BRAC University

Intro to Statistical Learning

Shreas

A collection of notes on

Data Science

Department of Mathematics

BRAC University

August 2025

Preface

Hi, my name is Shreas. I am writing this to be intended as a stripped down version of the book **An Introduction to Statistical Learning**. This must only be read if you have finished the book, otherwise it will be difficult to follow. All the errors are solely my fault. If you find any, please email me at shreas.labib.arion@.g.bracu.ac.bd. I hope you find this useful.

Contents

1 Introduction

1.1 An overview of Statistical Learning

Statistical learning can be classified into supervised or unsupervised learning. Supservised statistical learning involves building a statistical model for predicting, or estimating, an output based on one or more inputs. With unsupervised statistical learning, there are inputs but no supervising output; but we can learn relationships and structure from such data.

1.2 A Brief History of Statistical Learning

The term statistical learning is fairly new, many of the concepts that underlie the field were developed long ago. At the beginning of the nineteenth century, Legendre and Gauss published papers on the method of least squares, which implemented the earliest form of what is now known as linear regression.

Linear Regression is used for predicting quantitative values. In order to predict qualitative values, Fisher proposed linear discriminant analysis in 1936.

2 Statistical Learning $f(X)$

2.1 What is Statistical Learning?

Suppose that we are statistical consultants hired by a client to provide advice on how to improve sales of a particular product. It is not possible for our client to directly increase sales of the prodcut. On the other hand, they can control the advertising expenditure. Therefore, if we can determine that there is an association between advertising and sales, then we can instruct our client to adjust advertising budgets, thereby indirectly increasing sales. Our goal is to develop an accurate model that can be used to predict sales on the basis of advertising budgets.

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 ($X_1, X_2, \ldots$). The inputs go by different names, such as predictors, independent variables, features, or sometimes just variables.

The output variable is often called the response or dependent variable, and is typically denoted using the symbol Y.

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

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

Here f is some fixed but unknown function of $X_1, \ldots, X_p$, 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. The function f may involve more than one input variable.

In essence, statistical learning refers to a set of approaches for estimating f.

2.1.1 Why Estimate f?

There are two main reasons that we may wish to estimate f: prediction and inference.

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

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

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

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 beacause 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.

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.

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

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

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.

It is important to keep in mind that the irreducible error will always provide an upper bound on the accuracy of our predictions for Y. This bound is almost always unknown in practice.

Inference

We are often interested in understanding the way that Y is affected as $X_1, \ldots, X_p$ 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 $X_1, \ldots, X_p$. Now $\hat{f}$ cannot be treated as a black box, because we need to know its exact form. One may be interested in answering the following questions:

- Which predictors are associated with the response?
- What is the relationship between the response and each predictor?
- Can the relationship between Y and each predictor be adequately summarized using a linear equation, or is the relationship more complicated?

Depending on whether our goal is prediction, inference, or a combination of the two, different methods for estimating f may be appropriate. 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 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.

2.1.2 How Do We Estimate f?

Our goal is to apply a statistical learning method to the training data in order to estimate the unknown function f. We want to find a function $\hat{f}$ such that $Y \approx \hat{f}(X)$ for any observation (X, Y). Most statistical learning methods for this task can be characterized as either parametric or non-parametric.

Parametric Methods

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

1. We make an assumption about the functional form, or shape, of f. One very simple assumption is that f is linear in X:

   $$Y \approx \beta_0 + \beta_1 X.$$

   This is a linear model. 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 $\beta_0, \beta_1, \cdots, \beta_p$.

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, we need to estimate the parameters $\beta_1, \beta_1, \ldots, \beta_p$. We want to find values of these parameters such that

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

   The most common approach to fitting the model is referred to as (ordinary) least squares.

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 parameters. 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 address this problem by choosing flexible methods that can fit many different possible functional forms for f. Fitting a more flexible model requires estimating a greater number of parameters.

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. There is a 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. Since they do not reduce the problem of estimating f to a small number of parameters, a very large number of observations is required.

2.1.3 The Trade-Off Between Prediction Accuracy and Model Interpretability

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. In contrast, very flexible approaches, such as the splines and the boosting methods can lead to complicated estimates of f that it is difficult to understand how any individual predictor is associated with the response.

2.1.4 Supervised Versus Unsupervised Learning

Most statistical learning problems fall into one of two categories: supervised or unsupervised. For each observation of the predictor measurements $x_i, i = 1, \ldots, n$ 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 or better understanding the relationship between the response and the predictors.

In contrast, unsupervised learning describes somewhat more challenging situation in which for every observation $i = 1, \ldots, n$, we observe a vector of measurements x_i but no associated response y_i. 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 $x_1, \ldots, x_n$, whether the observations fall into relatively distinct groups. Identifying groups can be of interest beacause it might be that the

groups differ with respect to some property of interest. If there are p variables in our data set, then $p(p-1)/2$ distinct scatterplots can be made, and visual inspection is not a viable way to identify clusters. For this reason, automated clustering methods are important.

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. We refer to this setting as a semi-supervised learning problem.

2.1.5 Regression Versus Classification Problems

Variables can be characterized as either quantitative or qualitative. Quantitative variables take on numerical values. In contrast, qualitative variables take on values in one of K different classes, or categories. 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. Least squares linear regression is used with a quantitative response, whereas logistic regression is typically used with a binary qualitative response.

2.2 Assessing Model Accuracy

On a particular data set, one specific method may work best, but some other may work better on a similar but different data set.

2.2.1 Measuring the Quality of Fit

In order to evaluate the performance of a statistical learning method, 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

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

where $\hat{f}(x_i)$ is the prediction that $\hat{f}$ gives for the ith 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 is computed using the training data that was used to fit the model. We are interested in the accuracy of the predictions that we obtain when we apply our method to previously unseen

test data.

Suppose that we fit our statistical learning method on our training observations $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, and we obtain the estimate $\hat{f}$. We can then compute $\hat{f}(x_1), \hat{f}(x_2), \ldots, \hat{f}(x_n)$. If these are approximately equal to $y_1, y_2, \ldots, y_n$, then the training MSE is small. We are 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 and not used to train the learning method. We want to choose the method that gives the lowest test MSE, as opposed to the lowest training MSE. If we had a large number of test observations, we could compute

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

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.

There is no guarantee that the method with the lowest training MSE will also have the lowest test MSE. The training set MSE can be quite small, but the test MSE is often much larger.

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. When a given method yields a small trainig MSE but a large test MSE, we are said to be overfitting the data.

One can usually compute the training MSE with relative ease, but estimating test MSE is considerably more difficult.

2.2.2 The Bias-Variance Trade-Off

The U-shape in the test MSE curves is the result of two competing properties of statistical learning methods. The expected test MSE, for a given value x_0, can 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,

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

In order to minimsize the expected test error, we need to select a statistical learning method that simultaneously achieves low variance and low bias. 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.

Variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set. Different training data sets will result in a different $\hat{f}$. If a method has high variance then small changes in the training data can result in large changes in $\hat{f}$.

Bias refers to the error that is introduced by approximating a real life problem. Linear regression assumes that there is a linear relationship between Y and $X_1, X_2, \ldots, X_p$. It is unlikely that any real-life problem truly has such a simple linear relationship, and so performing linear regression will result in some bias in the estimate of f. Generally, more flexible methods result in less bias.

As we use more flexible methods, the variance will increase and the bias will decrease. 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.

The relationship between bias, variance, and test set MSE 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. We need to find a method for which both the variance and the squared bias are low.

2.2.3 The Classification Setting

Suppose that we seek to estimate f on the basis of training observations $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where now $y_1, \ldots, y_n$ 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:

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

Here $\hat{y}_i$ is the predicted class label for the ith 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 $y_i = \hat{y}_i$.

We are most interested in the error rates that result from applying our classifier to test obsservations 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

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

where $\hat{y}_0$ is the predicted class label that results from applying the classifier for the test observation with predictor x_0.

The Bayes Classifier

We should simply assign a test observation with predictor vector x_0 to the class j for which

$$\Pr(Y = j \mid X = x_0)$$

is largest. This very simple classifier is called the Bayes classifier. In a two-class problem where there are two possible response values, the Bayes classifier corresponds to predicting class one if $\Pr(Y = 1 \mid X = x_0) > 0.5$, and class two otherwise.

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 $\Pr(Y = j \mid X = x_0)$ is largest, the error rate at $X = x_0$ will be $1 - \max_j \Pr(Y = j \mid X = x_0)$. The overall Bayes error rate is given by

$$1 - E(\max_j \Pr(Y - j \mid X)),$$

where the expectation averages the probability over all possible values of X.

K-Nearest Neighbours

We would always like to predict qualitative responses using the Bayes classifier. For real data, we do not know the conditional distribution of Y given X, and so computing the Bayes classifier is impossible. Many approaches attemp 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 iz the 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:

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

The choice of K has a drastic effect on the KNN classifier obtained.

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. As we use more flexible classification methods, the training error rate will decline but the test error may not.

Choosing the correct level of flexibility is critical to the success of any statistical learning method.

3 Linear Regression

Linear regression is a useful tool for predicting a quantitative response. Here are a few important questions that we might seek to address:

1. Is there any relationship between the input and output?
2. How strong is the relationship between the input and output?
3. Which input category contributes to the output?
4. How accurately can we estimate the effect of each input criteria on the output?
5. How accurately can we predict future data?
6. Is the relationship linear?
7. Is there synergy among the input criteria?

Linear regression can be used to answer each of these questions.

3.1 Simple Linear Regression

It assumes that there is approximately a linear relationship between X and Y. We can write this linear relationship as

$$Y \approx \beta_0 + \beta_1 X.$$

This is a linear model. 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 $\beta_0, \beta_1, \cdots, \beta_p$.

There 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 data by computing

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x,$$

w where $\hat{y}$ indicates a prediction of Y on the basis of $X = x$.

3.1.1 Estimating the Coefficients

In practice, β_0 and β_1 are unknown. Let $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ represent n observation pairs each of which consists of a measurement of X and a measurement of Y. Our goal is to obtain coefficient estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ such that the linear model fits the data well, that is, so that $y_i \approx \hat{\beta}_0 + \hat{\beta}_1 x_i$ for $i = 1, \ldots, n$. We want to find an intercept $\hat{\beta}_0$ and a slope $\hat{\beta}_1$ such that the resulting line is close as possible to all the data points. The most common approach involves minimizing the least squares criterion. Let $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ be the prediction for Y based on the ith value of X. Then $e_i = y_i - \hat{y}_i$ represents that ith residual – this is the difference between the ith observed response value and the ith response value that is predicted by our linear model. We define the residual sum of squares (RSS) as

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

or equivalently as

$$\text{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + \cdots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2.$$

The least squares approach chooses $\hat{\beta}_0$ and $\hat{\beta}_1$ to minimize the RSS. The minimizers are

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2},$$

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x},$$

where $\overline{y} = \frac{1}{n} = \sum_{i=1}^{n} y_i$ and $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ are the sample means.

3.1.2 Assessing the Accuracy of the Coefficient Estimates

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. 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. It means that one the basis of one particular set of observations, $\hat{\mu}$ might underestimate μ and some might overestimate. Then this average would exactly equal μ. The property of unbiasedness holds for the least squares coefficient estimates as well.

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? We answer this question by computing the standard error of $\hat{\mu}$, written as

$\text{SE}(\hat{\mu})$. We have the well-known formula

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

where σ is the standard deviation of each of the realizations of y_i of Y. The standard error tells us the average amount that this estimate $\hat{\mu}$ differs from the actual value of μ. The more observations we have, the smaller the standard error of $\hat{\mu}$. To compute the standard errors associated with $\hat{\beta}_0$ and $\hat{\beta}_1$, we use the following formulas:

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

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. 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 $\text{RSE} = \sqrt{\text{RSS}/(n-2)}$.

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

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

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

$$H_0 : \text{There is no relationship between } X \text{ and } Y$$

versus the alternative hypothesis

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

M Mathematically, this corresponds to testing

$$H_0 : \beta_1 = 0$$

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

since if $\beta_1 = 0$ then $Y = \beta_0 + \epsilon$, and X is not associated with Y. To test the null hypothesis, we need to determine whether $\hat{\beta}_1$ is sufficiently far from 0 that we can be confident that β_1 is 0. 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 there is a relationship between X and Y. In practice, we compute a t-statistic, given by

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

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 we 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. It is a simple matter to compute the probability of observing any number equal to $|t|$ or larger in absolute value, assuming $\beta_1 = 0$. We call this probability the p-value. 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 cuttoffs for rejecting the null hypothesis are 5 or 1%.

3.1.3 Assessing the Accuracy of the Model

We want to quantify the extent to which the model fits the data. The quality of a linear regression is typically assessed using two related quantities: the residual standard error (RSE) and the R^2 statistic.

Residual Standard Error

In the model, 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, we would not be able to perfectly predict Y from X. The RSE is an estimate of the standard deviation of ϵ. It is the average amount that the response will deviate from the true regression line. It is computed using the formula

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

The RSE is considered a measure of the lack of fit of the model to the data. If $\hat{y}_i \approx y_i$ for $i = 1, \ldots, n$ – then the RSE will be small, and we can conclude that the model fits the data very well.

R^2 Statistic

The RSE provides an absolute measure of lack of fit of the model 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 alternativ 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 formuala

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

where $\text{TSS} = \sum(y_i - \overline{y})^2$ is the total sum of squares. 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, TSS - RSS measures the amount of variability in the response that is explained 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.

The R^2 statistic is a measure of the linear relationship between X and Y. Correlation is defined as

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

is also a measure of the linear relationship between X and Y. 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 the simple linear regression setting, $R^2 = r^2$.

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.

One option is to run three separate simple linear regressions, each of which uses a different medium as a predictor. However this approach is not entirely satisfactory. A better approach is to extend the simple linear regression model so that it can directly accommodate multiple predictors. We can do this by giving each predictor a separate slope coefficient in a single model. Suppose we have p distinct predictors, then the multiple linear regression model takes the form

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

where X_j represents the jth 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.

3.2.1 Estimating the Regression Coefficients

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

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p.$$

The parameters are estimated using the least squares approach. We choose $\beta_0, \ldots, \beta_p$ to minimize the sum of squared residuals

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

3.2.2 Some Important Questions

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

1. Is atleast one of the predictors $X_1, X_2, \ldots, X_p$ 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 ech of these questions in turn.

One: Is there a relationship between the response and predictors?

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 $\beta_1 = \beta_2 = \cdots = \beta_p = 0$. We use a hypothesis test to answer this question. We test the null hypothesis,

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

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,

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

If the linear model assumptions are correct then,

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

and that. provided H_0 is true

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

When there is no relationship between the response and predictors, the F-statistic woulld take on a value close to 1.

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.

Sometimes we want to test that a particular subset of q of the coefficients are zero. This corresponds to a null hypothesis

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

In this case we fit a second model that uses all of 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

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

Two: Deciding on Important Variables

The first step in a multiple regression analysis is to compute the F-statistic and to examine the

associated p-value.

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**.

We would like to perform variable selection by trying out a lot of different models, each containing a different subset of the predictors. How do we determine which model is the best? Various statistics can be used to judge the quality of the model. These include Mallow's C_p, Akaike information criterion (AIC), Bayesian information criterion (BIC), and adjusted R^2.

There are a total of 2^p models that contain subsets of p variables. 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 wit 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 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 the variables in the model, and remove the variable with the largeset 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.

- **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. 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.

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. 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. The R^2 statistic 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 more accurately. Thus, the R^2 statistic, which is also computed on the training data, must increase.

In addition to looking at the RSE and R^2 statistics, it can be useful to plot the data. Graphical summaries can reveal problems with a model that are not visible from numerical statistics.

Four: Predictions

Once we have fit the multiple regression model, in order to predict the response Y on the basis of a set of values for the predictors $X_1, X_2, \ldots, X_p$. However, there are three sorts of uncertainty associated with this prediction.

1. The coefficient estimates $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ are estimates for $\beta_0, \beta_1, \ldots, \beta_p$. 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 line

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

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

2. 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.

3. Even if we knew $f(X)$ – that is, even if we knew the true values for $\beta_0, \beta_1, \ldots, \beta_p$ – the response value cannot be predicted perfectly because of the random error ϵ in the model. How much

will Y vary from $\hat{Y}$? We use prediction intervals to answer this question.

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 are quantitative. But in practise, this is not necessarily the case; often some predictors are qualitative.

Predictors with Only Two Levels

If a qualitative predictor has only two levels, or possible values, then incorporating it into a regression model is very simple. We simply create a dummy variable that takes on two possible numerical values.

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.

3.3.2 Extensions of the Linear Model

The standard linear regression model 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 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 stats 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.

Removing the Additive Assumption

Consider the standard linear regression model with two variables,

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

According to the 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

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon.$$

The model can be rewritten as

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

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.

It is sometimes the case that an interaction term has a very small p-value, but the associated main effects do not. The hierarchial 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.

Non-Linear Relationships

The linear regression model 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.

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 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.

2. Correlation of Error terms

An important assumption of the linear regression model is that the error terms $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ are uncorrelated. 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}. 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.

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.

4. Outliers

   An outlier is a point for which y_i is far from the value predicted by the model. They can arise for a variety of reasons, such as incorrect recording of an observation during data collection. It is typical for an outlier that does not have an unusual predictor value to have little effect on the least squares fit.

5. High Leverage Points

   We just saw 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. High leverage observations tend to have a sizable impacton the estimated regression line. For this reason, it is important to identify high leverage observations.

   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,

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

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

6. Collinearity

   Collinearity refers to the situation in which two or more predictor variables are closely related to one another. 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.

Since collinearity reduces the accuracy of the estimates of the regression coefficients, it causes the standard error $\hat{\beta}_j$ to grow. 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 is reduced by collinearity.

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

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

where $R^2_{X_j|X_{-j}}$ is the R^2 from a regression of X_j onto all of the other predictors.

3.4

3.5 Comparison of Linear Regression with K-Nearest Neighbours

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. But parametric methods do have a disadvantage: by construction, they make strong assumptions about the form of $f(X)$.

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. One of the simplest and best-known non-parametric method is K-nearest neighbours regression (KNN) regression. The KNN regression is closely related to the KNN classifier. Given a value 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,

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

The optimal value for K will depend on the bias-variance tradeoff. A small value for K provides the most flexible fit, which will have low bias but high variance. In contrast, larger values of K provide a smoother and less variable fit; the prediciton in a region is an average of several points, and so changing one observation has a smaller effect.

The parametric approach will outperform non-parametric approach if the parametric form that has been selected is close to the true form of f.

4 Classification

The linear regression model assumes that the response variable Y is quantitative. But in many situations, the response variable is instead qualitative.

There are many possible classification techniques, or classifiers, that one might use to predict a qualitative response. Three of the most widely-used classifiers are: logistic regression, linear discriminant analysis and K-nearest neighbours.

4.1 An Overview of Classification

Just as in the regression setting, in the classification setting we have a set of training observations $(x_1, y_1), \ldots, (x_n, y_n)$ 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.

4.2 Why Not Linear Regression?

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:

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

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 $X_1, \ldots, X_p$. Unfortunately, this coding implies an ordering on the outcomes, putting drug overdose in between stroke and epileptic seizure. There is no particular reason that this needs to be the case.

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. 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 qualitative response, the situation is better.

4.3 Logistic Regression

Consider the Default data set, where the response dafualt falls into one of two categories, Yes or No. Rather than modeling this response Y directly, logistic regression models calculate the probability that Y belongs to a particular category. For example, the probability of default given balance can be written as

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

4.3.1 The Logistic Model

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

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

To fit the model, we use a method called maximum likelihood. The logistic function will always produce an S-shaped curve.

After a bit of manipulation, we find that

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

The quantity $p(X)/[1 - p(X)]$ is called the odds, and can take on any value between 0 and ∞.

By taking the logarithm of both sides, we arrive at

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

The left-hand side is called the log-odds or logit. In a logistic regression model, increasing X by one unit changes the log-odds by β_1, or equivalently it multiplies the odds by e^{β_1}. However, because the relationship between $p(X)$ and X 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.

4.3.2 Estimating the Regression Coefficients

The coefficients β_0 and β_1 are unknown, and must be estimated based on the available training data. 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, the more general method of maximum likelihood is preferred, since it has better statistical properties. We seek estimates for β_0 and β_1 such that the predicted probability $\hat{p}(x_i)$ of default for each individual, 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 $p(X)$, 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 the likelihood function:

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

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. In the linear regression setting, the least squares approach is in fact a special case of maximum likelihood.

4.3.3 Making Predictions

Once the coefficients have been estimaed, it is a simple matter to compute the probability of default for any given credit card balance. One can use qualitative predictors with the logistic regression model using the dummy variable approach.

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, we can generalize multiple logistic regression as follows:

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

where $X = (X_1, \ldots, X_p)$ are p predictors. The above equation can be rewritten as

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

We use the maximum likelihood method to estimate $\beta_0, \beta_1, \ldots, \beta_p$.

4.3.5 Logistic Regression for > 2 Response Classes

We sometimes wish to classify a response variable that has more than two classes.

4.4 Linear Discriminant Analysis

Logistic regresion involves directly modeling $\Pr(Y = k \mid X = x)$ using the logistic function, 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, and then use Baye's theorem to flip these around into estimates for $\Pr(Y = k \mid X = x)$.

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.

4.4.1 Using Baye's 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 kth class; this is the probability that a given observation is associated with the kth category of the response variable Y. Let $f_k(x) \equiv \Pr(X = x \mid Y = k)$ denote the density function of X for an observation that comes from the kth class. In other words, $f_k(x)$ is relatively large if there is a high probability that an observation in the kth class has $X \approx x$, and $f_k(x)$ is small if it is very unlikely that an observation in the kth class has $X \approx x$. Then Baye's theorem states that

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

We will use the abbreviation $p_k(X) = \Pr(Y = k \mid X)$. This suggests that instead of directly computing $p_k(X)$, we can simply plug in estimates of π_k and $f_k(X)$. In general, estimating π_k is easy if we have a random sample of Ys from the population: we simply compute the fraction of the training observations that belong the the kth 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 kth class, given the predictor value for that observation.

The Baye's 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. Therefore, if we can find a way to estimate $f_k(X)$, then we can develop a classifier that approximates the Baye's classifier.

4.4.2 Linear Discriminant Analysis for $p = 1$

For now we assume that $p = 1$, that is, we have only one predictor. We would like to obtain an estimate for $f_k(x)$ 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

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

where μ_k and σ_k^2 are the mean and variance parameters for the kth class. For now, let us further assume that $\sigma_1^2 = \cdots = \sigma_K^2$: that is, there is a shared variance term across all K classes, which for simplicity we can denote by σ^2. We find that

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

We can also equivalently assign the observation to the class for which

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

is largest.

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 $\mu_1, \ldots, \mu_K$, $\pi_1, \ldots, \pi_K$, and σ^2. The following estimates are used:

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

where n is the total number of training observations, and n_k is the number of training observations in the kth class. The estimate for μ_k is simply the average of all the training observations from the kth 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 $\pi_1, \ldots, \pi_K$, 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 kth class. In other words,

$$\hat{\pi}_k = n_k/n.$$

The LDA classifier assigns an observation $X = x$ to the class for which

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

is largest. The word linear in the classifier's name stems from the fact that the discriminant functions $\hat{\delta}_k(x)$ are linear functions of x.

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 $X = (X_1, X_2, \ldots, X_p)$ is drawn from a multivariate Gaussian distribution, with a class-specific mean vector and a common covariance matrix.

The multivariate Gaussian distribution assumes that each individual predictor follows a one-dimensional normal distribution, with some correlation between each pair of predictors. 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, and $\mathrm{Cov}(X) = \Sigma$ is the $p \times p$ covariance matrix of X. Formally, the multivariate Gaussian density is defined as

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

In the case of $p > 1$ predictors, the LDA classifier assumes that the observations in the kth 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. A little bit of algebra reveals

that the Baye's classifier assigns an observation $X = x$ to the class for which

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

is largest.

Once again, we need to estimate the unknown parameters $\mu_1, \ldots, \mu_k, \quad \pi_1, \ldots, \pi_K$, and Σ; the formulas are similar to those used in the one-dimensional case. To assign a new observation $X = x$, LDA classifies to the class for which $\hat{\delta}_k(x)$ is largest. Note that $\delta_k(x)$ 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 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. It is often of interest to determine which of these two types of errors are being made. A confusion matrix is a convenient way to display this information.

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.

Why does LDA 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. 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 a customer who does not default to the default class and vice-versa.

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

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

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.

The ROC curve is a popular graphic for simultaneously displaying the two types of errors for all possible thresholds. 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. ROC curves are useful for comparing different classifiers, since they take into account all possible thresholds.

As we have seen, varying the classifier threshold changes its true positive and false positive rate. These are also called the sensitivity and one minus the specificity of our classifier.

4.4.4 Quadratic Discriminant Analysis

As we have discussed, LDA assumes that the observations within each class are drawn from a multivariat 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 kth class is of the form $X \sim N(\mu_k, \Sigma_k)$, where Σ_k is a covariance matrix for the kth class. Under this assumption, the Bayes classifier assigns an observation $X = x$ to the class for which

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

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

Why would one prefer LDA to QDA, or vice-versa? The answer lies in the bias-variance tradeoff. When there are p predictors, then estimating a covariance matrix for each class, for a total of $Kp(p+1)/2$ 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.

4.5 A Comparison of Classification Methods

In this chapter, we have considered three different classification approaches: logistic regression, LDA and QDA. 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, the log odds is given by

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

where c_0 and c_1 are functions of μ_1, μ_2 and σ^2. We know that in logistic regression

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

Both of these 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.

The KNN method 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.

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.

5 Resampling Methods

Resampling methods are an indispensabel tool in modern statistics. They involve repeatedly drawing samples from a training data set and refitting a model of interest on each sample in order to obtain additional information about the fitted model.

5.1 Cross-Validation

We discussed the distinction between the test error rate and the training error rate. The test error rate is the average error that results from using a statistical learning method to predict the response on a new observation, that is, a measurement that was not used in training the method. Given a data set, the use of a statistical learning method is warranted if it results in a low test error. The test error can be easily calculated if a designated test set is available. In contrast, the training error can be easily calculated by applying the statistical learning method to the observations used in its training. The training error rate often is quite different from the test error rate, and in particular the former can dramatically underestimate the latter.

In the absence of a very large designated test set that can be usd to directly estimate the test error rate, a number of techniques can be used to estimate this quantity using the available training data. Some methods make a mathematical adjustment ot the training error rate in order to estimate the test error rate. We will consider a class of methods that estimate the test error rate by holding out a subset of the training observations from the fitting process, and then applying the statistical learning method to those held out observations.

5.1.1 The Validation Set Approach

Suppose that we would like to estimate the test error associated with fitting a particular statistical learning method on a set of observations. The validation set approach is a very simple strategy for this task. It involves randomly dividing the available set of observations into two parts, a training set and a validation set or hold-out set. The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set. The resulting validation set error rate, typically assessed using MSE in the case of a quantitative response, provides an estimate of the test error rate.

The validation set approach is conceptually simple and is easy to implement. But it has two potential drawbacks:

1. The validation estimate of the test error rate can be highly variable, depending on precisely

which observations are included in the training set and which observations are included in the validation set.

2. In the validation approach, only a subset of the observations, those that are included in the training set rather than in the validation set, are used to fit the model. Since statistical methods tend to perform worse when trained on fewer observations, this suggests that the validation set error rate may tend to overestimate the test error rate for the model fit on the entire data set.

5.1.2 Leave-One-Out Cross-Validation

Leave-one-out cross-validation (LOOCV) is closely related to the validation set approach, but it attempts to address that method's drawbacks.

Like the validation set approach, LOOCV involves splitting the set of observations into two parts. However, instead of creating two subsets of comparable size, a single observation (x_1, y_1) is used for the validation set, and the remaining observations $\{(x_2, y_2), \ldots, (x_n, y_n)\}$ make up the training set. The statistical learning method is fit on the $n-1$ training observations, and a prediction $\hat{y}_1$ is made for the excluded observation, using its value x_1. Since (x_1, y_1) was not used in the fitting process, $\text{MSE}_1 = (y_1 - \hat{y}_1)^2$ provides an approximately unbiased estimate for the test error. But even though MSE_1 is unbiased for the test error, it is a poor estimate because it is highly variable, since it is based upon a single observation (x_1, y_1).

We can repeat the procedure by selecting (x_2, y_2) for the validation data, training the statistical learning procedure on the $n-1$ observations $\{(x_1, y_1), (x_3, y_3), \ldots, (x_n, y_n)\}$, and computing $\text{MSE}_2 = (y_2 - \hat{y}_2)^2$. Repeating this approach n times produces n squared errors, $\text{MSE}_1, \ldots, \text{MSE}_n$. The LOOCV estimate for the test MSE is the average of these n test error estimates:

$$\text{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \text{MSE}_i.$$

LOOCV has a couple of major advantages over the validation set approach. First, it has far less bias. In LOOCV, we repeatedly fit the statistical learning method using training sets that contain $n-1$ observations, almost as many as are in the entire data set. This in contrast to the validation set approach, in which the training set is typically around half the size of the original data set. Consequently, the LOOCV approach tends not to overestimate the test error rate as much as the validation set approach does. Second, in contrast to the validation approach which will yield different results when applied repeatedly due to randomness in the training/validation set splits, performing LOOCV multiple times will always yield the same results: there is no randomness in the

training/validation set splits.

LOOCV has the potential to be expensive to implement, since the model has to be fit n times. This can be very time consuming if n is large, and if each individual model is slow to fit. With least squares linear or polynomial regression, an amazing shortcut makes the cost of LOOCV the same as that of a single model fit! The following formula holds:

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2,$$

where $\hat{y}_i$ is the ith fitted value from the original least squares fit, and h_i is the leverage. This is like the ordinary MSE, except the ith residual is divided by $1 - h_i$. The leverage lies between $1/n$ and 1, and reflects the amount that an observation influences its own fit. Hence the residuals for high-leverage points are inflated in this formula by exactly the right amount for this equality to hold.

LOOCV is a very general method, and can be used with any kind of predictive modeling.

5.1.3 k-Fold Cross-Validation

An alternative to LOOCV is k-fold CV. This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining $k - 1$ folds. The mean squared error, MSE_1, is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of the test error, $\mathrm{MSE}_1, \mathrm{MSE}_2, \ldots, \mathrm{MSE}_k$. The k-fold CV estimate is computed by averaging these values,

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i.$$

It is not hard to see that LOOCV is a special case of k-fold CV in which k is set to equal n. In practice, one typically performs k-fold CV using $k = 5$ or $k = 10$.

When we examine real data, we do not know the true test MSE, and so it is difficult to determine the accuracy of the cross-validation estimate. However, if we examine simulated data, then we can compute the true test MSE, and can thereby evaluate the accuracy of our cross-validation results.

When we perform a cross-validation, our goal might be to determine how well a given statistical learning procedure can be expected to perform on independent data; in this case, the actual estimate of the test MSE is of interest. But at other times we are interested only in the location of the minimum point in the estimated test MSE curve. This is because we might be performing cross-validation on

a number of statistical learning methods, or on a single method using different levels of flexibility, in order to identify the method that results in the lowest test error. For this purpose, the location of the minimum point in the estimated test MSE curve is important, but the actual value of the estimated test MSE is not.

5.1.4 Bias-Variance Trade-Off for k-Fold Cross-Validation

The k-fold CV has a computational advantage to LOOCV. But putting computational issues aside, a less obvious but potentially more important advantage of k-fold CV is that it often gives more accurate estimates of the test error rate than does LOOCV. This has to do with the bias-variance trade-off.

The validation set approach can lead to overestimates of the test error rate, since in this approach te training set used to fit the statistical learning method contains only half the observations of the entire data set. Using this logic, it is not hard to see that LOOCV will give approximately unbiased estimates of the test error, since each training set contains $n - 1$ observations, which is almost as many as the number of observations in the full data set. And performing k-fold CV for, say, $k = 5$ or $k = 10$ will lead to an intermediate level of bias, since each training set contains $(k - 1)n/k$ observations, fewer than in the LOOCV approach, but substantially more than in the validation set approach. Therefore, from the perspective of bias reduction, it is clear that LOOCV is to be preferred to k-fold CV.

However, we know that bias is not the only source for concern in an estimating procedure; we must also consider the procedure's variance. It turns out that LOOCV has higher variance than does k-fold CV with $k < n$. When we perform LOOCV, we are in effect averaging the outputs of n fitted models, each of which is trained on an almost identical set of observations; therefore, these outputs are highly correlated with each other. In contrast, wehn we perform k-fold CV with $k < n$, we are averaging the outputs of k fitted models that are somewhat less correlated with each other, since the overlap between the training sets in each model is smaller. Since the mean of mainy highly correlated quantities has higher variance than does the mean of many quantities that are not as highly correlated, the test error estimate resulting from LOOCV tends to have higher variance than does the test error estimate resulting from k-fold CV.

To summarize, there is a bias-variance trade-off associated with the choice of k in k-fold cross-validation. Typically, given these considerations, one performs k-fold cross-validation using $k = 5$ or $k = 10$, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance.

5.1.5 Cross-Validation on Classification Problems

Cross-validation can also be a very useful approach in the classification setting when Y is qualitative. In this setting, cross-validation works just as described earlier, except that rather than using MSE to quantify test error, we instead use the number of misclassified observations. For instance, in the Classification setting, the LOOCV error rate takes the form

$$\mathrm{CV}_{(n)} = \frac{1}{n} = \sum_{i=1}^{n} \mathrm{Err}_i,$$

where $\mathrm{Err}_i = I(y_i \neq \hat{y}_i)$. The k-fold CV error rate and validation set error rates are defined analogously.

5.2 The Bootstrap

The bootstrap is a widely applicable and extremely powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method. As a simple example, the bootstrap can be used to estimate the standard errors of the coefficients from a linear regression fit. The power of the bootstrap lies in the fact that it can be easily applied to a wide range of statistical learning methods, including some for which a measure of variability is otherwise difficult to obtain and is not automatically output by statistical software.

Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y, respectively, where X and Y are random quantities. We will invest a fraction α of our money in X, and will invest the remaining $1 - \alpha$ in Y. Since there is variability associated with the returns on these two assets, we wish to choose α to minimize the total risk, or variance, of our investment. In other words, we want to minimize $\mathrm{Var}(\alpha X + (1 - \alpha)Y)$. The value that minimizes the risk is given by

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}},$$

where $\sigma_X^2 = \mathrm{Var}(X), \sigma_Y^2 = \mathrm{Var}(Y)$, and $\sigma_{XY} = \mathrm{Cov}(X, Y)$.

In reality, the quantities σ_X^2, σ_Y^2, and σ_{XY} are unknown. We can compute estimates for these quantities $\hat{\sigma}_X^2, \hat{\sigma}_Y^2$, and $\hat{\sigma}_{XY}$, using a data set that contains past measurements for X and Y. We can then estimate the value of α that minimizes the variance of our investment using

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}.$$

It is natural to want to quantify the accuracy of our estimate of α. To estimate the standard deviation of $\hat{\alpha}$, we repeat the process of simulating 100 paired observations of X and Y, and estimating α using the above equation 1000 times. We thereby obtained 1000 estimates for α, which we can call $\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_{1000}$.

In practice, however, the procedure for estimating $\mathrm{SE}(\hat{\alpha})$ outlined above cannot be applied, because for real data we cannot generate new samples from the original population. However, the bootstrap method approach allows us to use a computer to emulate the process of obtaining new sample sets, so that we can estimate the variability of $\hat{\alpha}$ without generating additional samples. Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set.

Let us define a simple data set, which we call Z, that contains only $n = 3$ observations. We randomly select n observations from the data set in order to produce a bootstrap data set, Z^{*1}. The sampling is performed with replacement, which means that the same observation can occur more than once in the bootstrap data set. In this example, Z^{*1} contains the third observation twice, the first observation once, and no instances of the second observation. Note that if an observation is contained in Z^{*1}, then both its X and Y values are included. We can use Z^{*1} to produce a new bootstrap estimate for α, which we call $\hat{\alpha}^{*1}$. This procedure is repeated B times for some large value of B, in order to produce B different bootstrap data sets, $Z^{*1}, Z^{*2}, \ldots, Z^{*B}$, and B corresponding α estimates, $\hat{\alpha}^{*1}, \hat{\alpha}^{*2}, \ldots, \hat{\alpha}^{*B}$. We can compute the standard error of these bootstrap estimates using the formula

$$\mathrm{SE}_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\alpha}^{*r} - \frac{1}{B}\sum_{r'=1}^{B}\hat{\alpha}^{*r'}\right)^2}.$$

6 Linear Model Selection and Regularization

In the regression setting, the standard linear model

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$$

is commonly used to describe the relationship betwee a response Y and a set of variables $X_1, X_2, \ldots, X_p$. One typically fits this model using least squares.

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. 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 $n >> p$, 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. We will 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. We will discuss three important classes of methods.

- 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 shrunken towards zero relative to the least squares estimate. 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.

6.1 Subset Selection

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 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.

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.

Algorithm: 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 $k = 1, 2, \ldots, p$:

   (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 $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

In the above algorithm, 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 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. Therefore, in Step 3, we use cross-validated prediction error, C_p, BIC, or adjusted R^2 in order to select among $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_p$.

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, we instead use the deviance, a measure the 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.

The best subset selection 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 1000 possible models to be considered, and if $p = 20$, then there are over one million possibilities.

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 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 algorithm is given below.

Algorithm: Forward stepwise selection

1. Let $\mathcal{M}_0$ denote the null model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
   (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 $\mathcal{M}_0, \ldots, \mathcal{M}_p$ 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 kth iteration, for $k = 0, \ldots, p-1$. This amounts to a total of $1 + \sum_{k=0}^{p-1}(p-k) = 1 + p(p+1)/2$ models.

In Step 2(b) of the algorithm, we must identify the best model from among those $p-k$ that augment $\mathcal{M}_k$ with one additional predictor. We can do this simply by 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 number of variables. This is more challenging.

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.

Forward stepwise selection can be applied even in the high-dimensional setting where $n < p$; however, in this case, it is possible to construct sub-models $\mathcal{M}_0, \ldots, \mathcal{M}_{n-1}$ 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 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. The algorithm is given below.

Algorithm: Backward stepwise selection

1. Let $\mathcal{M}_p$ denote the full model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   (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 $\mathcal{M}_0, \ldots, \mathcal{M}_p$ 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.

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

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 reataining 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. 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. 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 number 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 vialidation set approach or a cross-validation approach.

C_p, AIC, BIC, and Adjusted R^2

The training set MSE is generally an underestimate of the test MSE. 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 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 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.

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

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

where $\hat{\sigma}^2$ is an estimate of the variance of the error ϵ associated with each response measurement. Typically $\hat{\sigma}^2$ is estimated using the full model containing all predictors. 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. The penalty increases as the number of predictors in the model increases; this is intended to adjust for the corresponding decrease in training RSS. The C_p statistic tends to take on a small value for models with a low test error, so when determining wwhich of a set of models is best, we choose the model with the lowest C_p value.

The AIC criterion is defined for a large class of models fit by maximum likelihood. In the case of the model 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.

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

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

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. 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.

The adjusted R^2 statistic is another popular approach for selecting among a set of models that contain different number of variables. The usual R^2 is defined as $1-\text{RSS}/\text{TSS}$, where $\text{TSS} = \sum(y_i-\overline{y})^2$ 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

$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}.$$

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.

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. 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 or hard to estimate the error variance σ^2.

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.

6.2 Shrinkage Methods

The subset selection methods discussed previously 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 constrains or regularizes the coefficient estimates, or equivalently, that shrinks the coefficient estimates toward zero. It turns out that srhinking 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

The least squares fitting procedure estimates $\beta_0, \beta_1, \ldots, \beta_p$ 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

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2,$$

where $\lambda \geq 0$ is a tuning parameter, to be determined separately. The above equation 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 $\beta_1, \ldots, \beta_p$ 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 $\lambda \to \infty$, 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.

Note that, the shrinkage penalty is applied to $\beta_1, \ldots, \beta_p$, 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 $x_{i1} = x_{i2} = \cdots = x_{ip} = 0$. If we assume that the variables, that is, the columns of the data matrix X, have been centered to have mean zero before ridge regression is performed, then the estimated intercept will take the form $\hat{\beta}_0 = \sum_{i=1}^n y_i/n$.

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.

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, 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. 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. The computations required, 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 $\lambda \sum \beta_j^2$ 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. 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

$$\sum_{i=1}^{n} \left(y_i - \beta_0 \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \mid \beta_j \mid = \text{RSS} + \lambda \sum_{j=1}^{p} \mid \beta_j \mid .$$

We see that lasso and ridge regression have similar formulations. The only difference is that the β_j^2 term in the ridge regression penalty has been replaced by $\mid \beta_j \mid$ in the lasso penalty. The lasso uses an l_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, 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.

Comparing the Lasso and Ridge Regression

Which method leads to better prediction accuracy? 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.

Bayesian Interpretation for Ridge Regression and the Lasso

One can view ridge regression and the lasso through a Bayesian lens. A Bayesian viewpoint for regression assumese that the coefficient vector β has some prior distribution, say $p(\beta)$, where $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$. The likelihood of the data can be written as $f(Y \mid X, \beta)$, where $X = (X_1, \ldots, X_p)$. Multiplying the prior distribution by likelihood gives us the posterior distribution, which takes the form

$$p(\beta \mid X, Y) \propto f(Y \mid X, \beta)p(\beta \mid X) = f(Y \mid X, \beta)p(\beta),$$

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,

$$Y = \beta_0 + X_1\beta_1 + \cdots + X_p\beta_p + \epsilon,$$

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.
- 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.

6.2.3 Selecting the Tuning Parameter

Implementing the ridge regression and the lasso requires a method for selecting a value for the tuning parameter λ. 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. 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.

6.3 Dimension Reduction Methods

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 $Z_1, Z_2, \ldots, Z_M$ represent $M < p$ linear combinations of our original p predictors. That is,

$$Z_m = \sum_{j=1}^{p} \Phi_{jm} X_j$$

for some constants $\Phi_{1m}, \Phi_{2m}, \ldots, \Phi_{pm}, \quad m = 1, \ldots, M$. We can then fit the linear regression model

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i, \quad i = 1, \ldots, n,$$

using least squares. The regression coefficients are given by $\theta_0, \theta_1, \ldots, \theta_M$. If the constants $\Phi_{1m}, \Phi_{2m}, \ldots, \Phi_{pm}$ are chosen wisely, then such dimension reduction approaches can often outperform least squares regression.

The term dimension reduction comes from the fact that this approach reduces the problem of estimating the $p + 1$ coefficients $\beta_0, \beta_1, \ldots \beta_p$ to the simpler problem of estimating the $M + 1$ coefficients $\theta_0, \theta_1, \ldots, \theta_M$, where $M < p$. In other words, the dimension of the problem has been reduced from $p + 1$ to $M + 1$.

Notice that

$$\sum_{m=1}^{M} \theta_m z_{im} = \sum_{m=1}^{M} \theta_m \sum_{j=1}^{p} \Phi_{jm} x_{ij} = \sum_{j=1}^{p} \sum_{m=1}^{M} \theta_m \Phi_{jm} x_{ij} = \sum_{j=1}^{p} \beta_j x_{ij},$$

where

$$\beta_j = \sum_{m=1}^{M} \theta_m \Phi_{jm}.$$

Hence this can be thought of as a special case of the original linear regression model. Dimension reduction serves to constrain the estimated β_j coefficients, since now they must take the above form. 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 the form poses no constraints.

All dimension reduction methods work in two steps. First, the transformed predictors $Z_1, Z_2, \ldots, Z_m$ are obtained. Second, the model is fit using these M predictors. The choice of $Z_1, Z_2, \ldots, Z_M$, or equivalently, the selection of the Φ_{jm}'s, can be achieved in different ways. 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. We will discuss its use as a dimension reduction technique for regression. PCA is a technique for reducing the dimension of a $n \times p$ data matrix X. The first principal component direction of the data is that along which the observations vary the most. For instance, consider the data set which contains 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 first principal component can be summarized mathematically given by the formula

$$Z_1 = 0.839 \times (\text{pop} - \overline{\text{pop}}) + 0.544 \times (\text{ad} - \overline{\text{ad}}).$$

Here $\Phi_{11} = 0.839$ and $\Phi_{21} = 0.544$ are principal component loadings, which define the direction referred to above. In the above formula, $\overline{\text{pop}}$ 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. The two above 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 Z_1. For instance,

$$z_{i1} = 0.839 \times (\text{pop}_i - \overline{\text{pop}}) + 0.544 \times (\text{ad}_i - \overline{\text{ad}}).$$

The values of $z_{11}, \ldots, z_{n1}$ are known as the principal component scores.

There is also another interpretation for PCA: the first principal component vector defines the line that is as close as possible to the data.

In general, one can construct up to p distinct principal componenets. 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. It turns out that the zero correlation condition of Z_1 and Z_2 is equivalent to the condition that the direction must be perpendicular, or orthogonal, to the first principal component direction.

With two-dimensional data, we can construct at most two principal components. However, if we had other predictors, then additional components could be constructed. They would successively maximize variance, subject to the constraint of being uncorrelated with the preceeding components.

The Principal Components Regression Approach

The principal components regression (PCR) approach involves constructing the first M principal components $Z_1, \ldots, Z_M$, 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 combonents 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 $X_1, \ldots, X_p$ show the most variation are the directions that are associated with Y. While this assumption is not guaranteed 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 $Z_1, \ldots, Z_M$ will lead to better results than fitting a least squares model to $X_1, \ldots, X_p$, since most or all of the information in the data that relates to the response is contained in $Z_1, \ldots, Z_M$, and by estimating only $M \ll p$ coefficients we can mitigate overfitting.

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 is a linear combination of all p of the original features.

In PCR, the number of principal components, M, is typically chosen by cross-validation. When performing PCR, it is generally recommended standardizing each predictor, 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.

6.3.2 Partial Least Squares

The PCR approach that we just described involves identifying linear combinations, or directions, that best represent the predictors $X_1, \ldots, X_p$. 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.

Partial least squares (PLS) is a supervised alternative to PCR. Like PCR, PLS is a dimension reduction method, which first identifies a new set of features $Z_1, \ldots, Z_M$ that are linear combinations of the original features, and then fits a linear model via least squares by 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.'

After standardizing the p predictors, PLS computes the first direction Z_1 by setting each Φ_{j1} equal to the coefficient from the simple linear regression of Y onto X_j. 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.

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 $Z_1, \ldots, Z_M$. Finally, at the end of this procedure, we use least squares to fit a linear model to predict Y using $Z_1, \ldots, Z_M$ in exactly the same fashion as for PCR.

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.

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.

6.4.2 What Goes Wrong in High Dimension?

When the number of features p is as large, or larger than, the number of observations n, least squares cannot be performed. Regardless of whether or not there truly is a relationship between the features and the response, least squares will yield a set of coefficients that result in a perfect fit to the data, such that the residuals are zero. This is problematic beacuse this perfect fit will almost certainly lead to overfitting of the data. 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.

6.4.3 Regression in High Dimensions

It turns out that many of the methods 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.

There are 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.
3. The test error tends to increase as the dimensionality of the problem increases, unless 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. 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 the 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 problem, exacerbating the risk of overfitting without any potential upside in terms of improved test set error.

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 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. 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.

It is 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 uuse 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.

7 Moving Beyond Linearity

So far, we have mostly focused on linear models. They are relatively simple to describe and implement, and have advantages over other approaches in terms of interpretation and inference. However, standard linear regression can have significant limitations in terms of predictive power. This is because the linearity assumption is almost always an approximation, and sometimes a poor one.

- Polynomial regression extends the linear model by adding extra predictors, obtained by raising each of the original predictors to a power. For example, a cubic regression uses three variables, X, X^2, and X^3, as predictors. This approach provides a simple way to provide a non-linear fit to data.
- Step functions cut the range of a variable into K distinct regions in order to produce a qualitative variable. This has the effect of fitting a piecewise constant function.
- Regression splines are more flexible than polynomials and step functions, and in fact are an extension of the two. They involve dividing the range of X into K distinct regions. Within each region, a polynomial function is fit to the data. However, these polynomials are constrained so that they join smoothly at the region boundaries, or knots. Provided that the interval is divided into enough regions, this can produce an extremely flexible fit.
- Smoothing splines are similar to regression splines, but arise in a slightly different situation. Smoothing splines result from minimizing a residual sum of squares criterion subject to a smoothness penalty.
- Local regression is similar to spines, but differs in an important way. The regions are allowed to overlap, and indeed they do so in a very smooth way.
- Generalized additive models allow us to extend the methods above to deal with multiple predictors.

7.1 Polynomial Regression

Historically, the standard way to extend linear regression to settings in which the relationship between the predictors and the response is non-linear has been to replace the standard linear model

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

with a polynomial function

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \cdots + \beta_d x_i^d + \epsilon_i,$$

where ϵ_i is the error term. This approach is known as polynomial regression. For large enough degree d, a polynomial regression will allow us to produce an extremely non-linear curve. The coefficients can be easily estimated using least squares linear regression because this is just a standard linear model with predictors $x_i, x_i^2, x_i^3, \ldots, x_i^d$. Generally speaking, it is unusual to use d greater than 3 or 4 because for large values of d, the polynomial curve can become overly flexible and can take on some very strange shapes. This is especially true near the boundary of the X variable.

7.2 Step Functions

Using polynomial functions of the features as predictors in a linear model imposes a global structure on the non-linear function of X. We can instead use step functions in order to avoid imposing such a global structure. Here we break the range of X into bins, and fit a different constant in each bin. This amounts to converting a continuous variable into an ordered categorical variable.

In greater detail, we create cutpoints $c_1, c_2, \ldots, c_K$ in the range of X, and then construct $K + 1$ new variables

$$\begin{aligned}
C_0(X) &= I(X < c_1), \\
C_1(X) &= I(c_1 \leq X \leq c_2), \\
C_2(X) &= I(c_2 \leq X \leq c_3), \\
&\vdots \\
C_{K-1}(X) &= I(c_{K-1} \leq X \leq c_K), \\
C_K(X) &= I(c_K \leq X),
\end{aligned}$$

where $I(\cdot)$ is an indicator function that returns a 1 if the condition is true, and returns a 0 otherwise. These are sometimes called dummy variables. Notice that for any value of X, $C_0(X) + C_1(X) + \cdots + C_K(X) = 1$ since X must be in exactly one of $K + 1$ intervals. We then use least squares to fit a linear model using $C_1(X), C_2(X), \ldots, C_K(X)$ as predictors:

$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \cdots + \beta_K C_K(x_i) + \epsilon_i.$$

For a given value of X, at most one of $C_1, C_2, \ldots, C_K$ can be non-zero.

7.3 Basis Functions

Polynomial and piecewise-constant regression models are in fact special cases of a basis function approach. The idea is to have at hand a family of functions or transformations that can be applied to a variable X: $b_1(X), b_2(X), \ldots, b_K(X)$. Instead of fitting a linear model in X, we fit the model

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_K b_K(x_i) + \epsilon_i.$$

Note that the basis functions $b_1(\cdot), b_2(\cdot), \ldots, b_K(\cdot)$ are fixed and known. For polynomial regression, the basis functions are $b_j(x_i) = x_i^j$, and for piecewise constant functions they are $b_j(x_i) = I(c_j \leq x_i \leq c_{j+1})$. We can use least squares to estimate the unknown regression coefficients. We can use wavelets or Fourier series to construct basis functions.

7.4 Regression Splines

Now we discuss a flexible class of basis functions that extends upon the polynomial regression and piecewise constant regression approaches.

7.4.1 Piecewise Polynomials

Instead of fitting a high-degree polynomial over the entire range of X, piecewise polynomial regression involves fitting separate low-degree polynomials over different regions of X. For example, a piecewise cubic polynomial works by fitting a cubic regression model of the form

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i,$$

where the coefficients $\beta_0, \beta_1, \beta_2$, and β_3 differ in different parts of the range of X. The points where the coefficients change are called knots.

Using more knots leads to a more flexible piecewise polynomial. In general, if we place K different knots throughout the range of X, then we will end up fitting $K+1$ different cubic polynomials. Note that we do not need to use a cubic polynomial. For example, we can instead fit piecewise linear functions.

7.4.2 Constraints and Splines

If the fitted curve is just too flexible, we can fit a piecewise polynomial under the constraint that the fitted curve must be continuous. Each constraint that we impose on a piecewise polynomial effectively frees ip one degree of freedom, by reducing the complexity of the resulting piecewise polynomial fit.

7.4.3 The Spline Basis Representation

How can we fit a piecewise degree-d polynomial under the constraint that it (and possibly its first $d-1$ derivatives) be continuous? It turns out that we can use the basis model to represent a regression spline. A cublic spline with K knots can be modeled as

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i),$$

for an appropriate choice of basis functions $b_1, b_2, \ldots, b_{K+3}$. The model can then be fit using least squares.

There are many equivalent ways to represent cublic splines using different choices of basis functions. The most direct way to represent a cubic spline is to start off with a basis for a cubic polynomial, namely, x, x^2, x^3, and then add one truncated power basis function per knot. A truncated power basis is defined as

$$h(x, \xi) = (x - \xi)_+^3 = \begin{cases} (x-\xi)^3 & x > \xi \\ 0 & \text{otherwise,} \end{cases}$$

where ξ is the knot.

In order to fit a cubic spline to a data set with K knots, we perform least squares regression with an intercept and $3 + K$ predictors, of the form $X, X^2, X^3, h(X, \xi_1), h(X, \xi_2), \ldots, h(X, \xi_K)$, where $\xi_1, \ldots, \xi_K$ are the knots. This amounts to estimating a total of $K + 4$ regression coefficients; for this reason, fitting a cubic spline with K knots uses $K + 4$ degrees of freedom.

Unfortunately, splines can have high variance at the outer range of the predictors, that is, when X takes on either a very small or very large value. A natural spline is a regression spline with additional boundary contraints: the function is required to be linear at the boundary (in the region where X is smaller than the smallest knot, or larger than the largest knot). This additional constraint means that natural splines generally produce more stable estimates at the boundaries.

7.4.4 Choosing the Number and Locations of the Knots

When we fit a spline, where should we place the knots? The regression spline is most flexible in regions that contain a lot of knots, because in those regions the polynomial coefficients can change rapidly. Hence, one option is to place more knots in places where we feel the function might vary most rapidly, and to place fewer knots where it seems more stable. In practice it is common to place knots in a uniform fashion. One way to do this is to specify the desired degrees of freedom, and then have the software automatically place the corresponding number of knots at uniform quantiles of the data.

How many degrees of freedom should our spline contain? One option is to try out different numbers of knots and see which produces the best looking curve. A somewhat more objective approach is to use cross-validation. With this method, we remove a portion of the data, fit a spline with a certain number of knots to the remaining data, and then use the spline to make predictions for the held-out portion. We repeat this process multiple times until each observation has been left out once, and then compute the overall cross-validated RSS. This procedure can be repeated for different numbers of knots K. Then the value of K giving the smallest RSS is chosen.

7.4.5 Comparison to Polynomial Regression

Regression splines often give superior results to polynomial regression. This is because unlike polynomials, which must use a high degree to produce flexible fits, splines introduce flexibility by increasing the number of knots but keeping the degree fixed. Generally, this approach produces more stable estimates. Splines also allow us to place more knots, and hence flexibility, over regions where the function f seems to be changing rapidly, and fewer knots where f appears more stable.

7.5 Smoothing Splines

7.5.1 An Overview of Smoothing Splines

In the last section we discuessed regression splines, which we create by specifying a set of knots, producing a sequence of basis functions, and then using least squares to estimate the spline coefficients. We now introduce a somewhat different approach that also produces a spline.

In fitting a smooth curve to a set of data, what we really want to do is find some function, say $g(x)$, that fits the observed data well: that is, we want RSS $= \sum_{i=1}^{n}(y_i - g(x_i))^2$ to be small. However, there is a problem with this approach. If we don't put any constraints on $g(x_i)$, then we can always make RSS zero simply by choosing g such that it interpolates all of the y_i. What we

really want is a function g that makes RSS small, but that is also smooth.

There are a number of ways to do this. A natural approach is to find the function g that minimizes

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2\, dt$$

where λ is a nonnegative tuning parameter. The function g that minimizes the above expression is known as a smoothing spline.

The equation takes the "Loss + Penalty" formulation that we encounter in the context of ridge regression and the lasso. The term $\sum_{i=1}^{n}(y_i - g(x_i))^2$ is a loss function that encourages g to fit the data well, and the term $\lambda \int g''(t)^2\, dt$ is a penalty term that penalizes the variability in g. The second derivative of a function is a measure of its roughness: it is large in absolute value if $g(t)$ is very wiggly near t, and it is close to zero otherwise. The expression $\int g''(t)^2\, dt$ is simply a measure of the total change in the function $g'(t)$, over its entire range. If g is very smooth, then $g'(t)$ will be close to constant and $\int g''(t)^2\, dt$ will take on a small value. Conversely, if g is jumpy and variable then $g'(t)$ will vary significantly and $\int g''(t)^2\, dt$ will take on a large value. Therefore, $\lambda \int g''(t)^2\, dt$ encourages g to be smooth. The larger the value of λ, the smoother g will be.

When $\lambda = 0$, then the penalty term has no effect, and so the function g will be very jumpy and will exactly interpolate the training observations. When $\lambda \to \infty$, g will be perfectly smooth, it will just be a straight line that passes as closely as possible to the training points. In this case, g will be the linear least squares line, since the loss function amounts to minimizing the residual sum of squares. For an intermediate value of λ, g will approximate the training observations but will be somewhat smooth. The value of λ controls the bias-variance trade-off of the smoothing spline.

The function $g(x)$ that minimizes the above equation can be shown to have some special properties: it is a piecewise cubic polynomial with knots at the unique values of $x_1, \ldots, x_n$, and continuous first and second derivatives at rach knot. Furthermore, it is linear in the region outside of the extreme knots.

7.5.2 Choosing the Smoothing Parameter λ

A smoothing spline is simply a natural cubic spline with knots at every unique value of x_i. It might seem that a smoothing spline will have far too many degrees of freedom, since a knot at each data point allows a great deal of flexibility. But the tuning parameter λ controls the roughness of the smoothing spline, and hence the effective degrees of freedom. As λ increases from 0 to ∞, the effective degrees of freedom, which we write df_λ, decreases from n to 2.

Degrees of freedom refer to the number of free parameters, such as the number of coefficients fit in a polynomial or cubic spline. Although a smoothing spline has n parameteres and hence n nominal degrees of freedom, these n parameters are heavily constrained or shrunk down. Hence df_λ is a measure of the flexibility of the smoothing spline, the higher it is, the more flexible the smoothing spline. The definition of effective degrees of freedom is somewhat technical. We can write

$$\hat{g}_\lambda = S_\lambda y,$$

where $\hat{g}$ is the solution to the main equation for a particular choice of λ, that is, it is a n-vector containing the fitted values of the smoothing spline at the training points $x_1, \ldots, x_n$. The above equation indicates that the vector of fitted values when applying a smoothing spline to the data can be written as a $n \times n$ matrix S_λ times the response vector y. Then the effective degrees of freedom is defined to be

$$df_\lambda = \sum_{i=1}^{n} \{S_\lambda\}_{ii},$$

the sum of the diagonal elements of the matrix S_λ.

In fitting a smoothing splines, we do not need to select the number or location of the knots, there will be a knot at each training observation, $x_1, \ldots, x_n$. Instead, we have another problem: we need to choose the value of λ. One possible solution to this problem is cross-validation. We can find the value of λ that makes the cross-validated RSS as small as possible. The LOOCV can be computed very efficiently for smoothing splines, with esentially the same cost as computing a single fit, using the following formula:

$$\text{RSS}_{cv}(\lambda) = \sum_{i=1}^{n} (y_i - \hat{g}_\lambda^{(-i)}(x_i))^2 = \sum_{i=1}^{n} \left[\frac{y_i - \hat{g}_\lambda(x_i)}{1 - \{S_\lambda\}_{ii}} \right]^2.$$

The notation $\hat{g}_\lambda^{(-i)}(x_i)$ indicates the fitted value for this smoothing spline evaluated at x_i, where the fit uses all of the training observations except for the ith observation (x_i, y_i). In contrast, $\hat{g}_\lambda(x_i)$ indicates the smoothing spline function fit to all of the training observations and evaluated at x_i. We can compute each of these leave-one-out fits using only $\hat{g}_\lambda$, the original fit to all of the data.

7.6 Local Regression

Local regression involves computing the fit at a target point x_0 using only the nearby training observations. It is sometimes referred to as a memory-based procedure, because like nearest-

neighbours, we need al the training data each time we wish to compute a prediction.

Algorithm: Local Regression At $X = x_0$

1. Gather the fraction $s = k/n$ of training points whose x_i are closest to x_0.
2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighbourhood, so that the point furthest to x_0 has weight zero, and the closest has the highest weight. All but these k nearest neighbours get weight zero.
3. Fit a weighted least squares regression of the y_i on the x_i using the aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2.$$

4. The fitted value at x_0 is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

7.7 Generalized Additive Models

Generalized additive models (GAMs) provide a general framework for extending a standard linear model by allowing non-linear functions of each of the variables, while maintaining additivity.

7.7.1 GAMs for Regression Problems

A natural way to extend the multiple linear regression model

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i$$

in order to allow for non-linear relationships between each feature and the response is to replace each linear component $\beta_j x_{ij}$ with a smooth non-linear function $f_j(x_{ij})$. We would then write the model as

$$\begin{aligned} y_i &= \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i \\ &= \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i. \end{aligned}$$

This is an example of a GAM. It is called an additive model because we calculate a separate f_j for each X_j, and then add together all of their contributions.

Pros and Cons of GAMs

- GAMs allow us to fit a non-linear f_j to each X_j, so that we can automatically model non-linear relationships that standard linear regression will miss. This means that we do not need to manually try out many different transformations on each variable individually.
- The non-linear fits can potentially make more accurate predictions for the response Y.
- Because the model is additive, we can still examine the effect of each X_j on Y individually while holding all of the other variables fixed. Hence if we are interested in inference, GAMs provide a useful representation.
- The smoothness of the function f_j for the variable X_j can be summarized via degrees of freedom.
- The main limitations of GAMs is that the model is restricted to be additive. With many variables, important interactions can be missed. However, as with linear regression, we can manually add interaction terms to the GAM model by including additional predictors of the form $X_j \times X_k$. In addition we can add low-dimensional interaction functions of the form $f_{jk}(X_j, X_k)$ into the model; such terms can be fit using two-dimensional smoothers such as local regression, or two-dimensional splines.

For fully general models, we have to look for even more flexible approaches such as random forests and boosting. GAMs provide a useful compromise between linear and fully nonparametric models.

7.7.2 GAMs for Classification Problems

GAMs can also be used in situations where Y is qualitative. Here we will assume Y takes on values zero or one, and let $p(X) = \Pr(Y = 1 \mid X)$ be the conditional probability that the response equals one. Recall the logistic regression model:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p.$$

This logit is the log of the odds of $P(Y = 1 \mid X)$ versus $P(Y = 0 \mid X)$, which represents a linear function of the predictors. A natural way to extend this to allow for non-linear relationships is to use the model

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p).$$

This is a logistic regression GAM. It has all the same pros and cons as for quantitative responses.

8 Tree-Based Methods

In this chapter, we describe tree-based methods for regression and classification. These involve stratifying or segmenting the predictor space into a number of simple regions. In order to make a prediction for a given observation, we typically use the mean or the mode of the training observations in the region to which it belongs. Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as decision tree methods.

8.1 The Basis of Decision Trees

Decision trees can be applied to both regression and classification problems.

8.1.1 Regression Trees

Prediction via Stratification of the Feature Space

In the process of building a regression tree, there are two steps.

1. We divide the predictor space, that is, the set of possible values for $X_1, X_2, \ldots, X_p$, into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.

2. For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observation in R_j.

For instance, suppose that in Step 1 we obtain two regions, R_1 and R_2, and that the response mean of the training observations in the first region is 10, while the response mean of the training observations in the second region is 20. Then for a given observation $X = x$, if $x \in R_1$ we will predict a value of 10, and if $x \in R_2$ we will predict a value of 20.

How do we construct the regions $R_1, \ldots, R_J$? In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or boxes, for simplicity and for ease of interpretation of the resulting predictive model. The goal is to find boxes $R_1, \ldots, R_J$ that minimize the RSS, given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box. It is computationally infeasible to consider every possible partition of the feature space into J boxes. For this reason, we take a top-down, greedy approach that is known as recursive binary splitting. The approach is top-down because it begins at the top of the tree and then successively splits the predictor space; each split is indicated via two new branches further down on the tree. It is greedy because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

In order to perform recursive binary splitting, we first select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X \mid X_j < s\}$ and $\{X \mid X_j \geq s\}$ leads to the greatest possible reduction in RSS. That is, we consider all predictors $X_1, \ldots, X_p$, and all possible values of the cutpoint s for each of the predictors, and then choose the predictor and cutpoint such that the resulting tree has the lowest RSS. For any j and s, we define the pair of half-planes

$$R_1(j, s) = \{X \mid X_j < s\} \text{ and } R_2(j, s) = \{X \mid X_j \geq s\},$$

and we seek the value of j and s that minimize the equation

$$\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2,$$

where $\hat{y}_{R_1}$ is the mean response for the training observations in $R_1(j, s)$, and $\hat{y}_{R_2}$ is the mean response for the training observations in $R_2(j, s)$.

Next, we repeat the process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions. However, this time, instead of splitting the entire predictor space, we split one of the two previously identified regions. We now have three regions. Again, we look to split one of these three regions further, so as to minimize the RSS. The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations.

Once the regions $R_1, \ldots, R_J$ have been created, we predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs.

Tree Pruning

The process described above may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performance. This is because the resulting tree might be too complex. A smaller tree with fewer splits might lead to lower variance and better interpretation

at the cost of a little bias. One possible alternative to the process is to build the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold. A better strategy is to grow a very large tree T_0, and then prune it back in order to obtain a subtree. Our goal is to select a subtree that leads to the lowest test error rate. Given a subtree, we can estimate its test error using cross-validation or the validation set approach. Estimating the cross-validation error for every possible subtree would be too cumbersome, since there is an extremely large number of possible subtrees.

Cost complexity pruning, also known as weakest link pruning, gives us a way to do just this. Rather than considering every possible subtree, we consider a sequence of trees indexed by a nonnegative tuning parameter α.

Algorithm: Building a Regression Tree

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the largest tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

For each value of α there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

is as small as possible. Here $|T|$ indicates the number of terminal nodes of the tree T, R_m is the rectangle corresponding to the mth terminal node, and $\hat{y}_{R_m}$ is the predicted response associated

with R_m, that is, the mean of the training observations in R_m. The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data. When $\alpha = 0$, then the subtree T will simply equal T_0. However, as α increases, there is a price to pay for having a tree with many terminal nodes, and so the above equation will tend to be minimized for a smaller subtree.

As we increase α, branches get pruned from the tree in a nested and predictable fashion, so obtaining the whole sequence of subtrees as a function of α is easy. We can select a value of α using validation set or using cross-validation. We then return to the full data set and obtain the subtree corresponding to α.

8.1.2 Classification Trees

A classification tree is very similar to a regression tree, except that it is used to predict a qualitative response rather than a quantitative one. For a regression tree, the predicted response for an observation is given by the mean response of the training observations that belong to the same terminal node. In contrast, for a classification tree, we predict that each observation belongs to the most commonly occuring class of training observations in the region to which it belongs.

The task of growing a classification tree is quite similar to the task of growing a regression tree. We use recursive binary splitting to grow a classification tree. However, in the classification setting, RSS cannot be used as a criterion for making the binary splits. A natural alternative to RSS is the classification error rate. It is simply the fraction of the training observations in that region that do not belong to the most common class:

$$E = 1 - \max_k(\hat{p}_{mk}).$$

Here $\hat{p}_{mk}$ represents the proportion of training observations in the mth region that are from the kth class. It turns out that classification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable.

The Gini index is defined by

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

a measure of total variance across the K classes. The Gini index takes on a small value if all of the $\hat{p}_{mk}$'s are close to zero or one. For this reason the Gini index is referred to as a measure of node purity, a small value indicates that a node contains predominantly observations from a single class.

An alternative to the Gini index is entropy, given by

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

Since $0 \leq \hat{p}_{mk} \leq 1$, it follows that $0 \leq -\hat{p}_{mk} \log \hat{p}_{mk}$. The entropy will take on a value near zero if the $\hat{p}_{mk}$'s are all near zero or one. Like Gini index, the entropy will take on a small value if the mth node is pure.

8.1.3 Trees Versus Linear Models

Linear regression assumes a model of the form

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j,$$

whereas regression trees assume a model of the form

$$f(X) = \sum_{m=1}^{M} c_m \cdot 1_{(X \in R_m)}$$

where $R_1, \ldots, R_M$ represent a partition of the feature space. Which model is better depends on the problem at hand. If the relationship between the features and the response is well approximated by a linear model, then an approach such as linear regression will likely work well, and will outperform a method such as a regression tree that does not exploit this linear structure. If instead there is a highly non-linear and complex relationship between the features and the response, then decision trees may outperform classical approaches. Sometimes, prediction using a tree may be preferred for the sake of interpretability and visualization.

8.1.4 Advantages and Disadvantages of Trees

Decision trees for regression and classification have a number of advantages over the more classical approaches.

- Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression.

- Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.
- Trees can be displayed graphically, and are easily interpreted even by a non-expert.
- Trees can easily handle qualitative predictors without the need to create dummy variables.
- Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches.
- Additionally, trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree.

8.2 Bagging, Random Forests, Boosting

Bagging, random forests, and boosting use trees as building blocks to construct more powerful prediction models.

8.2.1 Bagging

The bootstrap is an extremely powerful idea. It is used in many situations in which it is hard or even impossible to directly compute the standard deviation of a quantity of interest. The bootstrap can be used in a completely different context, in order to improve statistical learning methods such as decision trees.

Decision trees suffer from high variance. This means that if we split the training data into two parts at random, and fit a decision tree to both halves, the results that we get could be quite different. In contrast, a procedure with low variance will yield similar results if applied repeatedly to distinct data sets. Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method. It is particularly useful and frequently used in the context of decision trees.

Given a set of n independent observations $Z_1, \ldots, Z_n$, each with variance σ^2, the variance of the mean $\overline{Z}$ of the observations is given by σ^2/n. In other words, averaging a set of observations reduces variance. Hence a natural way to reduce the variance and hence increase prediction accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions. In other words, we could calculate $\hat{f}^1(x), \hat{f}^2(x), \ldots, \hat{f}^B(x)$ using B separate training sets, and average them in order to

obtain a single low-variance statistical learning model, given by

$$\hat{f}_{\text{avg}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{b}(x).$$

We can bootstrap, by taking repeated samples from the single training data set. In this approach we generate B different bootstrapped training data sets. We then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$, and finally average all the predictions, to obtain

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

This is called bagging.

While bagging can improve predictions for many regression methods, it is particularly useful for decision trees. To apply bagging to regression trees, we simply construct B regression trees using B bootstrapped training sets, and average the resulting predictions. These trees are grown deep, and are not pruned. Hence each individual tree has high variance, but low bias. Averaging these B trees reduces the variance.

How can bagging be extended to a classification problem where Y is qualitative? The simplest approach is as follows. For a given test observation, we can record the class predicted by each of the B trees, and take a majority vote.

Out-of-Bag Error Estimation

There is a very straightforward way to estimate the test error of a bagged model, without the need to perform cross-validation or the validation set approach. The key to bagging is that trees are repeatedly fit to bootstrapped subsets of the observations. On average, each bagged tree makes use of around two-thirds of the observations. The remaining one-third of the observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations. We can predict the response for the ith observation using each of the trees in which that observation was OOB. This will yield around $B/3$ predictions for the ith observation. In order to obtain a single prediction for the ith observation, we can average these predicted responses or can take a majority vote. This leads to a single OOB prediction for the ith observation. An OOB prediction can be obtained in this way for each of the n observations, from which the overall OOB MSE or classification error can be computed. The resulting OOB error is a valid estimate of the test error for the bagged model, since the response for each observation is predicted using only the trees that were not fit using that observation. With

B sufficiently large, OOB error is virtually equivalent to leave-one-out cross-validation error.

Variable Importance Measures

Bagging typically results in improved accuracy over prediction using a single tree. It can be difficult to interpret the resulting model. Although the collection of bagged trees is much more difficult to interpret than a single tree, one can obtain an overall summary of the importance of each predictor using RSS or the Gini index. In the case of bagging regression trees, we can record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees. A large value indicates an important predictor.

8.2.2 Random Forests

Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. As in bagging, we build a number of decision trees on bootstrapped training samples. But when building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors. A fresh sample of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$, that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors.

In building a random forest, at each split in the tree, the algorithm is not even allowed to consider a majority of the available predictors. This may sound crazy, but it has a clever rationale. Suppose there is one very strong predictor in the data set, along with a number of other moderately strong predictors. Then in the collection of bagged trees, most or all of the trees will use this strong predictor in the top split. Consequently, all of the bagged trees will look quite similar to each other. Hence the predictions from the bagged trees will by highly correlated. Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities. This means that bagging will not lead to a substantial reduction in variance over a single tree in this setting.

Random forests overcome this problem by forcing each split to consider only a subset of the predictors. Therefore, on average $(p - m)/p$ of the splits will not even consider the strong predictor, and so other predictors will have more of a chance. We can think of this process as decorrelating the trees, thereby making the average of the resulting trees less variable and hence more reliable.

The main difference between bagging and random forests is the choice of predictor subset size m. For instance, if a random forest is built using $m = p$, then this amounts simply to bagging.

Using a small value of m in building a random forest will typically be helpful when we have a

large number of correlated predictors.

8.2.3 Boosting

Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification.

Bagging involves creating multiple copies of the original training data set using the bootstrap, ftting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model. Each tree is built on a bootstrap data set, independent of the other trees. Boosting works in a similar way, except that the trees are grown sequentially: each tree is grown using information from previously grown trees. Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set.

Consider first the regression setting. Like bagging, boosting involves combining a large number of decision trees, $\hat{f}^1, \ldots, \hat{f}^B$.

Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly. Given the current model, we fit a decision tree to the residuals from the model. That is, we fit a tree using the current residuals, rather than the outcome Y, as the response. We then add this new decision tree into the fitted function in order to update the residuals. Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm. By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well. The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.

1. The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.

2. The shrinkage parameter λ. a small positive number. This controls ther rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.

3. The number d of splits in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a stump, consisting of a single split. In this case, the boosted ensemble is fitting an additive model, since each term involves only a single variable. More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

Algorithm: Boosting for Regression Trees

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.

2. For $b = 1, 2, \ldots, B$, repeat:

   (a) Fit a tree $\hat{f}^b$ with d splits to the training data (X, r).

   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$

   (c) Update the residuals

   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

9 Support Vector Machines

In this chapter, we discuss the support vector machine (SVM), an approach for classification. 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 SVM is a generalization of a simple and intuitive classifier called the maximal margin classifer. 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. SVMs are intended for the binary classification setting in which there are two classes.

9.1 Maximal Margin Classifier

9.1.1 What Is a Hyperplane?

In a p-dimensional space, a hyperplane is a flat affine subspace of dimension $p-1$. 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.

In two dimensions, a hyperplane is defined by the equation

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

for parameters β_0, β_1, and β_3. This can easily be extended to the p-dimensional setting:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

defines a p-dimensional hyperplane, in the sense that if a point $X = (X_1, X_2, \ldots, X_p)^T$ in p-dimensional space satisfies the above equation, then X lies on the hyperplane.

Now, suppose that X does not satisfy this; rather,

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots \beta_p X_p > 0.$$

Then this tells us that X lies to one side of the hyperplane. On the other hand if

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0,$$

then X lies on the other side of the hyperplane. We can think of the hyperplane as dividing

p-dimensional space into two halves.

9.1.2 Classification Using a Separating Hyperplane

Suppose that we have a $n \times p$ data matrix X that consists of n training data observations in p-dimensional space,

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix},$$

and that these observations fall into two classes, that is, $y_1, \ldots, y_n \in \{-1, 1\}$ where -1 represents one class and 1 the other class. We also have a test observation, a p-vector of observed features $x^* = (x_1^* \quad \ldots \quad x_p^*)^T$. Our goal is to develop a classifier based on the training data that will correctly classify the test observation using its feature measurements.

Suppose that it is possible to construct a hyperplane that separates the training observations perfectly according to their class labels. Then a separating hyperplane has the property that

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} > 0 \text{ if } y_i = 1,$$

and

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_p x_{ip} < 0 \text{ if } y_i = -1.$$

Equivalently, a separating hyperplane has the property that

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 0$$

for all $i = 1, \ldots, n$.

If a separating hyperplane exists, we can use it to construct a very natural classifier: a test observation is assigned to a class depending on which side of the hyperplane it is located. 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^*.

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. 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.

If $\beta_0, \beta_1, \ldots, \beta_p$ are the coefficients of the maximal margin hyperplane, then the maximal margin classifier classifies the test observation x^* based on the sign of $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$.

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 $x_1, \ldots, x_n \in \mathbb{R}^p$ and associated class labels $y_1, \ldots, y_n \in \{-1, 1\}$. The maximal margin hyperplane is the solution to the optimization problem:

1. Maximize $\beta_0, \beta_1, \ldots, \beta_p, M$

2. subject to $\sum_{j=1}^{p} \beta_j^2 = 1$,

3. $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \ldots, n.$

This optimization problem is simpler than it looks. The constraint in Step 3 that

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \ldots, n$$

guarantees that each observation will be on the correct side of the hyperplane, provided that M is positive. The constraints in Step 2 and 3 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 $\beta_0, \beta_1, \ldots, \beta_p$ to maximize M. This is exactly the definition of the maximal margin hyperplane.

9.1.5 The Non-separable Case

In many cases no separating hyperplane exists, and so there is no maximal margin classifier. The generaliztion of the maximal margin classifier to the non-separable case is known as the support vector classifier.

9.2 Support Vector Classifiers

9.2.1 Overview of the Support Vector Classifier

A classifier based on a separating hyperplane will necessarily perfectly classify all of the training observations; this can lead to sensitivity to individual observations.

We might be willing to consider a classifier based on a hyperplane that does not perfectly separate the two classes, in the interest of

- Greater robustness to individual observations, and
- Better classification of most of the training observations.

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.

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 separate most of the training observations into the two classes, but may misclassify a few observations. It is the solution to the optimization problem

1. Maximize M
2. subject to $\sum_{j=1}^{p} \beta_j^2 = 1$,
3. $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$,
4. $\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C$,

where C is a nonnegative tuning parameter. As before, M is the width of the margin; we seek to make this quantity as large as possible. In Step 3, $\epsilon_1, \ldots, \epsilon_n$ are slack variables that allow individual observations to be on the wrong side of the margin or the hyperplane. Once we have solved Step 1 through 4, we classify a test observation x^* as before, by simply determining on which side of the hyperplane it lies.

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 observation is on the correct side of the margin. 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.

The optimization problem hase 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.

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, that does this in an automatic way.

9.3.1 Classification with Non-linear Decision Boundaries

In practice we are sometimes faced with non-linear class boundaries. In the case of the support vector classifier, we could address the problem of possibly non-linear boundaries between classes 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

$$X_1, X_2, \ldots, X_p,$$

we could instead fit a support vector classifier using $2p$ features

$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2.$$

Then the optimization steps would become

1. maximize M
2. subject to $y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2\right) \geq M(1 - \epsilon_i)$,

3. $\sum_{i=1}^{n} \epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1.$

9.3.2 The Support Vector Machine

The SVM is an extension of the support vector classifier that results from enlarging the feature space in a specific way, using kernels.

It turns out that the solution to the support vector classifier problem involves only the inner products of the observations. 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

$$\langle x_i, x_i' \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}.$$

It can be shown that

- The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle,$$

  where there are n parameters $\alpha_i, i = 1, \ldots, n$, one per training observation.

- To estimate the parameters $\alpha_1, \ldots, \alpha_n$ 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.

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. 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 of $f(x)$ as

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle,$$

which typically involves far fewer terms.

The generalization of the inner product is the form

$$K(x_i, x_{i'}),$$

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

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j},$$

which would just give us back the support vector classifier. The above equation 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 correlation.

9.4 SVMs with More than Two Classes

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 $\binom{K}{2}$ SVMs, each of which compares a pair of classes.

9.4.2 One-Versus-All Classification

We fit K SVMs, each time comparing one of the K classes to the remaining $K - 1$ classes. Let $\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$ denote the parameters that result from fitting an SVM comparing the k th class to the others. Let x^* denote a test observation. We assign the observation to the class for which $\beta_{0k} + \beta_{1k} x_1^* + \beta_{2k} x_2^* + \cdots + \beta_{pk} x_p^*$ is largest.

References

[1] Gareth James et al. *An Introduction to Statistical Learning with Applications in R*. 2nd. ISBN 978-1-0716-1417-4. New York: Springer, 2021. URL: https://www.statlearning.com.