure 6.23 further illustrates the risk of carelessly applying least squares when the number of features p is large. Data were simulated with $n = 20$ observations, and regression was performed with between 1 and 20 features, each of which was completely unrelated to the response. As shown in the figure, the model R^2 increases to 1 as the number of features included in the model increases, and correspondingly the training set MSE decreases to 0 as the number of features increases, *even though the features are completely unrelated to the response*. On the other hand, the MSE on an *independent test set* becomes extremely large as the number of features included in the model increases, because including the additional predictors leads to a vast increase in the variance of the coefficient estimates. Looking at the test set MSE, it is clear that the best model contains at most a few variables. However, someone who carelessly examines only the R^2 or the training set MSE might erroneously conclude that the model with the greatest number of variables is best. This indicates the importance of applying extra care

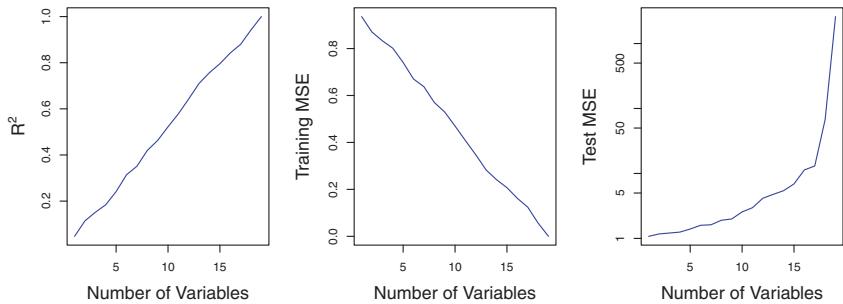


FIGURE 6.23. On a simulated example with $n = 20$ training observations, features that are completely unrelated to the outcome are added to the model. Left: The R^2 increases to 1 as more features are included. Center: The training set MSE decreases to 0 as more features are included. Right: The test set MSE increases as more features are included.

when analyzing data sets with a large number of variables, and of always evaluating model performance on an independent test set.

In Section 6.1.3, we saw a number of approaches for adjusting the training set RSS or R^2 in order to account for the number of variables used to fit a least squares model. Unfortunately, the C_p , AIC, and BIC approaches are not appropriate in the high-dimensional setting, because estimating $\hat{\sigma}^2$ is problematic. (For instance, the formula for $\hat{\sigma}^2$ from Chapter 3 yields an estimate $\hat{\sigma}^2 = 0$ in this setting.) Similarly, problems arise in the application of adjusted R^2 in the high-dimensional setting, since one can easily obtain a model with an adjusted R^2 value of 1. Clearly, alternative approaches that are better-suited to the high-dimensional setting are required.

6.4.3 Regression in High Dimensions

It turns out that many of the methods seen in this chapter for fitting *less flexible* least squares models, such as forward stepwise selection, ridge regression, the lasso, and principal components regression, are particularly useful for performing regression in the high-dimensional setting. Essentially, these approaches avoid overfitting by using a less flexible fitting approach than least squares.

Figure 6.24 illustrates the performance of the lasso in a simple simulated example. There are $p = 20, 50$, or $2,000$ features, of which 20 are truly associated with the outcome. The lasso was performed on $n = 100$ training observations, and the mean squared error was evaluated on an independent test set. As the number of features increases, the test set error increases. When $p = 20$, the lowest validation set error was achieved when λ in (6.7) was small; however, when p was larger then the lowest validation set error was achieved using a larger value of λ . In each boxplot, rather than reporting the values of λ used, the *degrees of freedom* of the resulting

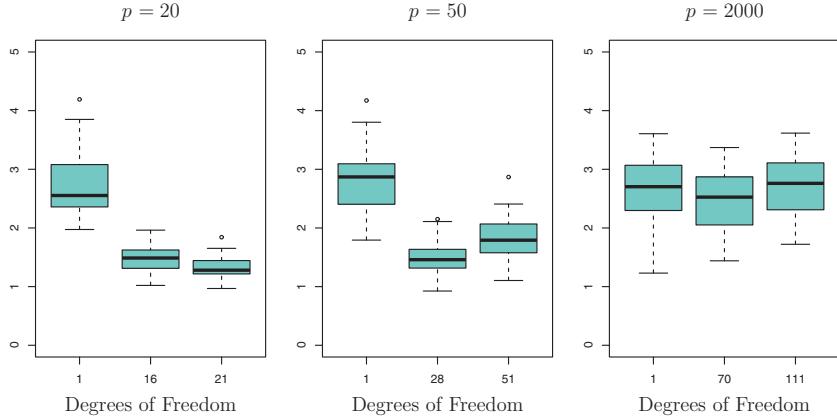


FIGURE 6.24. The lasso was performed with $n = 100$ observations and three values of p , the number of features. Of the p features, 20 were associated with the response. The boxplots show the test MSEs that result using three different values of the tuning parameter λ in (6.7). For ease of interpretation, rather than reporting λ , the degrees of freedom are reported; for the lasso this turns out to be simply the number of estimated non-zero coefficients. When $p = 20$, the lowest test MSE was obtained with the smallest amount of regularization. When $p = 50$, the lowest test MSE was achieved when there is a substantial amount of regularization. When $p = 2,000$ the lasso performed poorly regardless of the amount of regularization, due to the fact that only 20 of the 2,000 features truly are associated with the outcome.

lasso solution is displayed; this is simply the number of non-zero coefficient estimates in the lasso solution, and is a measure of the flexibility of the lasso fit. Figure 6.24 highlights three important points: (1) regularization or shrinkage plays a key role in high-dimensional problems, (2) appropriate tuning parameter selection is crucial for good predictive performance, and (3) the test error tends to increase as the dimensionality of the problem (i.e. the number of features or predictors) increases, unless the additional features are truly associated with the response.

The third point above is in fact a key principle in the analysis of high-dimensional data, which is known as the *curse of dimensionality*. One might think that as the number of features used to fit a model increases, the quality of the fitted model will increase as well. However, comparing the left-hand and right-hand panels in Figure 6.24, we see that this is not necessarily the case: in this example, the test set MSE almost doubles as p increases from 20 to 2,000. In general, *adding additional signal features that are truly associated with the response will improve the fitted model*, in the sense of leading to a reduction in test set error. However, adding noise features that are not truly associated with the response will lead to a deterioration in the fitted model, and consequently an increased test set error. This is because noise features increase the dimensionality of the

curse of dimensionality

problem, exacerbating the risk of overfitting (since noise features may be assigned nonzero coefficients due to chance associations with the response on the training set) without any potential upside in terms of improved test set error. Thus, we see that new technologies that allow for the collection of measurements for thousands or millions of features are a double-edged sword: they can lead to improved predictive models if these features are in fact relevant to the problem at hand, but will lead to worse results if the features are not relevant. Even if they are relevant, the variance incurred in fitting their coefficients may outweigh the reduction in bias that they bring.

6.4.4 Interpreting Results in High Dimensions

When we perform the lasso, ridge regression, or other regression procedures in the high-dimensional setting, we must be quite cautious in the way that we report the results obtained. In Chapter 3, we learned about *multicollinearity*, the concept that the variables in a regression might be correlated with each other. In the high-dimensional setting, the multicollinearity problem is extreme: any variable in the model can be written as a linear combination of all of the other variables in the model. Essentially, this means that we can never know exactly which variables (if any) truly are predictive of the outcome, and we can never identify the *best* coefficients for use in the regression. At most, we can hope to assign large regression coefficients to variables that are correlated with the variables that truly are predictive of the outcome.

For instance, suppose that we are trying to predict blood pressure on the basis of half a million SNPs, and that forward stepwise selection indicates that 17 of those SNPs lead to a good predictive model on the training data. It would be incorrect to conclude that these 17 SNPs predict blood pressure more effectively than the other SNPs not included in the model. There are likely to be many sets of 17 SNPs that would predict blood pressure just as well as the selected model. If we were to obtain an independent data set and perform forward stepwise selection on that data set, we would likely obtain a model containing a different, and perhaps even non-overlapping, set of SNPs. This does not detract from the value of the model obtained—for instance, the model might turn out to be very effective in predicting blood pressure on an independent set of patients, and might be clinically useful for physicians. But we must be careful not to overstate the results obtained, and to make it clear that what we have identified is simply *one of many possible models* for predicting blood pressure, and that it must be further validated on independent data sets.

It is also important to be particularly careful in reporting errors and measures of model fit in the high-dimensional setting. We have seen that when $p > n$, it is easy to obtain a useless model that has zero residuals. Therefore, one should *never* use sum of squared errors, p-values, R^2

statistics, or other traditional measures of model fit on the training data as evidence of a good model fit in the high-dimensional setting. For instance, as we saw in Figure 6.23, one can easily obtain a model with $R^2 = 1$ when $p > n$. Reporting this fact might mislead others into thinking that a statistically valid and useful model has been obtained, whereas in fact this provides absolutely no evidence of a compelling model. It is important to instead report results on an independent test set, or cross-validation errors. For instance, the MSE or R^2 on an independent test set is a valid measure of model fit, but the MSE on the training set certainly is not.

6.5 Lab 1: Subset Selection Methods

6.5.1 Best Subset Selection

Here we apply the best subset selection approach to the `Hitters` data. We wish to predict a baseball player's `Salary` on the basis of various statistics associated with performance in the previous year.

First of all, we note that the `Salary` variable is missing for some of the players. The `is.na()` function can be used to identify the missing observations. It returns a vector of the same length as the input vector, with a `TRUE` for any elements that are missing, and a `FALSE` for non-missing elements. The `sum()` function can then be used to count all of the missing elements.

```
> library(ISLR)
> fix(Hitters)
> names(Hitters)
[1] "AtBat"      "Hits"       "HmRun"      "Runs"       "RBI"
[6] "Walks"       "Years"      "CAtBat"     "CHits"      "CHmRun"
[11] "CRuns"      "CRBI"       "CWalks"     "League"     "Division"
[16] "PutOuts"    "Assists"    "Errors"     "Salary"     "NewLeague"
> dim(Hitters)
[1] 322 20
> sum(is.na(Hitters$Salary))
[1] 59
```

Hence we see that `Salary` is missing for 59 players. The `na.omit()` function removes all of the rows that have missing values in any variable.

```
> Hitters=na.omit(Hitters)
> dim(Hitters)
[1] 263 20
> sum(is.na(Hitters))
[1] 0
```

The `regsubsets()` function (part of the `leaps` library) performs best subset selection by identifying the best model that contains a given number of predictors, where *best* is quantified using RSS. The syntax is the same as for `lm()`. The `summary()` command outputs the best set of variables for each model size.

`is.na()`
`sum()`

`regsubsets()`

final seven-component model PCR fit, 46.69 %. This is because PCR only attempts to maximize the amount of variance explained in the predictors, while PLS searches for directions that explain variance in both the predictors and the response.

6.8 Exercises

Conceptual

1. We perform best subset, forward stepwise, and backward stepwise selection on a single data set. For each approach, we obtain $p + 1$ models, containing $0, 1, 2, \dots, p$ predictors. Explain your answers:
 - (a) Which of the three models with k predictors has the smallest *training* RSS?
 - (b) Which of the three models with k predictors has the smallest *test* RSS?
 - (c) True or False:
 - i. The predictors in the k -variable model identified by forward stepwise are a subset of the predictors in the $(k+1)$ -variable model identified by forward stepwise selection.
 - ii. The predictors in the k -variable model identified by backward stepwise are a subset of the predictors in the $(k+1)$ -variable model identified by backward stepwise selection.
 - iii. The predictors in the k -variable model identified by backward stepwise are a subset of the predictors in the $(k+1)$ -variable model identified by forward stepwise selection.
 - iv. The predictors in the k -variable model identified by forward stepwise are a subset of the predictors in the $(k+1)$ -variable model identified by backward stepwise selection.
 - v. The predictors in the k -variable model identified by best subset are a subset of the predictors in the $(k+1)$ -variable model identified by best subset selection.
2. For parts (a) through (c), indicate which of i. through iv. is correct. Justify your answer.
 - (a) The lasso, relative to least squares, is:
 - i. More flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.
 - ii. More flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

- iii. Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.
 - iv. Less flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.
- (b) Repeat (a) for ridge regression relative to least squares.
- (c) Repeat (a) for non-linear methods relative to least squares.
3. Suppose we estimate the regression coefficients in a linear regression model by minimizing
- $$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^p |\beta_j| \leq s$$
- for a particular value of s . For parts (a) through (e), indicate which of i. through v. is correct. Justify your answer.
- (a) As we increase s from 0, the training RSS will:
 - i. Increase initially, and then eventually start decreasing in an inverted U shape.
 - ii. Decrease initially, and then eventually start increasing in a U shape.
 - iii. Steadily increase.
 - iv. Steadily decrease.
 - v. Remain constant.
 - (b) Repeat (a) for test RSS.
 - (c) Repeat (a) for variance.
 - (d) Repeat (a) for (squared) bias.
 - (e) Repeat (a) for the irreducible error.
4. Suppose we estimate the regression coefficients in a linear regression model by minimizing

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

for a particular value of λ . For parts (a) through (e), indicate which of i. through v. is correct. Justify your answer.

- (a) As we increase λ from 0, the training RSS will:
- Increase initially, and then eventually start decreasing in an inverted U shape.
 - Decrease initially, and then eventually start increasing in a U shape.
 - Steadily increase.
 - Steadily decrease.
 - Remain constant.
- (b) Repeat (a) for test RSS.
- (c) Repeat (a) for variance.
- (d) Repeat (a) for (squared) bias.
- (e) Repeat (a) for the irreducible error.
5. It is well-known that ridge regression tends to give similar coefficient values to correlated variables, whereas the lasso may give quite different coefficient values to correlated variables. We will now explore this property in a very simple setting. 
- Suppose that $n = 2$, $p = 2$, $x_{11} = x_{12}$, $x_{21} = x_{22}$. Furthermore, suppose that $y_1 + y_2 = 0$ and $x_{11} + x_{21} = 0$ and $x_{12} + x_{22} = 0$, so that the estimate for the intercept in a least squares, ridge regression, or lasso model is zero: $\hat{\beta}_0 = 0$.
- Write out the ridge regression optimization problem in this setting.
 - Argue that in this setting, the ridge coefficient estimates satisfy $\hat{\beta}_1 = \hat{\beta}_2$.
 - Write out the lasso optimization problem in this setting.
 - Argue that in this setting, the lasso coefficients $\hat{\beta}_1$ and $\hat{\beta}_2$ are not unique—in other words, there are many possible solutions to the optimization problem in (c). Describe these solutions.
6. We will now explore (6.12) and (6.13) further.

- Consider (6.12) with $p = 1$. For some choice of y_1 and $\lambda > 0$, plot (6.12) as a function of β_1 . Your plot should confirm that (6.12) is solved by (6.14).
- Consider (6.13) with $p = 1$. For some choice of y_1 and $\lambda > 0$, plot (6.13) as a function of β_1 . Your plot should confirm that (6.13) is solved by (6.15).

7. We will now derive the Bayesian connection to the lasso and ridge regression discussed in Section 6.2.2.



- (a) Suppose that $y_i = \beta_0 + \sum_{j=1}^p x_{ij}\beta_j + \epsilon_i$ where $\epsilon_1, \dots, \epsilon_n$ are independent and identically distributed from a $N(0, \sigma^2)$ distribution. Write out the likelihood for the data.
- (b) Assume the following prior for β : β_1, \dots, β_p are independent and identically distributed according to a double-exponential distribution with mean 0 and common scale parameter b : i.e. $p(\beta) = \frac{1}{2b} \exp(-|\beta|/b)$. Write out the posterior for β in this setting.
- (c) Argue that the lasso estimate is the *mode* for β under this posterior distribution.
- (d) Now assume the following prior for β : β_1, \dots, β_p are independent and identically distributed according to a normal distribution with mean zero and variance c . Write out the posterior for β in this setting.
- (e) Argue that the ridge regression estimate is both the *mode* and the *mean* for β under this posterior distribution.

Applied

8. In this exercise, we will generate simulated data, and will then use this data to perform best subset selection.

- (a) Use the `rnorm()` function to generate a predictor X of length $n = 100$, as well as a noise vector ϵ of length $n = 100$.
- (b) Generate a response vector Y of length $n = 100$ according to the model

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon,$$

where $\beta_0, \beta_1, \beta_2$, and β_3 are constants of your choice.

- (c) Use the `regsubsets()` function to perform best subset selection in order to choose the best model containing the predictors X, X^2, \dots, X^{10} . What is the best model obtained according to C_p , BIC, and adjusted R^2 ? Show some plots to provide evidence for your answer, and report the coefficients of the best model obtained. Note you will need to use the `data.frame()` function to create a single data set containing both X and Y .

9

Support Vector Machines

In this chapter, we discuss the *support vector machine* (SVM), an approach for classification that was developed in the computer science community in the 1990s and that has grown in popularity since then. SVMs have been shown to perform well in a variety of settings, and are often considered one of the best “out of the box” classifiers.

The support vector machine is a generalization of a simple and intuitive classifier called the *maximal margin classifier*, which we introduce in Section 9.1. Though it is elegant and simple, we will see that this classifier unfortunately cannot be applied to most data sets, since it requires that the classes be separable by a linear boundary. In Section 9.2, we introduce the *support vector classifier*, an extension of the maximal margin classifier that can be applied in a broader range of cases. Section 9.3 introduces the *support vector machine*, which is a further extension of the support vector classifier in order to accommodate non-linear class boundaries. Support vector machines are intended for the binary classification setting in which there are two classes; in Section 9.4 we discuss extensions of support vector machines to the case of more than two classes. In Section 9.5 we discuss the close connections between support vector machines and other statistical methods such as logistic regression.

People often loosely refer to the maximal margin classifier, the support vector classifier, and the support vector machine as “support vector machines”. To avoid confusion, we will carefully distinguish between these three notions in this chapter.

9.1 Maximal Margin Classifier

In this section, we define a hyperplane and introduce the concept of an optimal separating hyperplane.

9.1.1 What Is a Hyperplane?

In a p -dimensional space, a *hyperplane* is a flat affine subspace of dimension $p - 1$.¹ For instance, in two dimensions, a hyperplane is a flat one-dimensional subspace—in other words, a line. In three dimensions, a hyperplane is a flat two-dimensional subspace—that is, a plane. In $p > 3$ dimensions, it can be hard to visualize a hyperplane, but the notion of a $(p - 1)$ -dimensional flat subspace still applies.

The mathematical definition of a hyperplane is quite simple. In two dimensions, a hyperplane is defined by the equation

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0 \quad (9.1)$$

for parameters β_0, β_1 , and β_2 . When we say that (9.1) “defines” the hyperplane, we mean that any $X = (X_1, X_2)^T$ for which (9.1) holds is a point on the hyperplane. Note that (9.1) is simply the equation of a line, since indeed in two dimensions a hyperplane is a line.

Equation 9.1 can be easily extended to the p -dimensional setting:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p = 0 \quad (9.2)$$

defines a p -dimensional hyperplane, again in the sense that if a point $X = (X_1, X_2, \dots, X_p)^T$ in p -dimensional space (i.e. a vector of length p) satisfies (9.2), then X lies on the hyperplane.

Now, suppose that X does not satisfy (9.2); rather,

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p > 0. \quad (9.3)$$

Then this tells us that X lies to one side of the hyperplane. On the other hand, if

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p < 0, \quad (9.4)$$

then X lies on the other side of the hyperplane. So we can think of the hyperplane as dividing p -dimensional space into two halves. One can easily determine on which side of the hyperplane a point lies by simply calculating the sign of the left hand side of (9.2). A hyperplane in two-dimensional space is shown in Figure 9.1.

¹The word *affine* indicates that the subspace need not pass through the origin.

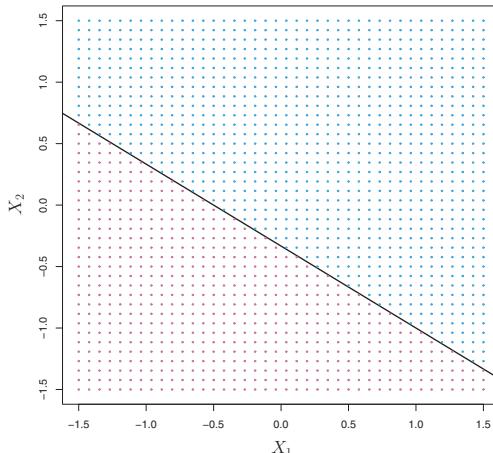


FIGURE 9.1. The hyperplane $1 + 2X_1 + 3X_2 = 0$ is shown. The blue region is the set of points for which $1 + 2X_1 + 3X_2 > 0$, and the purple region is the set of points for which $1 + 2X_1 + 3X_2 < 0$.

9.1.2 Classification Using a Separating Hyperplane

Now suppose that we have a $n \times p$ data matrix \mathbf{X} that consists of n training observations in p -dimensional space,

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}, \quad (9.5)$$

and that these observations fall into two classes—that is, $y_1, \dots, y_n \in \{-1, 1\}$ where -1 represents one class and 1 the other class. We also have a test observation, a p -vector of observed features $x^* = (x_1^* \ \dots \ x_p^*)^T$. Our goal is to develop a classifier based on the training data that will correctly classify the test observation using its feature measurements. We have seen a number of approaches for this task, such as linear discriminant analysis and logistic regression in Chapter 4, and classification trees, bagging, and boosting in Chapter 8. We will now see a new approach that is based upon the concept of a *separating hyperplane*.

Suppose that it is possible to construct a hyperplane that separates the training observations perfectly according to their class labels. Examples of three such *separating hyperplanes* are shown in the left-hand panel of Figure 9.2. We can label the observations from the blue class as $y_i = 1$ and

separating
hyperplane

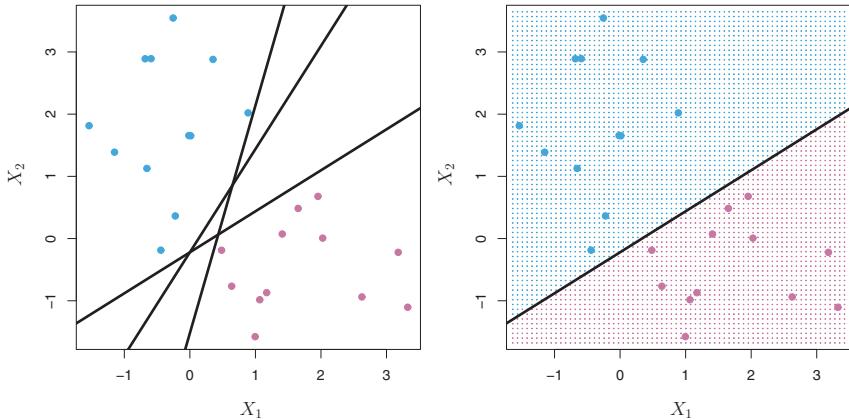


FIGURE 9.2. Left: There are two classes of observations, shown in blue and in purple, each of which has measurements on two variables. Three separating hyperplanes, out of many possible, are shown in black. Right: A separating hyperplane is shown in black. The blue and purple grid indicates the decision rule made by a classifier based on this separating hyperplane: a test observation that falls in the blue portion of the grid will be assigned to the blue class, and a test observation that falls into the purple portion of the grid will be assigned to the purple class.

those from the purple class as $y_i = -1$. Then a separating hyperplane has the property that

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} > 0 \text{ if } y_i = 1, \quad (9.6)$$

and

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} < 0 \text{ if } y_i = -1. \quad (9.7)$$

Equivalently, a separating hyperplane has the property that

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) > 0 \quad (9.8)$$

for all $i = 1, \dots, n$.

If a separating hyperplane exists, we can use it to construct a very natural classifier: a test observation is assigned a class depending on which side of the hyperplane it is located. The right-hand panel of Figure 9.2 shows an example of such a classifier. That is, we classify the test observation x^* based on the sign of $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \dots + \beta_p x_p^*$. If $f(x^*)$ is positive, then we assign the test observation to class 1, and if $f(x^*)$ is negative, then we assign it to class -1 . We can also make use of the *magnitude* of $f(x^*)$. If $f(x^*)$ is far from zero, then this means that x^* lies far from the hyperplane, and so we can be confident about our class assignment for x^* . On the other

hand, if $f(x^*)$ is close to zero, then x^* is located near the hyperplane, and so we are less certain about the class assignment for x^* . Not surprisingly, and as we see in Figure 9.2, a classifier that is based on a separating hyperplane leads to a linear decision boundary.

9.1.3 The Maximal Margin Classifier

In general, if our data can be perfectly separated using a hyperplane, then there will in fact exist an infinite number of such hyperplanes. This is because a given separating hyperplane can usually be shifted a tiny bit up or down, or rotated, without coming into contact with any of the observations. Three possible separating hyperplanes are shown in the left-hand panel of Figure 9.2. In order to construct a classifier based upon a separating hyperplane, we must have a reasonable way to decide which of the infinite possible separating hyperplanes to use.

A natural choice is the *maximal margin hyperplane* (also known as the *optimal separating hyperplane*), which is the separating hyperplane that is farthest from the training observations. That is, we can compute the (perpendicular) distance from each training observation to a given separating hyperplane; the smallest such distance is the minimal distance from the observations to the hyperplane, and is known as the *margin*. The maximal margin hyperplane is the separating hyperplane for which the margin is largest—that is, it is the hyperplane that has the farthest minimum distance to the training observations. We can then classify a test observation based on which side of the maximal margin hyperplane it lies. This is known as the *maximal margin classifier*. We hope that a classifier that has a large margin on the training data will also have a large margin on the test data, and hence will classify the test observations correctly. Although the maximal margin classifier is often successful, it can also lead to overfitting when p is large.

If $\beta_0, \beta_1, \dots, \beta_p$ are the coefficients of the maximal margin hyperplane, then the maximal margin classifier classifies the test observation x^* based on the sign of $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \dots + \beta_p x_p^*$.

Figure 9.3 shows the maximal margin hyperplane on the data set of Figure 9.2. Comparing the right-hand panel of Figure 9.2 to Figure 9.3, we see that the maximal margin hyperplane shown in Figure 9.3 does indeed result in a greater minimal distance between the observations and the separating hyperplane—that is, a larger margin. In a sense, the maximal margin hyperplane represents the mid-line of the widest “slab” that we can insert between the two classes.

Examining Figure 9.3, we see that three training observations are equidistant from the maximal margin hyperplane and lie along the dashed lines indicating the width of the margin. These three observations are known as

maximal
margin
hyperplane
optimal
separating
hyperplane
margin

maximal
margin
classifier

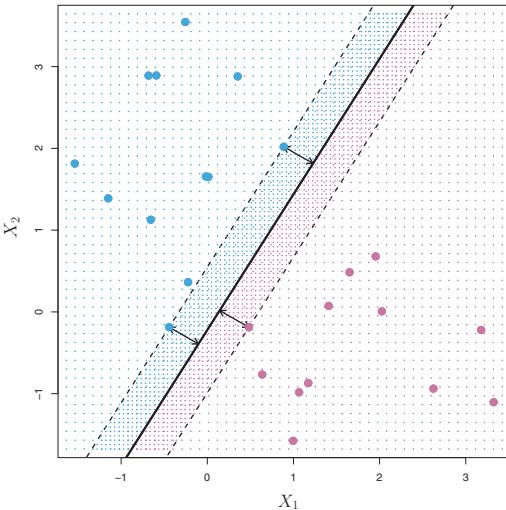


FIGURE 9.3. There are two classes of observations, shown in blue and in purple. The maximal margin hyperplane is shown as a solid line. The margin is the distance from the solid line to either of the dashed lines. The two blue points and the purple point that lie on the dashed lines are the support vectors, and the distance from those points to the hyperplane is indicated by arrows. The purple and blue grid indicates the decision rule made by a classifier based on this separating hyperplane.

support vectors, since they are vectors in p -dimensional space (in Figure 9.3, $p = 2$) and they “support” the maximal margin hyperplane in the sense that if these points were moved slightly then the maximal margin hyperplane would move as well. Interestingly, the maximal margin hyperplane depends directly on the support vectors, but not on the other observations: a movement to any of the other observations would not affect the separating hyperplane, provided that the observation’s movement does not cause it to cross the boundary set by the margin. The fact that the maximal margin hyperplane depends directly on only a small subset of the observations is an important property that will arise later in this chapter when we discuss the support vector classifier and support vector machines.

support
vector

9.1.4 Construction of the Maximal Margin Classifier

We now consider the task of constructing the maximal margin hyperplane based on a set of n training observations $x_1, \dots, x_n \in \mathbb{R}^p$ and associated class labels $y_1, \dots, y_n \in \{-1, 1\}$. Briefly, the maximal margin hyperplane is the solution to the optimization problem

$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{maximize}} M \quad (9.9)$$

$$\text{subject to } \sum_{j=1}^p \beta_j^2 = 1, \quad (9.10)$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n. \quad (9.11)$$

This optimization problem (9.9)–(9.11) is actually simpler than it looks. First of all, the constraint in (9.11) that

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n$$

guarantees that each observation will be on the correct side of the hyperplane, provided that M is positive. (Actually, for each observation to be on the correct side of the hyperplane we would simply need $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) > 0$, so the constraint in (9.11) in fact requires that each observation be on the correct side of the hyperplane, with some cushion, provided that M is positive.)

Second, note that (9.10) is not really a constraint on the hyperplane, since if $\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} = 0$ defines a hyperplane, then so does $k(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) = 0$ for any $k \neq 0$. However, (9.10) adds meaning to (9.11); one can show that with this constraint the perpendicular distance from the i th observation to the hyperplane is given by

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}).$$

Therefore, the constraints (9.10) and (9.11) ensure that each observation is on the correct side of the hyperplane and at least a distance M from the hyperplane. Hence, M represents the margin of our hyperplane, and the optimization problem chooses $\beta_0, \beta_1, \dots, \beta_p$ to maximize M . This is exactly the definition of the maximal margin hyperplane! The problem (9.9)–(9.11) can be solved efficiently, but details of this optimization are outside of the scope of this book.

9.1.5 The Non-separable Case

The maximal margin classifier is a very natural way to perform classification, *if a separating hyperplane exists*. However, as we have hinted, in many cases no separating hyperplane exists, and so there is no maximal margin classifier. In this case, the optimization problem (9.9)–(9.11) has no solution with $M > 0$. An example is shown in Figure 9.4. In this case, we cannot *exactly* separate the two classes. However, as we will see in the next section, we can extend the concept of a separating hyperplane in order to develop a hyperplane that *almost* separates the classes, using a so-called *soft margin*. The generalization of the maximal margin classifier to the non-separable case is known as the *support vector classifier*.

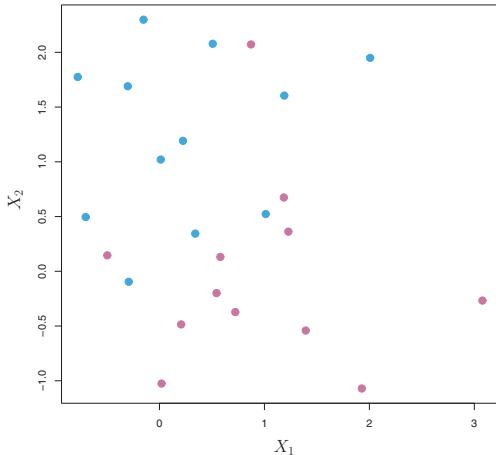


FIGURE 9.4. There are two classes of observations, shown in blue and in purple. In this case, the two classes are not separable by a hyperplane, and so the maximal margin classifier cannot be used.

9.2 Support Vector Classifiers

9.2.1 Overview of the Support Vector Classifier

In Figure 9.4, we see that observations that belong to two classes are not necessarily separable by a hyperplane. In fact, even if a separating hyperplane does exist, then there are instances in which a classifier based on a separating hyperplane might not be desirable. A classifier based on a separating hyperplane will necessarily perfectly classify all of the training observations; this can lead to sensitivity to individual observations. An example is shown in Figure 9.5. The addition of a single observation in the right-hand panel of Figure 9.5 leads to a dramatic change in the maximal margin hyperplane. The resulting maximal margin hyperplane is not satisfactory—for one thing, it has only a tiny margin. This is problematic because as discussed previously, the distance of an observation from the hyperplane can be seen as a measure of our confidence that the observation was correctly classified. Moreover, the fact that the maximal margin hyperplane is extremely sensitive to a change in a single observation suggests that it may have overfit the training data.

In this case, we might be willing to consider a classifier based on a hyperplane that does *not* perfectly separate the two classes, in the interest of

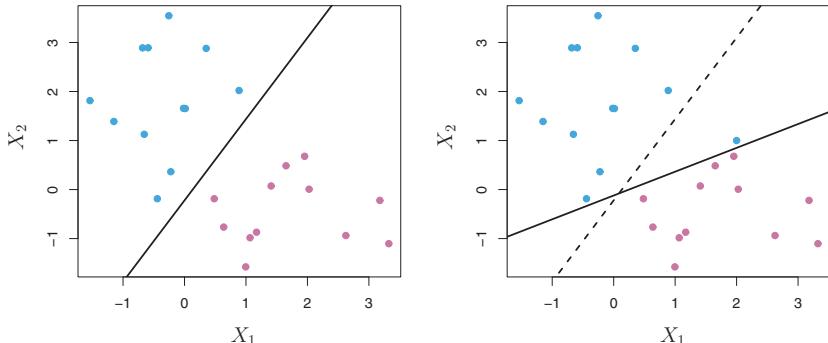


FIGURE 9.5. Left: Two classes of observations are shown in blue and in purple, along with the maximal margin hyperplane. Right: An additional blue observation has been added, leading to a dramatic shift in the maximal margin hyperplane shown as a solid line. The dashed line indicates the maximal margin hyperplane that was obtained in the absence of this additional point.

- Greater robustness to individual observations, and
- Better classification of *most* of the training observations.

That is, it could be worthwhile to misclassify a few training observations in order to do a better job in classifying the remaining observations.

The *support vector classifier*, sometimes called a *soft margin classifier*, does exactly this. Rather than seeking the largest possible margin so that every observation is not only on the correct side of the hyperplane but also on the correct side of the margin, we instead allow some observations to be on the incorrect side of the margin, or even the incorrect side of the hyperplane. (The margin is *soft* because it can be violated by some of the training observations.) An example is shown in the left-hand panel of Figure 9.6. Most of the observations are on the correct side of the margin. However, a small subset of the observations are on the wrong side of the margin.

support
vector
classifier
soft margin
classifier

An observation can be not only on the wrong side of the margin, but also on the wrong side of the hyperplane. In fact, when there is no separating hyperplane, such a situation is inevitable. Observations on the wrong side of the hyperplane correspond to training observations that are misclassified by the support vector classifier. The right-hand panel of Figure 9.6 illustrates such a scenario.

9.2.2 Details of the Support Vector Classifier

The support vector classifier classifies a test observation depending on which side of a hyperplane it lies. The hyperplane is chosen to correctly

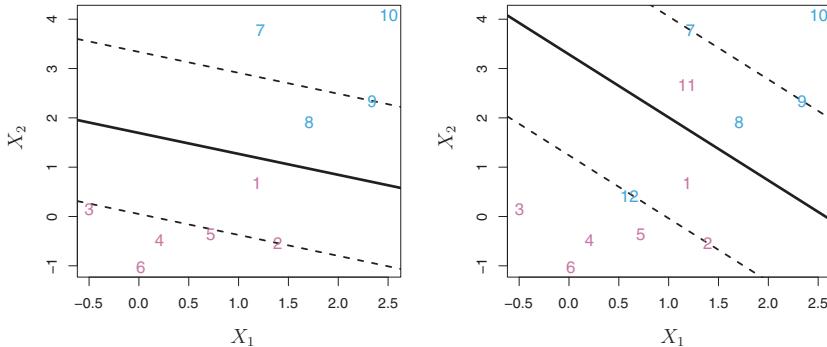


FIGURE 9.6. Left: A support vector classifier was fit to a small data set. The hyperplane is shown as a solid line and the margins are shown as dashed lines. Purple observations: Observations 3, 4, 5, and 6 are on the correct side of the margin, observation 2 is on the margin, and observation 1 is on the wrong side of the margin. Blue observations: Observations 7 and 10 are on the correct side of the margin, observation 9 is on the margin, and observation 8 is on the wrong side of the margin. No observations are on the wrong side of the hyperplane. Right: Same as left panel with two additional points, 11 and 12. These two observations are on the wrong side of the hyperplane and the wrong side of the margin.

separate most of the training observations into the two classes, but may misclassify a few observations. It is the solution to the optimization problem

$$\underset{\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n}{\text{maximize}} \quad M \quad (9.12)$$

$$\text{subject to} \quad \sum_{j=1}^p \beta_j^2 = 1, \quad (9.13)$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), \quad (9.14)$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^n \epsilon_i \leq C, \quad (9.15)$$

where C is a nonnegative tuning parameter. As in (9.11), M is the width of the margin; we seek to make this quantity as large as possible. In (9.14), $\epsilon_1, \dots, \epsilon_n$ are *slack variables* that allow individual observations to be on the wrong side of the margin or the hyperplane; we will explain them in greater detail momentarily. Once we have solved (9.12)–(9.15), we classify a test observation x^* as before, by simply determining on which side of the hyperplane it lies. That is, we classify the test observation based on the sign of $f(x^*) = \beta_0 + \beta_1 x_1^* + \dots + \beta_p x_p^*$.

The problem (9.12)–(9.15) seems complex, but insight into its behavior can be made through a series of simple observations presented below. First of all, the slack variable ϵ_i tells us where the i th observation is located, relative to the hyperplane and relative to the margin. If $\epsilon_i = 0$ then the i th

slack
variable

observation is on the correct side of the margin, as we saw in Section 9.1.4. If $\epsilon_i > 0$ then the i th observation is on the wrong side of the margin, and we say that the i th observation has *violated* the margin. If $\epsilon_i > 1$ then it is on the wrong side of the hyperplane.

We now consider the role of the tuning parameter C . In (9.14), C bounds the sum of the ϵ_i 's, and so it determines the number and severity of the violations to the margin (and to the hyperplane) that we will tolerate. We can think of C as a *budget* for the amount that the margin can be violated by the n observations. If $C = 0$ then there is no budget for violations to the margin, and it must be the case that $\epsilon_1 = \dots = \epsilon_n = 0$, in which case (9.12)–(9.15) simply amounts to the maximal margin hyperplane optimization problem (9.9)–(9.11). (Of course, a maximal margin hyperplane exists only if the two classes are separable.) For $C > 0$ no more than C observations can be on the wrong side of the hyperplane, because if an observation is on the wrong side of the hyperplane then $\epsilon_i > 1$, and (9.14) requires that $\sum_{i=1}^n \epsilon_i \leq C$. As the budget C increases, we become more tolerant of violations to the margin, and so the margin will widen. Conversely, as C decreases, we become less tolerant of violations to the margin and so the margin narrows. An example is shown in Figure 9.7.

In practice, C is treated as a tuning parameter that is generally chosen via cross-validation. As with the tuning parameters that we have seen throughout this book, C controls the bias-variance trade-off of the statistical learning technique. When C is small, we seek narrow margins that are rarely violated; this amounts to a classifier that is highly fit to the data, which may have low bias but high variance. On the other hand, when C is larger, the margin is wider and we allow more violations to it; this amounts to fitting the data less hard and obtaining a classifier that is potentially more biased but may have lower variance.

The optimization problem (9.12)–(9.15) has a very interesting property: it turns out that only observations that either lie on the margin or that violate the margin will affect the hyperplane, and hence the classifier obtained. In other words, an observation that lies strictly on the correct side of the margin does not affect the support vector classifier! Changing the position of that observation would not change the classifier at all, provided that its position remains on the correct side of the margin. Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as *support vectors*. These observations do affect the support vector classifier.

The fact that only support vectors affect the classifier is in line with our previous assertion that C controls the bias-variance trade-off of the support vector classifier. When the tuning parameter C is large, then the margin is wide, many observations violate the margin, and so there are many support vectors. In this case, many observations are involved in determining the hyperplane. The top left panel in Figure 9.7 illustrates this setting: this classifier has low variance (since many observations are support vectors)

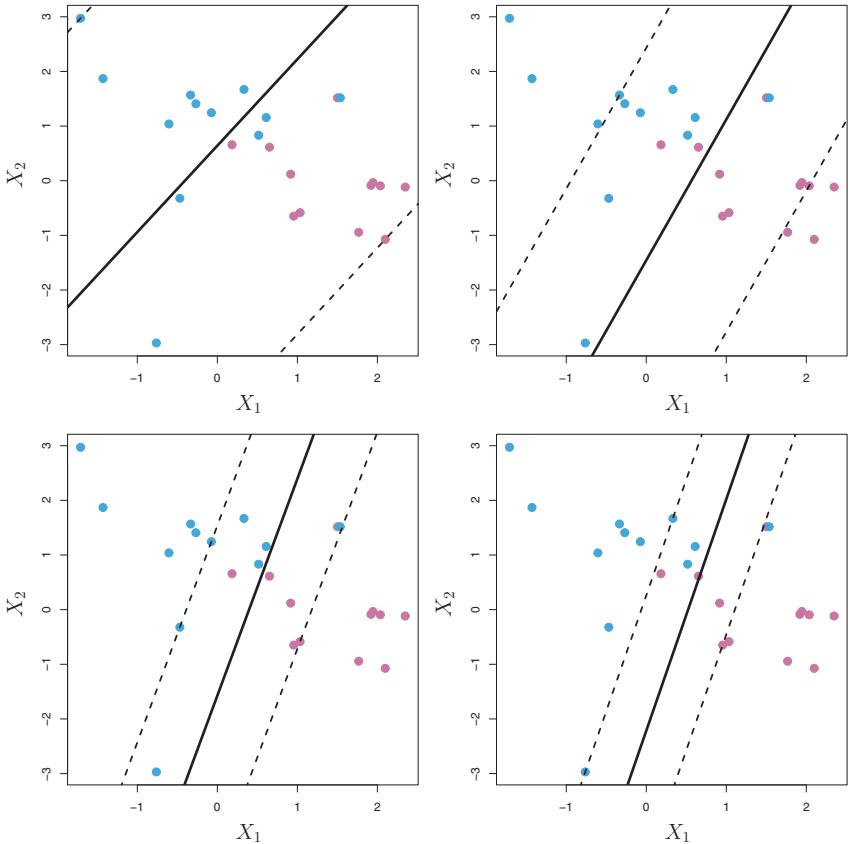


FIGURE 9.7. A support vector classifier was fit using four different values of the tuning parameter C in (9.12)–(9.15). The largest value of C was used in the top left panel, and smaller values were used in the top right, bottom left, and bottom right panels. When C is large, then there is a high tolerance for observations being on the wrong side of the margin, and so the margin will be large. As C decreases, the tolerance for observations being on the wrong side of the margin decreases, and the margin narrows.

but potentially high bias. In contrast, if C is small, then there will be fewer support vectors and hence the resulting classifier will have low bias but high variance. The bottom right panel in Figure 9.7 illustrates this setting, with only eight support vectors.

The fact that the support vector classifier's decision rule is based only on a potentially small subset of the training observations (the support vectors) means that it is quite robust to the behavior of observations that are far away from the hyperplane. This property is distinct from some of the other classification methods that we have seen in preceding chapters, such as linear discriminant analysis. Recall that the LDA classification rule

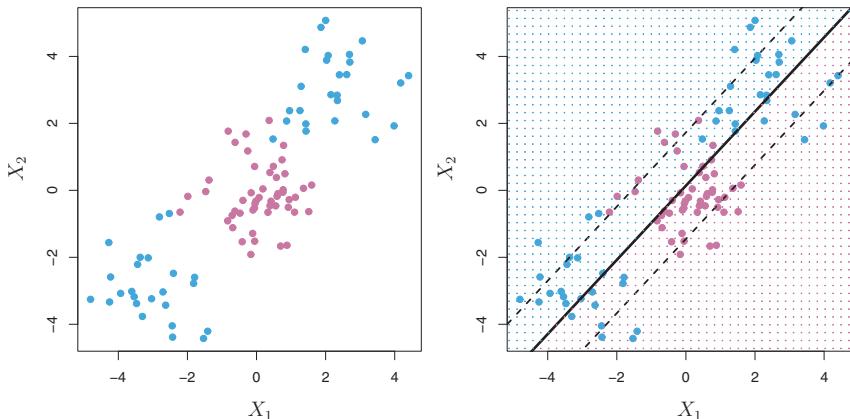


FIGURE 9.8. Left: The observations fall into two classes, with a non-linear boundary between them. Right: The support vector classifier seeks a linear boundary, and consequently performs very poorly.

depends on the mean of *all* of the observations within each class, as well as the within-class covariance matrix computed using *all* of the observations. In contrast, logistic regression, unlike LDA, has very low sensitivity to observations far from the decision boundary. In fact we will see in Section 9.5 that the support vector classifier and logistic regression are closely related.

9.3 Support Vector Machines

We first discuss a general mechanism for converting a linear classifier into one that produces non-linear decision boundaries. We then introduce the support vector machine, which does this in an automatic way.

9.3.1 Classification with Non-linear Decision Boundaries

The support vector classifier is a natural approach for classification in the two-class setting, if the boundary between the two classes is linear. However, in practice we are sometimes faced with non-linear class boundaries. For instance, consider the data in the left-hand panel of Figure 9.8. It is clear that a support vector classifier or any linear classifier will perform poorly here. Indeed, the support vector classifier shown in the right-hand panel of Figure 9.8 is useless here.

In Chapter 7, we are faced with an analogous situation. We see there that the performance of linear regression can suffer when there is a non-linear relationship between the predictors and the outcome. In that case, we consider enlarging the feature space using functions of the predictors,

such as quadratic and cubic terms, in order to address this non-linearity. In the case of the support vector classifier, we could address the problem of possibly non-linear boundaries between classes in a similar way, by enlarging the feature space using quadratic, cubic, and even higher-order polynomial functions of the predictors. For instance, rather than fitting a support vector classifier using p features

$$X_1, X_2, \dots, X_p,$$

we could instead fit a support vector classifier using $2p$ features

$$X_1, X_1^2, X_2, X_2^2, \dots, X_p, X_p^2.$$

Then (9.12)–(9.15) would become

$$\begin{aligned} & \underset{\beta_0, \beta_{11}, \beta_{12}, \dots, \beta_{p1}, \beta_{p2}, \epsilon_1, \dots, \epsilon_n}{\text{maximize}} && M \\ & \text{subject to } y_i \left(\beta_0 + \sum_{j=1}^p \beta_{j1} x_{ij} + \sum_{j=1}^p \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i), \\ & \sum_{i=1}^n \epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^p \sum_{k=1}^2 \beta_{jk}^2 = 1. \end{aligned} \tag{9.16}$$

Why does this lead to a non-linear decision boundary? In the enlarged feature space, the decision boundary that results from (9.16) is in fact linear. But in the original feature space, the decision boundary is of the form $q(x) = 0$, where q is a quadratic polynomial, and its solutions are generally non-linear. One might additionally want to enlarge the feature space with higher-order polynomial terms, or with interaction terms of the form $X_j X_{j'}$ for $j \neq j'$. Alternatively, other functions of the predictors could be considered rather than polynomials. It is not hard to see that there are many possible ways to enlarge the feature space, and that unless we are careful, we could end up with a huge number of features. Then computations would become unmanageable. The support vector machine, which we present next, allows us to enlarge the feature space used by the support vector classifier in a way that leads to efficient computations.

9.3.2 The Support Vector Machine

The *support vector machine* (SVM) is an extension of the support vector classifier that results from enlarging the feature space in a specific way, using *kernels*. We will now discuss this extension, the details of which are somewhat complex and beyond the scope of this book. However, the main idea is described in Section 9.3.1: we may want to enlarge our feature space

support
vector
machine
kernel

in order to accommodate a non-linear boundary between the classes. The kernel approach that we describe here is simply an efficient computational approach for enacting this idea.

We have not discussed exactly how the support vector classifier is computed because the details become somewhat technical. However, it turns out that the solution to the support vector classifier problem (9.12)–(9.15) involves only the *inner products* of the observations (as opposed to the observations themselves). The inner product of two r -vectors a and b is defined as $\langle a, b \rangle = \sum_{i=1}^r a_i b_i$. Thus the inner product of two observations $x_i, x_{i'}$ is given by

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^p x_{ij} x_{i'j}. \quad (9.17)$$

It can be shown that

- The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle, \quad (9.18)$$

where there are n parameters α_i , $i = 1, \dots, n$, one per training observation.

- To estimate the parameters $\alpha_1, \dots, \alpha_n$ and β_0 , all we need are the $\binom{n}{2}$ inner products $\langle x_i, x_{i'} \rangle$ between all pairs of training observations. (The notation $\binom{n}{2}$ means $n(n - 1)/2$, and gives the number of pairs among a set of n items.)

Notice that in (9.18), in order to evaluate the function $f(x)$, we need to compute the inner product between the new point x and each of the training points x_i . However, it turns out that α_i is nonzero only for the support vectors in the solution—that is, if a training observation is not a support vector, then its α_i equals zero. So if \mathcal{S} is the collection of indices of these support points, we can rewrite any solution function of the form (9.18) as

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle, \quad (9.19)$$

which typically involves far fewer terms than in (9.18).²

To summarize, in representing the linear classifier $f(x)$, and in computing its coefficients, all we need are inner products.

Now suppose that every time the inner product (9.17) appears in the representation (9.18), or in a calculation of the solution for the support

²By expanding each of the inner products in (9.19), it is easy to see that $f(x)$ is a linear function of the coordinates of x . Doing so also establishes the correspondence between the α_i and the original parameters β_j .

vector classifier, we replace it with a *generalization* of the inner product of the form

$$K(x_i, x_{i'}), \quad (9.20)$$

where K is some function that we will refer to as a *kernel*. A kernel is a function that quantifies the similarity of two observations. For instance, we could simply take

$$K(x_i, x_{i'}) = \sum_{j=1}^p x_{ij} x_{i'j}, \quad (9.21)$$

which would just give us back the support vector classifier. Equation 9.21 is known as a *linear kernel* because the support vector classifier is linear in the features; the linear kernel essentially quantifies the similarity of a pair of observations using Pearson (standard) correlation. But one could instead choose another form for (9.20). For instance, one could replace every instance of $\sum_{j=1}^p x_{ij} x_{i'j}$ with the quantity

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^p x_{ij} x_{i'j})^d. \quad (9.22)$$

This is known as a *polynomial kernel* of degree d , where d is a positive integer. Using such a kernel with $d > 1$, instead of the standard linear kernel (9.21), in the support vector classifier algorithm leads to a much more flexible decision boundary. It essentially amounts to fitting a support vector classifier in a higher-dimensional space involving polynomials of degree d , rather than in the original feature space. When the support vector classifier is combined with a non-linear kernel such as (9.22), the resulting classifier is known as a support vector machine. Note that in this case the (non-linear) function has the form

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i). \quad (9.23)$$

The left-hand panel of Figure 9.9 shows an example of an SVM with a polynomial kernel applied to the non-linear data from Figure 9.8. The fit is a substantial improvement over the linear support vector classifier. When $d = 1$, then the SVM reduces to the support vector classifier seen earlier in this chapter.

The polynomial kernel shown in (9.22) is one example of a possible non-linear kernel, but alternatives abound. Another popular choice is the *radial kernel*, which takes the form

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2). \quad (9.24)$$

kernel

polynomial kernel

radial kernel

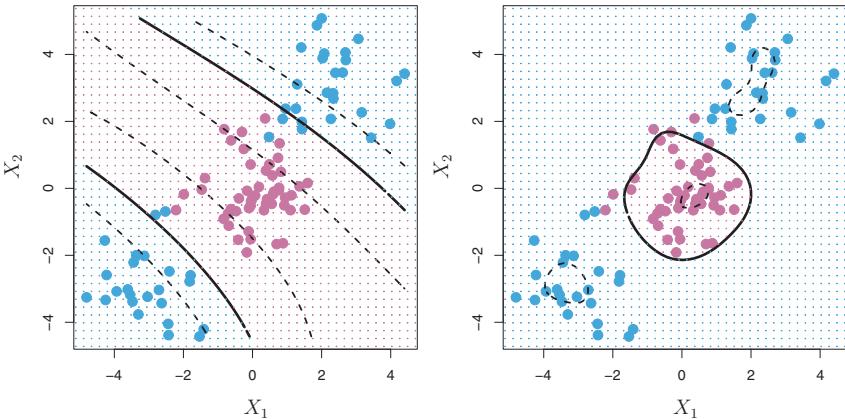


FIGURE 9.9. Left: An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule. Right: An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.

In (9.24), γ is a positive constant. The right-hand panel of Figure 9.9 shows an example of an SVM with a radial kernel on this non-linear data; it also does a good job in separating the two classes.

How does the radial kernel (9.24) actually work? If a given test observation $x^* = (x_1^* \dots x_p^*)^T$ is far from a training observation x_i in terms of Euclidean distance, then $\sum_{j=1}^p (x_j^* - x_{ij})^2$ will be large, and so $K(x_i, x_i') = \exp(-\gamma \sum_{j=1}^p (x_j^* - x_{ij})^2)$ will be very tiny. This means that in (9.23), x_i will play virtually no role in $f(x^*)$. Recall that the predicted class label for the test observation x^* is based on the sign of $f(x^*)$. In other words, training observations that are far from x^* will play essentially no role in the predicted class label for x^* . This means that the radial kernel has very *local* behavior, in the sense that only nearby training observations have an effect on the class label of a test observation.

What is the advantage of using a kernel rather than simply enlarging the feature space using functions of the original features, as in (9.16)? One advantage is computational, and it amounts to the fact that using kernels, one need only compute $K(x_i, x_{i'})$ for all $\binom{n}{2}$ distinct pairs i, i' . This can be done without explicitly working in the enlarged feature space. This is important because in many applications of SVMs, the enlarged feature space is so large that computations are intractable. For some kernels, such as the radial kernel (9.24), the feature space is *implicit* and infinite-dimensional, so we could never do the computations there anyway!

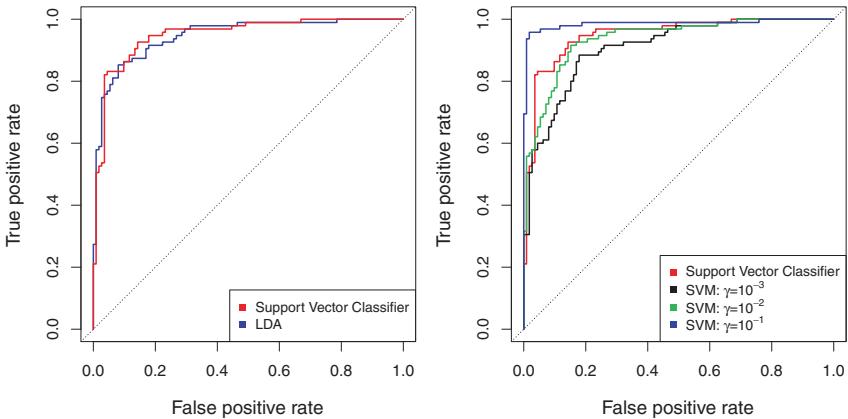


FIGURE 9.10. ROC curves for the `Heart` data training set. Left: The support vector classifier and LDA are compared. Right: The support vector classifier is compared to an SVM using a radial basis kernel with $\gamma = 10^{-3}$, 10^{-2} , and 10^{-1} .

9.3.3 An Application to the Heart Disease Data

In Chapter 8 we apply decision trees and related methods to the `Heart` data. The aim is to use 13 predictors such as `Age`, `Sex`, and `Chol` in order to predict whether an individual has heart disease. We now investigate how an SVM compares to LDA on this data. After removing 6 missing observations, the data consist of 297 subjects, which we randomly split into 207 training and 90 test observations.

We first fit LDA and the support vector classifier to the training data. Note that the support vector classifier is equivalent to a SVM using a polynomial kernel of degree $d = 1$. The left-hand panel of Figure 9.10 displays ROC curves (described in Section 4.4.3) for the training set predictions for both LDA and the support vector classifier. Both classifiers compute scores of the form $\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \dots + \hat{\beta}_p X_p$ for each observation. For any given cutoff t , we classify observations into the *heart disease* or *no heart disease* categories depending on whether $\hat{f}(X) < t$ or $\hat{f}(X) \geq t$. The ROC curve is obtained by forming these predictions and computing the false positive and true positive rates for a range of values of t . An optimal classifier will hug the top left corner of the ROC plot. In this instance LDA and the support vector classifier both perform well, though there is a suggestion that the support vector classifier may be slightly superior.

The right-hand panel of Figure 9.10 displays ROC curves for SVMs using a radial kernel, with various values of γ . As γ increases and the fit becomes more non-linear, the ROC curves improve. Using $\gamma = 10^{-1}$ appears to give an almost perfect ROC curve. However, these curves represent training error rates, which can be misleading in terms of performance on new test data. Figure 9.11 displays ROC curves computed on the 90 test observa-

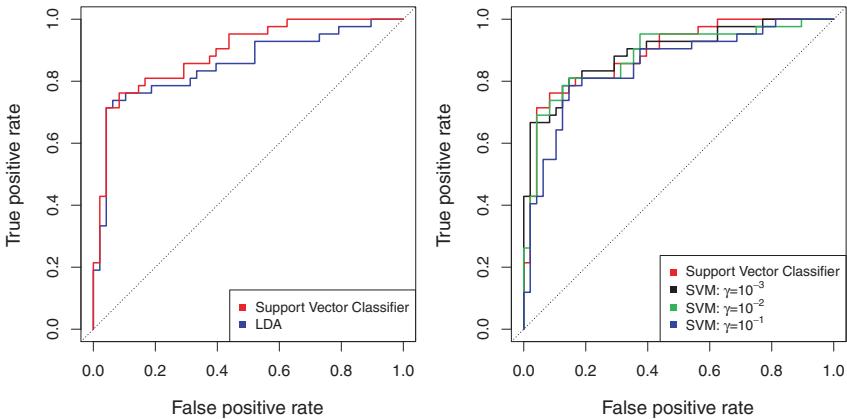


FIGURE 9.11. ROC curves for the test set of the [Heart](#) data. Left: The support vector classifier and LDA are compared. Right: The support vector classifier is compared to an SVM using a radial basis kernel with $\gamma = 10^{-3}$, 10^{-2} , and 10^{-1} .

tions. We observe some differences from the training ROC curves. In the left-hand panel of Figure 9.11, the support vector classifier appears to have a small advantage over LDA (although these differences are not statistically significant). In the right-hand panel, the SVM using $\gamma = 10^{-1}$, which showed the best results on the training data, produces the worst estimates on the test data. This is once again evidence that while a more flexible method will often produce lower training error rates, this does not necessarily lead to improved performance on test data. The SVMs with $\gamma = 10^{-2}$ and $\gamma = 10^{-3}$ perform comparably to the support vector classifier, and all three outperform the SVM with $\gamma = 10^{-1}$.

9.4 SVMs with More than Two Classes

So far, our discussion has been limited to the case of binary classification: that is, classification in the two-class setting. How can we extend SVMs to the more general case where we have some arbitrary number of classes? It turns out that the concept of separating hyperplanes upon which SVMs are based does not lend itself naturally to more than two classes. Though a number of proposals for extending SVMs to the K -class case have been made, the two most popular are the *one-versus-one* and *one-versus-all* approaches. We briefly discuss those two approaches here.

9.4.1 One-Versus-One Classification

Suppose that we would like to perform classification using SVMs, and there are $K > 2$ classes. A *one-versus-one* or *all-pairs* approach constructs $\binom{K}{2}$ one-versus-one

SVMs, each of which compares a pair of classes. For example, one such SVM might compare the k th class, coded as $+1$, to the k' th class, coded as -1 . We classify a test observation using each of the $\binom{K}{2}$ classifiers, and we tally the number of times that the test observation is assigned to each of the K classes. The final classification is performed by assigning the test observation to the class to which it was most frequently assigned in these $\binom{K}{2}$ pairwise classifications.

9.4.2 One-Versus-All Classification

The *one-versus-all* approach is an alternative procedure for applying SVMs in the case of $K > 2$ classes. We fit K SVMs, each time comparing one of the K classes to the remaining $K - 1$ classes. Let $\beta_{0k}, \beta_{1k}, \dots, \beta_{pk}$ denote the parameters that result from fitting an SVM comparing the k th class (coded as $+1$) to the others (coded as -1). Let x^* denote a test observation. We assign the observation to the class for which $\beta_{0k} + \beta_{1k}x_1^* + \beta_{2k}x_2^* + \dots + \beta_{pk}x_p^*$ is largest, as this amounts to a high level of confidence that the test observation belongs to the k th class rather than to any of the other classes.

9.5 Relationship to Logistic Regression

When SVMs were first introduced in the mid-1990s, they made quite a splash in the statistical and machine learning communities. This was due in part to their good performance, good marketing, and also to the fact that the underlying approach seemed both novel and mysterious. The idea of finding a hyperplane that separates the data as well as possible, while allowing some violations to this separation, seemed distinctly different from classical approaches for classification, such as logistic regression and linear discriminant analysis. Moreover, the idea of using a kernel to expand the feature space in order to accommodate non-linear class boundaries appeared to be a unique and valuable characteristic.

However, since that time, deep connections between SVMs and other more classical statistical methods have emerged. It turns out that one can rewrite the criterion (9.12)–(9.15) for fitting the support vector classifier $f(X) = \beta_0 + \beta_1X_1 + \dots + \beta_pX_p$ as

$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{minimize}} \left\{ \sum_{i=1}^n \max [0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^p \beta_j^2 \right\}, \quad (9.25)$$

where λ is a nonnegative tuning parameter. When λ is large then β_1, \dots, β_p are small, more violations to the margin are tolerated, and a low-variance but high-bias classifier will result. When λ is small then few violations to the margin will occur; this amounts to a high-variance but low-bias classifier. Thus, a small value of λ in (9.25) amounts to a small value of C in (9.15). Note that the $\lambda \sum_{j=1}^p \beta_j^2$ term in (9.25) is the ridge penalty term from Section 6.2.1, and plays a similar role in controlling the bias-variance trade-off for the support vector classifier.

Now (9.25) takes the “Loss + Penalty” form that we have seen repeatedly throughout this book:

$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{minimize}} \{L(\mathbf{X}, \mathbf{y}, \beta) + \lambda P(\beta)\}. \quad (9.26)$$

In (9.26), $L(\mathbf{X}, \mathbf{y}, \beta)$ is some loss function quantifying the extent to which the model, parametrized by β , fits the data (\mathbf{X}, \mathbf{y}) , and $P(\beta)$ is a penalty function on the parameter vector β whose effect is controlled by a nonnegative tuning parameter λ . For instance, ridge regression and the lasso both take this form with

$$L(\mathbf{X}, \mathbf{y}, \beta) = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j \right)^2$$

and with $P(\beta) = \sum_{j=1}^p \beta_j^2$ for ridge regression and $P(\beta) = \sum_{j=1}^p |\beta_j|$ for the lasso. In the case of (9.25) the loss function instead takes the form

$$L(\mathbf{X}, \mathbf{y}, \beta) = \sum_{i=1}^n \max [0, 1 - y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip})].$$

This is known as *hinge loss*, and is depicted in Figure 9.12. However, it turns out that the hinge loss function is closely related to the loss function used in logistic regression, also shown in Figure 9.12.

An interesting characteristic of the support vector classifier is that only support vectors play a role in the classifier obtained; observations on the correct side of the margin do not affect it. This is due to the fact that the loss function shown in Figure 9.12 is exactly zero for observations for which $y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq 1$; these correspond to observations that are on the correct side of the margin.³ In contrast, the loss function for logistic regression shown in Figure 9.12 is not exactly zero anywhere. But it is very small for observations that are far from the decision boundary. Due to the similarities between their loss functions, logistic regression and the support vector classifier often give very similar results. When the classes are well separated, SVMs tend to behave better than logistic regression; in more overlapping regimes, logistic regression is often preferred.

³With this hinge-loss + penalty representation, the margin corresponds to the value one, and the width of the margin is determined by $\sum \beta_j^2$.

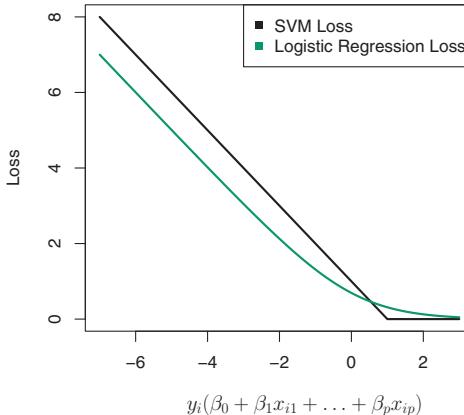


FIGURE 9.12. The SVM and logistic regression loss functions are compared, as a function of $y_i(\beta_0 + \beta_1x_{i1} + \dots + \beta_px_{ip})$. When $y_i(\beta_0 + \beta_1x_{i1} + \dots + \beta_px_{ip})$ is greater than 1, then the SVM loss is zero, since this corresponds to an observation that is on the correct side of the margin. Overall, the two loss functions have quite similar behavior.

When the support vector classifier and SVM were first introduced, it was thought that the tuning parameter C in (9.15) was an unimportant “nuisance” parameter that could be set to some default value, like 1. However, the “Loss + Penalty” formulation (9.25) for the support vector classifier indicates that this is not the case. The choice of tuning parameter is very important and determines the extent to which the model underfits or overfits the data, as illustrated, for example, in Figure 9.7.

We have established that the support vector classifier is closely related to logistic regression and other preexisting statistical methods. Is the SVM unique in its use of kernels to enlarge the feature space to accommodate non-linear class boundaries? The answer to this question is “no”. We could just as well perform logistic regression or many of the other classification methods seen in this book using non-linear kernels; this is closely related to some of the non-linear approaches seen in Chapter 7. However, for historical reasons, the use of non-linear kernels is much more widespread in the context of SVMs than in the context of logistic regression or other methods.

Though we have not addressed it here, there is in fact an extension of the SVM for regression (i.e. for a quantitative rather than a qualitative response), called *support vector regression*. In Chapter 3, we saw that least squares regression seeks coefficients $\beta_0, \beta_1, \dots, \beta_p$ such that the sum of squared residuals is as small as possible. (Recall from Chapter 3 that residuals are defined as $y_i - \beta_0 - \beta_1x_{i1} - \dots - \beta_px_{ip}$.) Support vector regression instead seeks coefficients that minimize a different type of loss, where only residuals larger in absolute value than some positive constant

support
vector
regression

contribute to the loss function. This is an extension of the margin used in support vector classifiers to the regression setting.

9.6 Lab: Support Vector Machines

We use the `e1071` library in `R` to demonstrate the support vector classifier and the SVM. Another option is the `LiblineaR` library, which is useful for very large linear problems.

9.6.1 Support Vector Classifier

The `e1071` library contains implementations for a number of statistical learning methods. In particular, the `svm()` function can be used to fit a support vector classifier when the argument `kernel="linear"` is used. This function uses a slightly different formulation from (9.14) and (9.25) for the support vector classifier. A `cost` argument allows us to specify the cost of a violation to the margin. When the `cost` argument is small, then the margins will be wide and many support vectors will be on the margin or will violate the margin. When the `cost` argument is large, then the margins will be narrow and there will be few support vectors on the margin or violating the margin.

We now use the `svm()` function to fit the support vector classifier for a given value of the `cost` parameter. Here we demonstrate the use of this function on a two-dimensional example so that we can plot the resulting decision boundary. We begin by generating the observations, which belong to two classes, and checking whether the classes are linearly separable.

```
> set.seed(1)
> x=matrix(rnorm(20*2), ncol=2)
> y=c(rep(-1,10), rep(1,10))
> x[y==1,]=x[y==1,] + 1
> plot(x, col=(3-y))
```

They are not. Next, we fit the support vector classifier. Note that in order for the `svm()` function to perform classification (as opposed to SVM-based regression), we must encode the response as a factor variable. We now create a data frame with the response coded as a factor.

```
> dat=data.frame(x=x, y=as.factor(y))
> library(e1071)
> svmfit=svm(y~., data=dat, kernel="linear", cost=10,
  scale=FALSE)
```

We see that using `cost=10` yields two test set errors on this data.

9.7 Exercises

Conceptual

1. This problem involves hyperplanes in two dimensions.
 - (a) Sketch the hyperplane $1 + 3X_1 - X_2 = 0$. Indicate the set of points for which $1 + 3X_1 - X_2 > 0$, as well as the set of points for which $1 + 3X_1 - X_2 < 0$.
 - (b) On the same plot, sketch the hyperplane $-2 + X_1 + 2X_2 = 0$. Indicate the set of points for which $-2 + X_1 + 2X_2 > 0$, as well as the set of points for which $-2 + X_1 + 2X_2 < 0$.
2. We have seen that in $p = 2$ dimensions, a linear decision boundary takes the form $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$. We now investigate a non-linear decision boundary.
 - (a) Sketch the curve

$$(1 + X_1)^2 + (2 - X_2)^2 = 4.$$
 - (b) On your sketch, indicate the set of points for which

$$(1 + X_1)^2 + (2 - X_2)^2 > 4,$$
 as well as the set of points for which

$$(1 + X_1)^2 + (2 - X_2)^2 \leq 4.$$
 - (c) Suppose that a classifier assigns an observation to the blue class if

$$(1 + X_1)^2 + (2 - X_2)^2 > 4,$$
 and to the red class otherwise. To what class is the observation $(0, 0)$ classified? $(-1, 1)$? $(2, 2)$? $(3, 8)$?
 - (d) Argue that while the decision boundary in (c) is not linear in terms of X_1 and X_2 , it is linear in terms of X_1 , X_1^2 , X_2 , and X_2^2 .
3. Here we explore the maximal margin classifier on a toy data set.
 - (a) We are given $n = 7$ observations in $p = 2$ dimensions. For each observation, there is an associated class label.

Obs.	X_1	X_2	Y
1	3	4	Red
2	2	2	Red
3	4	4	Red
4	1	4	Red
5	2	1	Blue
6	4	3	Blue
7	4	1	Blue

Sketch the observations.

- (b) Sketch the optimal separating hyperplane, and provide the equation for this hyperplane (of the form (9.1)).
- (c) Describe the classification rule for the maximal margin classifier. It should be something along the lines of “Classify to Red if $\beta_0 + \beta_1 X_1 + \beta_2 X_2 > 0$, and classify to Blue otherwise.” Provide the values for β_0 , β_1 , and β_2 .
- (d) On your sketch, indicate the margin for the maximal margin hyperplane.
- (e) Indicate the support vectors for the maximal margin classifier.
- (f) Argue that a slight movement of the seventh observation would not affect the maximal margin hyperplane.
- (g) Sketch a hyperplane that is *not* the optimal separating hyperplane, and provide the equation for this hyperplane.
- (h) Draw an additional observation on the plot so that the two classes are no longer separable by a hyperplane.

Applied

- 4. Generate a simulated two-class data set with 100 observations and two features in which there is a visible but non-linear separation between the two classes. Show that in this setting, a support vector machine with a polynomial kernel (with degree greater than 1) or a radial kernel will outperform a support vector classifier on the training data. Which technique performs best on the test data? Make plots and report training and test error rates in order to back up your assertions.
- 5. We have seen that we can fit an SVM with a non-linear kernel in order to perform classification using a non-linear decision boundary. We will now see that we can also obtain a non-linear decision boundary by performing logistic regression using non-linear transformations of the features.

12

Support Vector Machines and Flexible Discriminants

12.1 Introduction

In this chapter we describe generalizations of linear decision boundaries for classification. Optimal separating hyperplanes are introduced in Chapter 4 for the case when two classes are linearly separable. Here we cover extensions to the nonseparable case, where the classes overlap. These techniques are then generalized to what is known as the *support vector machine*, which produces nonlinear boundaries by constructing a linear boundary in a large, transformed version of the feature space. The second set of methods generalize Fisher’s linear discriminant analysis (LDA). The generalizations include *flexible discriminant analysis* which facilitates construction of nonlinear boundaries in a manner very similar to the support vector machines, *penalized discriminant analysis* for problems such as signal and image classification where the large number of features are highly correlated, and *mixture discriminant analysis* for irregularly shaped classes.

12.2 The Support Vector Classifier

In Chapter 4 we discussed a technique for constructing an *optimal* separating hyperplane between two perfectly separated classes. We review this and generalize to the nonseparable case, where the classes may not be separable by a linear boundary.

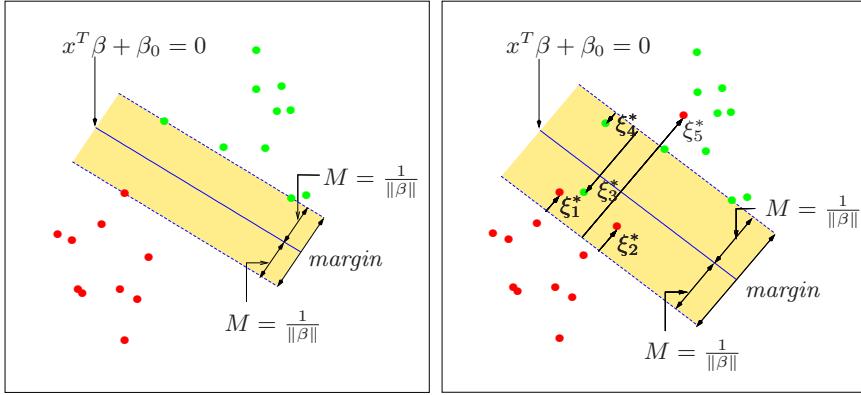


FIGURE 12.1. Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width $2M = 2/\|\beta\|$. The right panel shows the nonseparable (overlap) case. The points labeled ξ_j^* are on the wrong side of their margin by an amount $\xi_j^* = M\xi_j$; points on the correct side have $\xi_j^* = 0$. The margin is maximized subject to a total budget $\sum \xi_i \leq \text{constant}$. Hence $\sum \xi_j^*$ is the total distance of points on the wrong side of their margin.

Our training data consists of N pairs $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$, with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$. Define a hyperplane by

$$\{x : f(x) = x^T \beta + \beta_0 = 0\}, \quad (12.1)$$

where β is a unit vector: $\|\beta\| = 1$. A classification rule induced by $f(x)$ is

$$G(x) = \text{sign}[x^T \beta + \beta_0]. \quad (12.2)$$

The geometry of hyperplanes is reviewed in Section 4.5, where we show that $f(x)$ in (12.1) gives the signed distance from a point x to the hyperplane $f(x) = x^T \beta + \beta_0 = 0$. Since the classes are separable, we can find a function $f(x) = x^T \beta + \beta_0$ with $y_i f(x_i) > 0 \forall i$. Hence we are able to find the hyperplane that creates the biggest *margin* between the training points for class 1 and -1 (see Figure 12.1). The optimization problem

$$\begin{aligned} & \max_{\beta, \beta_0, \|\beta\|=1} M \\ & \text{subject to } y_i(x_i^T \beta + \beta_0) \geq M, \quad i = 1, \dots, N, \end{aligned} \quad (12.3)$$

captures this concept. The band in the figure is M units away from the hyperplane on either side, and hence $2M$ units wide. It is called the *margin*.

We showed that this problem can be more conveniently rephrased as

$$\begin{aligned} & \min_{\beta, \beta_0} \|\beta\| \\ & \text{subject to } y_i(x_i^T \beta + \beta_0) \geq 1, \quad i = 1, \dots, N, \end{aligned} \quad (12.4)$$

where we have dropped the norm constraint on β . Note that $M = 1/\|\beta\|$. Expression (12.4) is the usual way of writing the support vector criterion for separated data. This is a convex optimization problem (quadratic criterion, linear inequality constraints), and the solution is characterized in Section 4.5.2.

Suppose now that the classes overlap in feature space. One way to deal with the overlap is to still maximize M , but allow for some points to be on the wrong side of the margin. Define the slack variables $\xi = (\xi_1, \xi_2, \dots, \xi_N)$. There are two natural ways to modify the constraint in (12.3):

$$y_i(x_i^T \beta + \beta_0) \geq M - \xi_i, \quad (12.5)$$

or

$$y_i(x_i^T \beta + \beta_0) \geq M(1 - \xi_i), \quad (12.6)$$

$\forall i, \xi_i \geq 0, \sum_{i=1}^N \xi_i \leq \text{constant}$. The two choices lead to different solutions. The first choice seems more natural, since it measures overlap in actual distance from the margin; the second choice measures the overlap in relative distance, which changes with the width of the margin M . However, the first choice results in a nonconvex optimization problem, while the second is convex; thus (12.6) leads to the “standard” support vector classifier, which we use from here on.

Here is the idea of the formulation. The value ξ_i in the constraint $y_i(x_i^T \beta + \beta_0) \geq M(1 - \xi_i)$ is the proportional amount by which the prediction $f(x_i) = x_i^T \beta + \beta_0$ is on the wrong side of its margin. Hence by bounding the sum $\sum \xi_i$, we bound the total proportional amount by which predictions fall on the wrong side of their margin. Misclassifications occur when $\xi_i > 1$, so bounding $\sum \xi_i$ at a value K say, bounds the total number of training misclassifications at K .

As in (4.48) in Section 4.5.2, we can drop the norm constraint on β , define $M = 1/\|\beta\|$, and write (12.4) in the equivalent form

$$\min \|\beta\| \quad \text{subject to} \quad \begin{cases} y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i \quad \forall i, \\ \xi_i \geq 0, \quad \sum \xi_i \leq \text{constant}. \end{cases} \quad (12.7)$$

This is the usual way the support vector classifier is defined for the non-separable case. However we find confusing the presence of the fixed scale “1” in the constraint $y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i$, and prefer to start with (12.6). The right panel of Figure 12.1 illustrates this overlapping case.

By the nature of the criterion (12.7), we see that points well inside their class boundary do not play a big role in shaping the boundary. This seems like an attractive property, and one that differentiates it from linear discriminant analysis (Section 4.3). In LDA, the decision boundary is determined by the covariance of the class distributions and the positions of the class centroids. We will see in Section 12.3.3 that logistic regression is more similar to the support vector classifier in this regard.

12.2.1 Computing the Support Vector Classifier



The problem (12.7) is quadratic with linear inequality constraints, hence it is a convex optimization problem. We describe a quadratic programming solution using Lagrange multipliers. Computationally it is convenient to re-express (12.7) in the equivalent form

$$\min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^N \xi_i \quad (12.8)$$

$$\text{subject to } \xi_i \geq 0, y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i \forall i,$$

where the “cost” parameter C replaces the constant in (12.7); the separable case corresponds to $C = \infty$.

The Lagrange (primal) function is

$$L_P = \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i [y_i(x_i^T \beta + \beta_0) - (1 - \xi_i)] - \sum_{i=1}^N \mu_i \xi_i, \quad (12.9)$$

which we minimize w.r.t β , β_0 and ξ_i . Setting the respective derivatives to zero, we get

$$\beta = \sum_{i=1}^N \alpha_i y_i x_i, \quad (12.10)$$

$$0 = \sum_{i=1}^N \alpha_i y_i, \quad (12.11)$$

$$\alpha_i = C - \mu_i, \quad \forall i, \quad (12.12)$$

as well as the positivity constraints $\alpha_i, \mu_i, \xi_i \geq 0 \forall i$. By substituting (12.10)–(12.12) into (12.9), we obtain the Lagrangian (Wolfe) dual objective function

$$L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{i'=1}^N \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'}, \quad (12.13)$$

which gives a lower bound on the objective function (12.8) for any feasible point. We maximize L_D subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^N \alpha_i y_i = 0$. In addition to (12.10)–(12.12), the Karush–Kuhn–Tucker conditions include the constraints

$$\alpha_i [y_i(x_i^T \beta + \beta_0) - (1 - \xi_i)] = 0, \quad (12.14)$$

$$\mu_i \xi_i = 0, \quad (12.15)$$

$$y_i(x_i^T \beta + \beta_0) - (1 - \xi_i) \geq 0, \quad (12.16)$$

for $i = 1, \dots, N$. Together these equations (12.10)–(12.16) uniquely characterize the solution to the primal and dual problem.

From (12.10) we see that the solution for β has the form

$$\hat{\beta} = \sum_{i=1}^N \hat{\alpha}_i y_i x_i, \quad (12.17)$$

with nonzero coefficients $\hat{\alpha}_i$ only for those observations i for which the constraints in (12.16) are exactly met (due to (12.14)). These observations are called the *support vectors*, since $\hat{\beta}$ is represented in terms of them alone. Among these support points, some will lie on the edge of the margin ($\hat{\xi}_i = 0$), and hence from (12.15) and (12.12) will be characterized by $0 < \hat{\alpha}_i < C$; the remainder ($\hat{\xi}_i > 0$) have $\hat{\alpha}_i = C$. From (12.14) we can see that any of these margin points ($0 < \hat{\alpha}_i, \hat{\xi}_i = 0$) can be used to solve for β_0 , and we typically use an average of all the solutions for numerical stability.

Maximizing the dual (12.13) is a simpler convex quadratic programming problem than the primal (12.9), and can be solved with standard techniques (Murray et al., 1981, for example).

Given the solutions $\hat{\beta}_0$ and $\hat{\beta}$, the decision function can be written as

$$\begin{aligned} \hat{G}(x) &= \text{sign}[\hat{f}(x)] \\ &= \text{sign}[x^T \hat{\beta} + \hat{\beta}_0]. \end{aligned} \quad (12.18)$$

The tuning parameter of this procedure is the cost parameter C .

12.2.2 Mixture Example (Continued)

Figure 12.2 shows the support vector boundary for the mixture example of Figure 2.5 on page 21, with two overlapping classes, for two different values of the cost parameter C . The classifiers are rather similar in their performance. Points on the wrong side of the boundary are support vectors. In addition, points on the correct side of the boundary but close to it (in the margin), are also support vectors. The margin is larger for $C = 0.01$ than it is for $C = 10,000$. Hence larger values of C focus attention more on (correctly classified) points near the decision boundary, while smaller values involve data further away. Either way, misclassified points are given weight, no matter how far away. In this example the procedure is not very sensitive to choices of C , because of the rigidity of a linear boundary.

The optimal value for C can be estimated by cross-validation, as discussed in Chapter 7. Interestingly, the leave-one-out cross-validation error can be bounded above by the proportion of support points in the data. The reason is that leaving out an observation that is not a support vector will not change the solution. Hence these observations, being classified correctly by the original boundary, will be classified correctly in the cross-validation process. However this bound tends to be too high, and not generally useful for choosing C (62% and 85%, respectively, in our examples).

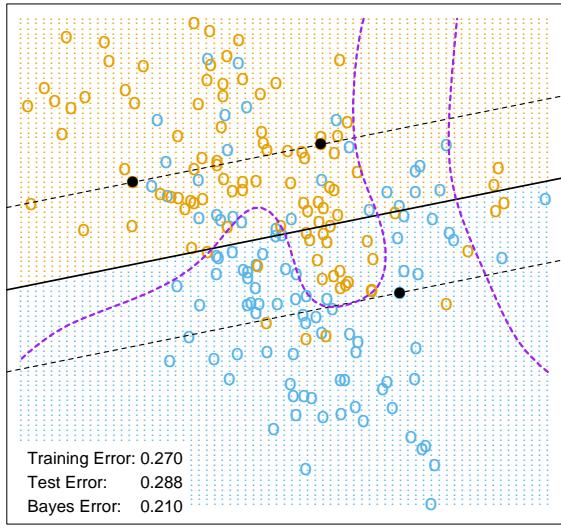
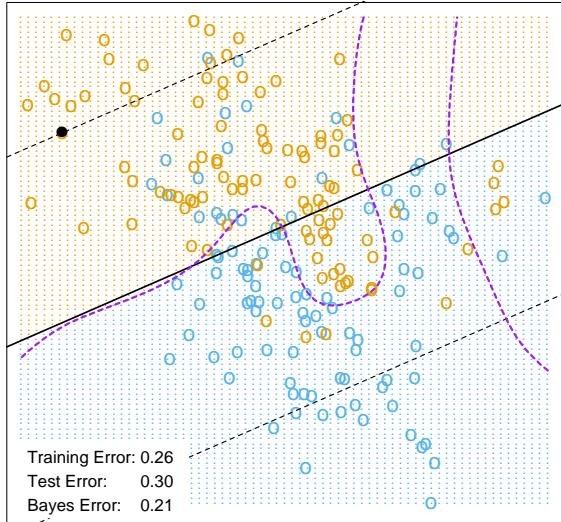
 $C = 10000$  $C = 0.01$

FIGURE 12.2. The linear support vector boundary for the mixture data example with two overlapping classes, for two different values of C . The broken lines indicate the margins, where $f(x) = \pm 1$. The support points ($\alpha_i > 0$) are all the points on the wrong side of their margin. The black solid dots are those support points falling exactly on the margin ($\xi_i = 0, \alpha_i > 0$). In the upper panel 62% of the observations are support points, while in the lower panel 85% are. The broken purple curve in the background is the Bayes decision boundary.

12.3 Support Vector Machines and Kernels

The support vector classifier described so far finds linear boundaries in the input feature space. As with other linear methods, we can make the procedure more flexible by enlarging the feature space using basis expansions such as polynomials or splines (Chapter 5). Generally linear boundaries in the enlarged space achieve better training-class separation, and translate to nonlinear boundaries in the original space. Once the basis functions $h_m(x)$, $m = 1, \dots, M$ are selected, the procedure is the same as before. We fit the SV classifier using input features $h(x_i) = (h_1(x_i), h_2(x_i), \dots, h_M(x_i))$, $i = 1, \dots, N$, and produce the (nonlinear) function $\hat{f}(x) = h(x)^T \hat{\beta} + \hat{\beta}_0$. The classifier is $\hat{G}(x) = \text{sign}(\hat{f}(x))$ as before.

The *support vector machine* classifier is an extension of this idea, where the dimension of the enlarged space is allowed to get very large, infinite in some cases. It might seem that the computations would become prohibitive. It would also seem that with sufficient basis functions, the data would be separable, and overfitting would occur. We first show how the SVM technology deals with these issues. We then see that in fact the SVM classifier is solving a function-fitting problem using a particular criterion and form of regularization, and is part of a much bigger class of problems that includes the smoothing splines of Chapter 5. The reader may wish to consult Section 5.8, which provides background material and overlaps somewhat with the next two sections.

12.3.1 Computing the SVM for Classification

We can represent the optimization problem (12.9) and its solution in a special way that only involves the input features via inner products. We do this directly for the transformed feature vectors $h(x_i)$. We then see that for particular choices of h , these inner products can be computed very cheaply.

The Lagrange dual function (12.13) has the form

$$L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{i'=1}^N \alpha_i \alpha_{i'} y_i y_{i'} \langle h(x_i), h(x_{i'}) \rangle. \quad (12.19)$$

From (12.10) we see that the solution function $f(x)$ can be written

$$\begin{aligned} f(x) &= h(x)^T \beta + \beta_0 \\ &= \sum_{i=1}^N \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0. \end{aligned} \quad (12.20)$$

As before, given α_i , β_0 can be determined by solving $y_i f(x_i) = 1$ in (12.20) for any (or all) x_i for which $0 < \alpha_i < C$.

So both (12.19) and (12.20) involve $h(x)$ only through inner products. In fact, we need not specify the transformation $h(x)$ at all, but require only knowledge of the kernel function

$$K(x, x') = \langle h(x), h(x') \rangle \quad (12.21)$$

that computes inner products in the transformed space. K should be a symmetric positive (semi-) definite function; see Section 5.8.1.

Three popular choices for K in the SVM literature are

$$\begin{aligned} \text{dth-Degree polynomial: } & K(x, x') = (1 + \langle x, x' \rangle)^d, \\ \text{Radial basis: } & K(x, x') = \exp(-\gamma \|x - x'\|^2), \\ \text{Neural network: } & K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2). \end{aligned} \quad (12.22)$$

Consider for example a feature space with two inputs X_1 and X_2 , and a polynomial kernel of degree 2. Then

$$\begin{aligned} K(X, X') &= (1 + \langle X, X' \rangle)^2 \\ &= (1 + X_1 X'_1 + X_2 X'_2)^2 \\ &= 1 + 2X_1 X'_1 + 2X_2 X'_2 + (X_1 X'_1)^2 + (X_2 X'_2)^2 + 2X_1 X'_1 X_2 X'_2. \end{aligned} \quad (12.23)$$

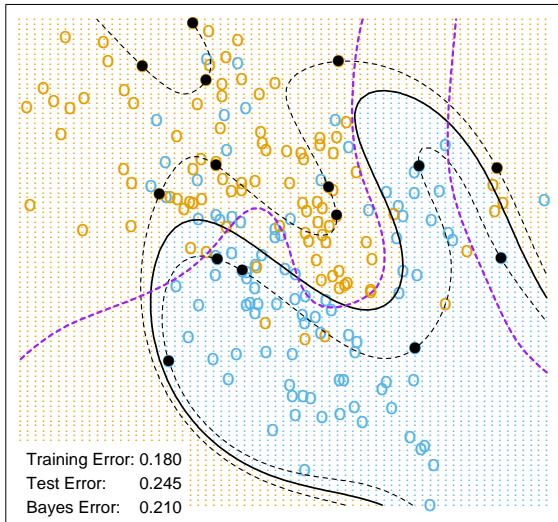
Then $M = 6$, and if we choose $h_1(X) = 1$, $h_2(X) = \sqrt{2}X_1$, $h_3(X) = \sqrt{2}X_2$, $h_4(X) = X_1^2$, $h_5(X) = X_2^2$, and $h_6(X) = \sqrt{2}X_1 X_2$, then $K(X, X') = \langle h(X), h(X') \rangle$. From (12.20) we see that the solution can be written

$$\hat{f}(x) = \sum_{i=1}^N \hat{\alpha}_i y_i K(x, x_i) + \hat{\beta}_0. \quad (12.24)$$

The role of the parameter C is clearer in an enlarged feature space, since perfect separation is often achievable there. A large value of C will discourage any positive ξ_i , and lead to an overfit wiggly boundary in the original feature space; a small value of C will encourage a small value of $\|\beta\|$, which in turn causes $f(x)$ and hence the boundary to be smoother. Figure 12.3 show two nonlinear support vector machines applied to the mixture example of Chapter 2. The regularization parameter was chosen in both cases to achieve good test error. The radial basis kernel produces a boundary quite similar to the Bayes optimal boundary for this example; compare Figure 2.5.

In the early literature on support vectors, there were claims that the kernel property of the support vector machine is unique to it and allows one to finesse the curse of dimensionality. Neither of these claims is true, and we go into both of these issues in the next three subsections.

SVM - Degree-4 Polynomial in Feature Space



SVM - Radial Kernel in Feature Space

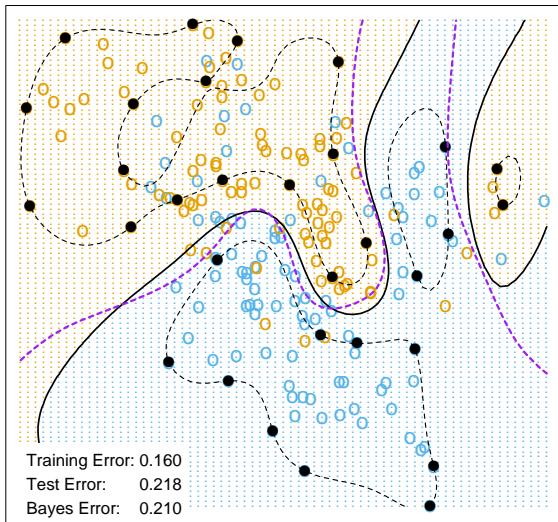


FIGURE 12.3. Two nonlinear SVMs for the mixture data. The upper plot uses a 4th degree polynomial kernel, the lower a radial basis kernel (with $\gamma = 1$). In each case C was tuned to approximately achieve the best test error performance, and $C = 1$ worked well in both cases. The radial basis kernel performs the best (close to Bayes optimal), as might be expected given the data arise from mixtures of Gaussians. The broken purple curve in the background is the Bayes decision boundary.

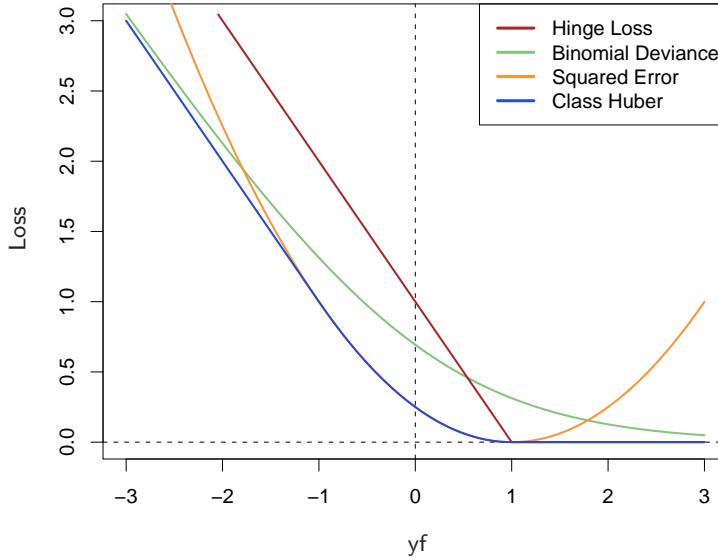


FIGURE 12.4. The support vector loss function (hinge loss), compared to the negative log-likelihood loss (binomial deviance) for logistic regression, squared-error loss, and a “Huberized” version of the squared hinge loss. All are shown as a function of yf rather than f , because of the symmetry between the $y = +1$ and $y = -1$ case. The deviance and Huber have the same asymptotes as the SVM loss, but are rounded in the interior. All are scaled to have the limiting left-tail slope of -1 .

12.3.2 The SVM as a Penalization Method

With $f(x) = h(x)^T \beta + \beta_0$, consider the optimization problem

$$\min_{\beta_0, \beta} \sum_{i=1}^N [1 - y_i f(x_i)]_+ + \frac{\lambda}{2} \|\beta\|^2 \quad (12.25)$$

where the subscript “+” indicates positive part. This has the form *loss* + *penalty*, which is a familiar paradigm in function estimation. It is easy to show (Exercise 12.1) that the solution to (12.25), with $\lambda = 1/C$, is the same as that for (12.8).

Examination of the “hinge” loss function $L(y, f) = [1 - yf]_+$ shows that it is reasonable for two-class classification, when compared to other more traditional loss functions. Figure 12.4 compares it to the log-likelihood loss for logistic regression, as well as squared-error loss and a variant thereof. The (negative) log-likelihood or binomial deviance has similar tails as the SVM loss, giving zero penalty to points well inside their margin, and a

TABLE 12.1. The population minimizers for the different loss functions in Figure 12.4. Logistic regression uses the binomial log-likelihood or deviance. Linear discriminant analysis (Exercise 4.2) uses squared-error loss. The SVM hinge loss estimates the mode of the posterior class probabilities, whereas the others estimate a linear transformation of these probabilities.

Loss Function	$L[y, f(x)]$	Minimizing Function
Binomial Deviance	$\log[1 + e^{-yf(x)}]$	$f(x) = \log \frac{\Pr(Y = +1 x)}{\Pr(Y = -1 x)}$
SVM Hinge Loss	$[1 - yf(x)]_+$	$f(x) = \text{sign}[\Pr(Y = +1 x) - \frac{1}{2}]$
Squared Error	$[y - f(x)]^2 = [1 - yf(x)]^2$	$f(x) = 2\Pr(Y = +1 x) - 1$
“Huberised” Square Hinge Loss	$-4yf(x), \quad yf(x) < -1$ $[1 - yf(x)]_+^2 \quad \text{otherwise}$	$f(x) = 2\Pr(Y = +1 x) - 1$

linear penalty to points on the wrong side and far away. Squared-error, on the other hand gives a quadratic penalty, and points well inside their own margin have a strong influence on the model as well. The squared hinge loss $L(y, f) = [1 - yf]_+^2$ is like the quadratic, except it is zero for points inside their margin. It still rises quadratically in the left tail, and will be less robust than hinge or deviance to misclassified observations. Recently Rosset and Zhu (2007) proposed a “Huberized” version of the squared hinge loss, which converts smoothly to a linear loss at $yf = -1$.

We can characterize these loss functions in terms of what they are estimating at the population level. We consider minimizing $EL(Y, f(X))$. Table 12.1 summarizes the results. Whereas the hinge loss estimates the classifier $G(x)$ itself, all the others estimate a transformation of the class posterior probabilities. The “Huberized” square hinge loss shares attractive properties of logistic regression (smooth loss function, estimates probabilities), as well as the SVM hinge loss (support points).

Formulation (12.25) casts the SVM as a regularized function estimation problem, where the coefficients of the linear expansion $f(x) = \beta_0 + h(x)^T \beta$ are shrunk toward zero (excluding the constant). If $h(x)$ represents a hierarchical basis having some ordered structure (such as ordered in roughness),

then the uniform shrinkage makes more sense if the rougher elements h_j in the vector h have smaller norm.

All the loss-function in Table 12.1 except squared-error are so called “margin maximizing loss-functions” (Rosset et al., 2004b). This means that if the data are separable, then the limit of $\hat{\beta}_\lambda$ in (12.25) as $\lambda \rightarrow 0$ defines the optimal separating hyperplane¹.

12.3.3 Function Estimation and Reproducing Kernels



Here we describe SVMs in terms of function estimation in reproducing kernel Hilbert spaces, where the kernel property abounds. This material is discussed in some detail in Section 5.8. This provides another view of the support vector classifier, and helps to clarify how it works.

Suppose the basis h arises from the (possibly finite) eigen-expansion of a positive definite kernel K ,

$$K(x, x') = \sum_{m=1}^{\infty} \phi_m(x)\phi_m(x')\delta_m \quad (12.26)$$

and $h_m(x) = \sqrt{\delta_m}\phi_m(x)$. Then with $\theta_m = \sqrt{\delta_m}\beta_m$, we can write (12.25) as

$$\min_{\beta_0, \theta} \sum_{i=1}^N \left[1 - y_i(\beta_0 + \sum_{m=1}^{\infty} \theta_m \phi_m(x_i)) \right]_+ + \frac{\lambda}{2} \sum_{m=1}^{\infty} \frac{\theta_m^2}{\delta_m}. \quad (12.27)$$

Now (12.27) is identical in form to (5.49) on page 169 in Section 5.8, and the theory of reproducing kernel Hilbert spaces described there guarantees a finite-dimensional solution of the form

$$f(x) = \beta_0 + \sum_{i=1}^N \alpha_i K(x, x_i). \quad (12.28)$$

In particular we see there an equivalent version of the optimization criterion (12.19) [Equation (5.67) in Section 5.8.2; see also Wahba et al. (2000)],

$$\min_{\beta_0, \alpha} \sum_{i=1}^N (1 - y_i f(x_i))_+ + \frac{\lambda}{2} \alpha^T \mathbf{K} \alpha, \quad (12.29)$$

where \mathbf{K} is the $N \times N$ matrix of kernel evaluations for all pairs of training features (Exercise 12.2).

These models are quite general, and include, for example, the entire family of smoothing splines, additive and interaction spline models discussed

¹For logistic regression with separable data, $\hat{\beta}_\lambda$ diverges, but $\hat{\beta}_\lambda / \|\hat{\beta}_\lambda\|$ converges to the optimal separating direction.

in Chapters 5 and 9, and in more detail in Wahba (1990) and Hastie and Tibshirani (1990). They can be expressed more generally as

$$\min_{f \in \mathcal{H}} \sum_{i=1}^N [1 - y_i f(x_i)]_+ + \lambda J(f), \quad (12.30)$$

where \mathcal{H} is the structured space of functions, and $J(f)$ an appropriate regularizer on that space. For example, suppose \mathcal{H} is the space of additive functions $f(x) = \sum_{j=1}^p f_j(x_j)$, and $J(f) = \sum_j \int \{f''_j(x_j)\}^2 dx_j$. Then the solution to (12.30) is an additive cubic spline, and has a kernel representation (12.28) with $K(x, x') = \sum_{j=1}^p K_j(x_j, x'_j)$. Each of the K_j is the kernel appropriate for the univariate smoothing spline in x_j (Wahba, 1990).

Conversely this discussion also shows that, for example, *any* of the kernels described in (12.22) above can be used with *any* convex loss function, and will also lead to a finite-dimensional representation of the form (12.28). Figure 12.5 uses the same kernel functions as in Figure 12.3, except using the binomial log-likelihood as a loss function². The fitted function is hence an estimate of the log-odds,

$$\begin{aligned} \hat{f}(x) &= \log \frac{\hat{\Pr}(Y = +1|x)}{\hat{\Pr}(Y = -1|x)} \\ &= \hat{\beta}_0 + \sum_{i=1}^N \hat{\alpha}_i K(x, x_i), \end{aligned} \quad (12.31)$$

or conversely we get an estimate of the class probabilities

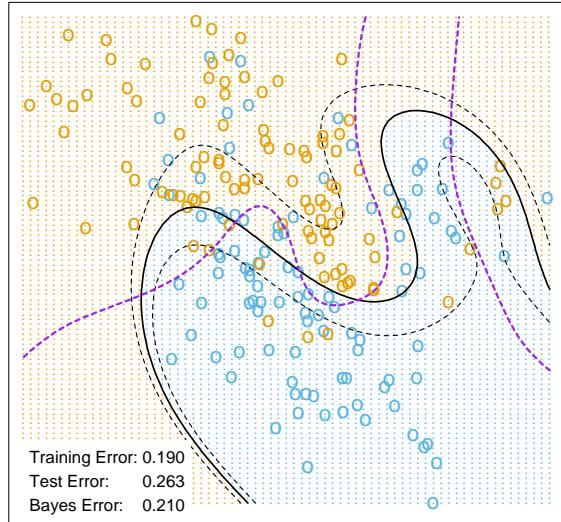
$$\hat{\Pr}(Y = +1|x) = \frac{1}{1 + e^{-\hat{\beta}_0 - \sum_{i=1}^N \hat{\alpha}_i K(x, x_i)}}. \quad (12.32)$$

The fitted models are quite similar in shape and performance. Examples and more details are given in Section 5.8.

It does happen that for SVMs, a sizable fraction of the N values of α_i can be zero (the nonsupport points). In the two examples in Figure 12.3, these fractions are 42% and 45%, respectively. This is a consequence of the piecewise linear nature of the first part of the criterion (12.25). The lower the class overlap (on the training data), the greater this fraction will be. Reducing λ will generally reduce the overlap (allowing a more flexible f). A small number of support points means that $\hat{f}(x)$ can be evaluated more quickly, which is important at lookup time. Of course, reducing the overlap too much can lead to poor generalization.

²Ji Zhu assisted in the preparation of these examples.

LR - Degree-4 Polynomial in Feature Space



LR - Radial Kernel in Feature Space

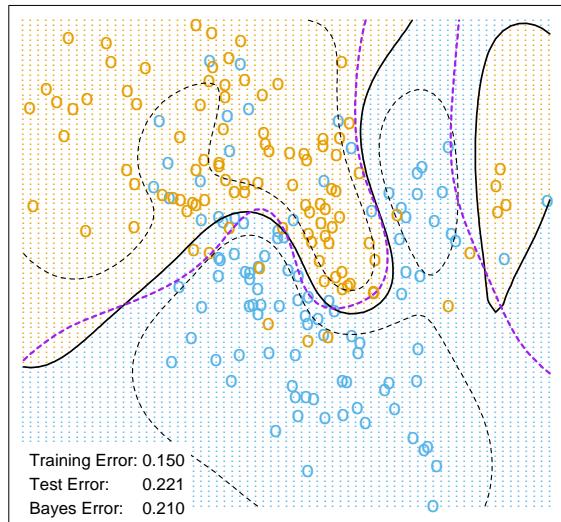


FIGURE 12.5. The logistic regression versions of the SVM models in Figure 12.3, using the identical kernels and hence penalties, but the log-likelihood loss instead of the SVM loss function. The two broken contours correspond to posterior probabilities of 0.75 and 0.25 for the +1 class (or vice versa). The broken purple curve in the background is the Bayes decision boundary.

TABLE 12.2. Skin of the orange: Shown are mean (standard error of the mean) of the test error over 50 simulations. BRUTO fits an additive spline model adaptively, while MARS fits a low-order interaction model adaptively.

Method	Test Error (SE)	
	No Noise Features	Six Noise Features
1 SV Classifier	0.450 (0.003)	0.472 (0.003)
2 SVM/poly 2	0.078 (0.003)	0.152 (0.004)
3 SVM/poly 5	0.180 (0.004)	0.370 (0.004)
4 SVM/poly 10	0.230 (0.003)	0.434 (0.002)
5 BRUTO	0.084 (0.003)	0.090 (0.003)
6 MARS	0.156 (0.004)	0.173 (0.005)
Bayes	0.029	0.029

12.3.4 SVMs and the Curse of Dimensionality

In this section, we address the question of whether SVMs have some edge on the curse of dimensionality. Notice that in expression (12.23) we are not allowed a fully general inner product in the space of powers and products. For example, all terms of the form $2X_j X'_j$ are given equal weight, and the kernel cannot adapt itself to concentrate on subspaces. If the number of features p were large, but the class separation occurred only in the linear subspace spanned by say X_1 and X_2 , this kernel would not easily find the structure and would suffer from having many dimensions to search over. One would have to build knowledge about the subspace into the kernel; that is, tell it to ignore all but the first two inputs. If such knowledge were available a priori, much of statistical learning would be made much easier. A major goal of adaptive methods is to discover such structure.

We support these statements with an illustrative example. We generated 100 observations in each of two classes. The first class has four standard normal independent features X_1, X_2, X_3, X_4 . The second class also has four standard normal independent features, but conditioned on $9 \leq \sum X_j^2 \leq 16$. This is a relatively easy problem. As a second harder problem, we augmented the features with an additional six standard Gaussian noise features. Hence the second class almost completely surrounds the first, like the skin surrounding the orange, in a four-dimensional subspace. The Bayes error rate for this problem is 0.029 (irrespective of dimension). We generated 1000 test observations to compare different procedures. The average test errors over 50 simulations, with and without noise features, are shown in Table 12.2.

Line 1 uses the support vector classifier in the original feature space. Lines 2–4 refer to the support vector machine with a 2-, 5- and 10-dimensional polynomial kernel. For all support vector procedures, we chose the cost parameter C to minimize the test error, to be as fair as possible to the

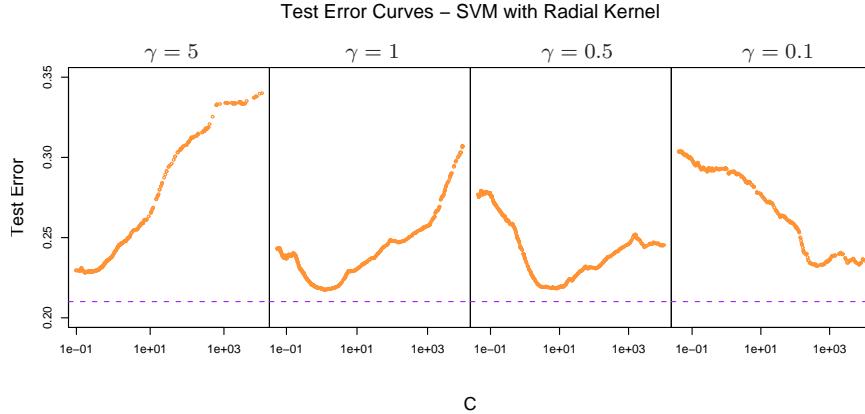


FIGURE 12.6. Test-error curves as a function of the cost parameter C for the radial-kernel SVM classifier on the mixture data. At the top of each plot is the scale parameter γ for the radial kernel: $K_\gamma(x, y) = \exp -\gamma \|x - y\|^2$. The optimal value for C depends quite strongly on the scale of the kernel. The Bayes error rate is indicated by the broken horizontal lines.

method. Line 5 fits an additive spline model to the $(-1, +1)$ response by least squares, using the BRUTO algorithm for additive models, described in Hastie and Tibshirani (1990). Line 6 uses MARS (multivariate adaptive regression splines) allowing interaction of all orders, as described in Chapter 9; as such it is comparable with the SVM/poly 10. Both BRUTO and MARS have the ability to ignore redundant variables. Test error was not used to choose the smoothing parameters in either of lines 5 or 6.

In the original feature space, a hyperplane cannot separate the classes, and the support vector classifier (line 1) does poorly. The polynomial support vector machine makes a substantial improvement in test error rate, but is adversely affected by the six noise features. It is also very sensitive to the choice of kernel: the second degree polynomial kernel (line 2) does best, since the true decision boundary is a second-degree polynomial. However, higher-degree polynomial kernels (lines 3 and 4) do much worse. BRUTO performs well, since the boundary is additive. BRUTO and MARS adapt well: their performance does not deteriorate much in the presence of noise.

12.3.5 A Path Algorithm for the SVM Classifier



The regularization parameter for the SVM classifier is the cost parameter C , or its inverse λ in (12.25). Common usage is to set C high, leading often to somewhat overfit classifiers.

Figure 12.6 shows the test error on the mixture data as a function of C , using different radial-kernel parameters γ . When $\gamma = 5$ (narrow peaked kernels), the heaviest regularization (small C) is called for. With $\gamma = 1$

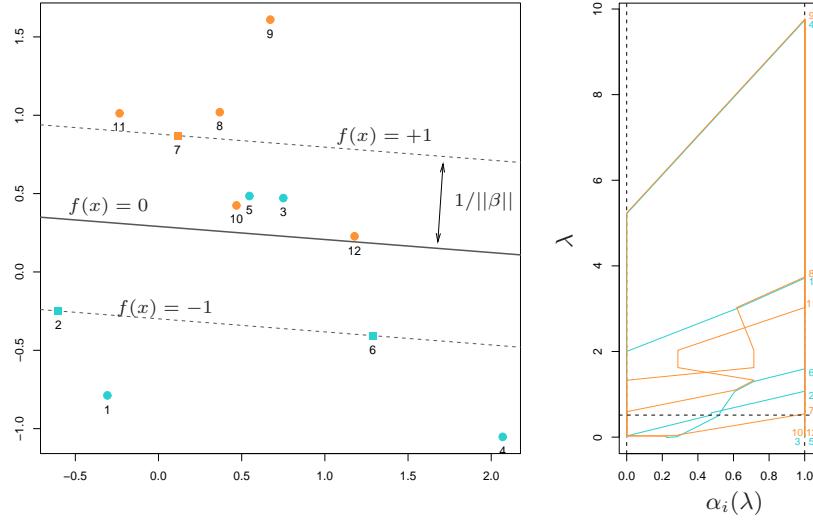


FIGURE 12.7. A simple example illustrates the SVM path algorithm. (left panel:) This plot illustrates the state of the model at $\lambda = 0.05$. The “+1” points are orange, the “−1” blue. $\lambda = 1/2$, and the width of the soft margin is $2/\|\beta\| = 2 \times 0.587$. Two blue points {3, 5} are misclassified, while the two orange points {10, 12} are correctly classified, but on the wrong side of their margin $f(x) = +1$; each of these has $y_i f(x_i) < 1$. The three square shaped points {2, 6, 7} are exactly on their margins. (right panel:) This plot shows the piecewise linear profiles $\alpha_i(\lambda)$. The horizontal broken line at $\lambda = 1/2$ indicates the state of the α_i for the model in the left plot.

(the value used in Figure 12.3), an intermediate value of C is required. Clearly in situations such as these, we need to determine a good choice for C , perhaps by cross-validation. Here we describe a path algorithm (in the spirit of Section 3.8) for efficiently fitting the entire sequence of SVM models obtained by varying C .

It is convenient to use the loss+penalty formulation (12.25), along with Figure 12.4. This leads to a solution for β at a given value of λ :

$$\beta_\lambda = \frac{1}{\lambda} \sum_{i=1}^N \alpha_i y_i x_i. \quad (12.33)$$

The α_i are again Lagrange multipliers, but in this case they all lie in $[0, 1]$.

Figure 12.7 illustrates the setup. It can be shown that the KKT optimality conditions imply that the labeled points (x_i, y_i) fall into three distinct groups:

- Observations correctly classified and outside their margins. They have $y_i f(x_i) > 1$, and Lagrange multipliers $\alpha_i = 0$. Examples are the orange points 8, 9 and 11, and the blue points 1 and 4.
- Observations sitting on their margins with $y_i f(x_i) = 1$, with Lagrange multipliers $\alpha_i \in [0, 1]$. Examples are the orange 7 and the blue 2 and 8.
- Observations inside their margins have $y_i f(x_i) < 1$, with $\alpha_i = 1$. Examples are the blue 3 and 5, and the orange 10 and 12.

The idea for the path algorithm is as follows. Initially λ is large, the margin $1/\|\beta_\lambda\|$ is wide, and all points are inside their margin and have $\alpha_i = 1$. As λ decreases, $1/\|\beta_\lambda\|$ decreases, and the margin gets narrower. Some points will move from inside their margins to outside their margins, and their α_i will change from 1 to 0. By continuity of the $\alpha_i(\lambda)$, these points will *linger* on the margin during this transition. From (12.33) we see that the points with $\alpha_i = 1$ make fixed contributions to $\beta(\lambda)$, and those with $\alpha_i = 0$ make no contribution. So all that changes as λ decreases are the $\alpha_i \in [0, 1]$ of those (small number) of points on the margin. Since all these points have $y_i f(x_i) = 1$, this results in a small set of linear equations that prescribe how $\alpha_i(\lambda)$ and hence β_λ changes during these transitions. This results in piecewise linear paths for each of the $\alpha_i(\lambda)$. The breaks occur when points cross the margin. Figure 12.7 (right panel) shows the $\alpha_i(\lambda)$ profiles for the small example in the left panel.

Although we have described this for linear SVMs, exactly the same idea works for nonlinear models, in which (12.33) is replaced by

$$f_\lambda(x) = \frac{1}{\lambda} \sum_{i=1}^N \alpha_i y_i K(x, x_i). \quad (12.34)$$

Details can be found in Hastie et al. (2004). An R package `svmpath` is available on CRAN for fitting these models.

12.3.6 Support Vector Machines for Regression

In this section we show how SVMs can be adapted for regression with a quantitative response, in ways that inherit some of the properties of the SVM classifier. We first discuss the linear regression model

$$f(x) = x^T \beta + \beta_0, \quad (12.35)$$

and then handle nonlinear generalizations. To estimate β , we consider minimization of

$$H(\beta, \beta_0) = \sum_{i=1}^N V(y_i - f(x_i)) + \frac{\lambda}{2} \|\beta\|^2, \quad (12.36)$$

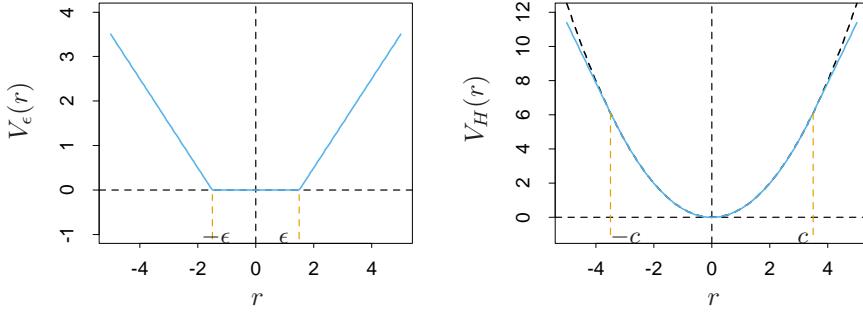


FIGURE 12.8. The left panel shows the ϵ -insensitive error function used by the support vector regression machine. The right panel shows the error function used in Huber’s robust regression (blue curve). Beyond $|c|$, the function changes from quadratic to linear.

where

$$V_\epsilon(r) = \begin{cases} 0 & \text{if } |r| < \epsilon, \\ |r| - \epsilon, & \text{otherwise.} \end{cases} \quad (12.37)$$

This is an “ ϵ -insensitive” error measure, ignoring errors of size less than ϵ (left panel of Figure 12.8). There is a rough analogy with the support vector classification setup, where points on the correct side of the decision boundary and far away from it, are ignored in the optimization. In regression, these “low error” points are the ones with small residuals.

It is interesting to contrast this with error measures used in robust regression in statistics. The most popular, due to Huber (1964), has the form

$$V_H(r) = \begin{cases} r^2/2 & \text{if } |r| \leq c, \\ c|r| - c^2/2, & |r| > c, \end{cases} \quad (12.38)$$

shown in the right panel of Figure 12.8. This function reduces from quadratic to linear the contributions of observations with absolute residual greater than a prechosen constant c . This makes the fitting less sensitive to outliers. The support vector error measure (12.37) also has linear tails (beyond ϵ), but in addition it flattens the contributions of those cases with small residuals.

If $\hat{\beta}$, $\hat{\beta}_0$ are the minimizers of H , the solution function can be shown to have the form

$$\hat{\beta} = \sum_{i=1}^N (\hat{\alpha}_i^* - \hat{\alpha}_i) x_i, \quad (12.39)$$

$$\hat{f}(x) = \sum_{i=1}^N (\hat{\alpha}_i^* - \hat{\alpha}_i) \langle x, x_i \rangle + \beta_0, \quad (12.40)$$

where $\hat{\alpha}_i, \hat{\alpha}_i^*$ are positive and solve the quadratic programming problem

$$\min_{\alpha_i, \alpha_i^*} \epsilon \sum_{i=1}^N (\alpha_i^* + \alpha_i) - \sum_{i=1}^N y_i (\alpha_i^* - \alpha_i) + \frac{1}{2} \sum_{i,i'=1}^N (\alpha_i^* - \alpha_i)(\alpha_{i'}^* - \alpha_{i'}) \langle x_i, x_{i'} \rangle$$

subject to the constraints

$$\begin{aligned} 0 &\leq \alpha_i, \quad \alpha_i^* \leq 1/\lambda, \\ \sum_{i=1}^N (\alpha_i^* - \alpha_i) &= 0, \\ \alpha_i \alpha_i^* &= 0. \end{aligned} \tag{12.41}$$

Due to the nature of these constraints, typically only a subset of the solution values $(\hat{\alpha}_i^* - \hat{\alpha}_i)$ are nonzero, and the associated data values are called the support vectors. As was the case in the classification setting, the solution depends on the input values only through the inner products $\langle x_i, x_{i'} \rangle$. Thus we can generalize the methods to richer spaces by defining an appropriate inner product, for example, one of those defined in (12.22).

Note that there are parameters, ϵ and λ , associated with the criterion (12.36). These seem to play different roles. ϵ is a parameter of the loss function V_ϵ , just like c is for V_H . Note that both V_ϵ and V_H depend on the scale of y and hence r . If we scale our response (and hence use $V_H(r/\sigma)$ and $V_\epsilon(r/\sigma)$ instead), then we might consider using preset values for c and ϵ (the value $c = 1.345$ achieves 95% efficiency for the Gaussian). The quantity λ is a more traditional regularization parameter, and can be estimated for example by cross-validation.

12.3.7 Regression and Kernels

As discussed in Section 12.3.3, this kernel property is not unique to support vector machines. Suppose we consider approximation of the regression function in terms of a set of basis functions $\{h_m(x)\}, m = 1, 2, \dots, M$:

$$f(x) = \sum_{m=1}^M \beta_m h_m(x) + \beta_0. \tag{12.42}$$

To estimate β and β_0 we minimize

$$H(\beta, \beta_0) = \sum_{i=1}^N V(y_i - f(x_i)) + \frac{\lambda}{2} \sum \beta_m^2 \tag{12.43}$$

for some general error measure $V(r)$. For any choice of $V(r)$, the solution $\hat{f}(x) = \sum \hat{\beta}_m h_m(x) + \hat{\beta}_0$ has the form

$$\hat{f}(x) = \sum_{i=1}^N \hat{\alpha}_i K(x, x_i) \tag{12.44}$$

with $K(x, y) = \sum_{m=1}^M h_m(x)h_m(y)$. Notice that this has the same form as both the radial basis function expansion and a regularization estimate, discussed in Chapters 5 and 6.

For concreteness, let's work out the case $V(r) = r^2$. Let \mathbf{H} be the $N \times M$ basis matrix with i th element $h_m(x_i)$, and suppose that $M > N$ is large. For simplicity we assume that $\beta_0 = 0$, or that the constant is absorbed in h ; see Exercise 12.3 for an alternative.

We estimate β by minimizing the penalized least squares criterion

$$H(\beta) = (\mathbf{y} - \mathbf{H}\beta)^T(\mathbf{y} - \mathbf{H}\beta) + \lambda\|\beta\|^2. \quad (12.45)$$

The solution is

$$\hat{\mathbf{y}} = \mathbf{H}\hat{\beta} \quad (12.46)$$

with $\hat{\beta}$ determined by

$$-\mathbf{H}^T(\mathbf{y} - \mathbf{H}\hat{\beta}) + \lambda\hat{\beta} = 0. \quad (12.47)$$

From this it appears that we need to evaluate the $M \times M$ matrix of inner products in the transformed space. However, we can premultiply by \mathbf{H} to give

$$\mathbf{H}\hat{\beta} = (\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I})^{-1}\mathbf{H}\mathbf{y}. \quad (12.48)$$

The $N \times N$ matrix $\mathbf{H}\mathbf{H}^T$ consists of inner products between pairs of observations i, i' ; that is, the evaluation of an inner product kernel $\{\mathbf{H}\mathbf{H}^T\}_{i,i'} = K(x_i, x_{i'})$. It is easy to show (12.44) directly in this case, that the predicted values at an arbitrary x satisfy

$$\begin{aligned} \hat{f}(x) &= h(x)^T\hat{\beta} \\ &= \sum_{i=1}^N \hat{\alpha}_i K(x, x_i), \end{aligned} \quad (12.49)$$

where $\hat{\alpha} = (\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I})^{-1}\mathbf{y}$. As in the support vector machine, we need not specify or evaluate the large set of functions $h_1(x), h_2(x), \dots, h_M(x)$. Only the inner product kernel $K(x_i, x_{i'})$ need be evaluated, at the N training points for each i, i' and at points x for predictions there. Careful choice of h_m (such as the eigenfunctions of particular, easy-to-evaluate kernels K) means, for example, that $\mathbf{H}\mathbf{H}^T$ can be computed at a cost of $N^2/2$ evaluations of K , rather than the direct cost N^2M .

Note, however, that this property depends on the choice of squared norm $\|\beta\|^2$ in the penalty. It does not hold, for example, for the L_1 norm $|\beta|$, which may lead to a superior model.

12.3.8 Discussion

The support vector machine can be extended to multiclass problems, essentially by solving many two-class problems. A classifier is built for each pair of classes, and the final classifier is the one that dominates the most (Kressel, 1999; Friedman, 1996; Hastie and Tibshirani, 1998). Alternatively, one could use the multinomial loss function along with a suitable kernel, as in Section 12.3.3. SVMs have applications in many other supervised and unsupervised learning problems. At the time of this writing, empirical evidence suggests that it performs well in many real learning problems.

Finally, we mention the connection of the support vector machine and structural risk minimization (7.9). Suppose the training points (or their basis expansion) are contained in a sphere of radius R , and let $G(x) = \text{sign}[f(x)] = \text{sign}[\beta^T x + \beta_0]$ as in (12.2). Then one can show that the class of functions $\{G(x), \|\beta\| \leq A\}$ has VC-dimension h satisfying

$$h \leq R^2 A^2. \quad (12.50)$$

If $f(x)$ separates the training data, optimally for $\|\beta\| \leq A$, then with probability at least $1 - \eta$ over training sets (Vapnik, 1996, page 139):

$$\text{Error}_{\text{Test}} \leq 4 \frac{h[\log(2N/h) + 1] - \log(\eta/4)}{N}. \quad (12.51)$$

The support vector classifier was one of the first practical learning procedures for which useful bounds on the VC dimension could be obtained, and hence the SRM program could be carried out. However in the derivation, balls are put around the data points—a process that depends on the observed values of the features. Hence in a strict sense, the VC complexity of the class is not fixed a priori, before seeing the features.

The regularization parameter C controls an upper bound on the VC dimension of the classifier. Following the SRM paradigm, we could choose C by minimizing the upper bound on the test error, given in (12.51). However, it is not clear that this has any advantage over the use of cross-validation for choice of C .

12.4 Generalizing Linear Discriminant Analysis

In Section 4.3 we discussed linear discriminant analysis (LDA), a fundamental tool for classification. For the remainder of this chapter we discuss a class of techniques that produce better classifiers than LDA by directly generalizing LDA.

Some of the virtues of LDA are as follows:

- It is a simple prototype classifier. A new observation is classified to the class with closest centroid. A slight twist is that distance is measured in the Mahalanobis metric, using a pooled covariance estimate.