Book Notes

Statistical Learning with R

Jon Geiger

January 25, 2022

Tentative Schedule

Week 1: Chapter 1

Week 2: Chapter 2.1-2.2

Week 3: Chapter 4.1-4.3 (4.4 optional). Lab 4.7.1-2 (4.7.3-5 optional).

Week 4: Chapter 4.5-4.6. Lab 4.7.7.

Week 5: Chapter 5.1-5.2. Lab 5.3.1-4.

Week 6: Chapter 6.2-6.3 (6.4 optional). Lab 6.5.1-3.

Week 7: Chapter 8.1-8.2. Lab 8.3.1-5

Week 8: Chapter 12.1-12.2. Lab 12.5.1.

Week 9: Chapter 12.4. Lab 12.5.3-4

Week 10: Grace Week

Organization

Chapters get single-# headers, and sections get double-## headers.

Sub-sections (e.g. Chapter 2.1.3) are separated with horizontal lines (created with three asterisks):

Chapter 1: Introduction

Statistical Learning is split up into **supervised learning** and **unsupervised learning**.

- *Supervised learning* "involves building a statistical modeling for predicting, or estimating, an output based on one or more inputs."
- *Unsupervised learning* involves "inputs but no supervising output," allowing us to learn structure from the data.

Some Key Datasets

`Wage` Data:

- Includes a number of factors relating to wages for a specific group of men from the Atlantic region of the U.S.
- Involves predicting a *quantitative* output, useful in *regression*

`Smarket` (Stock Market) Data:

- Contains daily movements in the S&P 500 in the five-year period between 2001 and 2005
- The goal is *classification*, or to predict whether the prices will increase or decrease on a given day.

`NCI60` Gene Expression Data:

- Contains 6,830 gene expression measurements for each of 64 cancer cell lines.
- This is a *clustering* problem, and we can analyze *principal components* of the data.

Purpose of the Book

"The purpose of *An Introduction to Statistical Learning* (ISL) is to facilitate the transition of statistical learning from an academic to a mainstream field."

ISL is based on four principles:

1. Many statistical learning methods are relevant and useful in a wide range of academic and non-academic disciplines, beyond just the statistical sciences.
2. Statistical learning should not be viewed as a series of black boxes.
3. While it is important to know what job is performed by each cog, it is not necessary to have the skills to construct the machine inside the box.
4. We presume that the reader is interested in applying statistical learning methods to real-world problems.

Notation

- $\mathbf{X}$ is an $n \times p$ matrix whose (i, j)th element is x_{ij}
  - Rather than $m \times n$ for m observations of n variables, we can think of $\mathbf{X}$ as a matrix (or spreadsheet) with n rows and p columns, or n observations of p variables.
- Vectors are column vectors by default.
- x_i is the vector of **rows** of $\mathbf{X}$, with length p:

$$x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix}$$

- $\mathbf{x}$ is the vector of the **columns** of $\mathbf{X}$, with length n:

$$\mathbf{x}_j = \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{nj} \end{pmatrix}$$

- y_i denotes the ith observation of the variable on which we wish to make predictions. We can write the set of all n observations in vector form as:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

- A vector of length n will be denoted in lower-case bold, such as:

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

Chapter 2: Statistical Learning

2.1: What is Statistical Learning?

Input Variables are also called *predictors*, *independent variables*, *features*, or just *variables*, denoted by X_i's. **Output Variables** are also called *responses* or *dependent variables*, denoted using the symbol Y.

With a quantitative response Y and p predictors $X = (X_1, X_2, \ldots, X_p)$, we assume there is some relationship between Y and X which takes the general form:

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

f is an unknown function of all the X_i's, and ϵ is a random *error term*, independent of X with a zero mean. f represents the *systematic* information that X provides about Y.

Statistical Learning refers to a set of approaches for estimating f.

We estimate f for two reasons: **prediction**, and **inference**.

We can *predict* Y with $\hat{Y} = \hat{f}(X)$, where $\hat{f}(X)$ represents our estimate for f, and $\hat{Y}$ is the resulting prediction of Y. Our goal is to minimize the reducible error (the error which can be changed by modifying f), keeping in mind that ϵ cannot be reduced.

In *inference*, we wish to understand the association between Y and each of the X_i's. A problem of inference could be driven by the following questions:

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

There are two methods to estimating f: **Parametric Methods**, and **Non-Parametric Methods**.

Parametric Methods involve a two-step, model-based approach:

1. We make an assumption about the functional form of f. One simple example might be that f is linear in X:
   $$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$
   With a linear model, we estimate $p + 1$ coefficients from β_0 to β_p.
2. After model selection, we need to *fit* or *train* the model. This is the process of estimating the parameters β_0 through β_p such that in the case of the linear model above,
   $$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

The process is called *parametric* because we are *estimating parameters* to fit the model. This is generally simpler because it is easier to fit a set of parameters than it is to fit an entirely arbitrary function f. A parametric model has the disadvantage in that it typically does not match the true unknown form of f.

Non-parametric models have the advantage of not assuming a form for f, and can thus fit a wider range of possible shapes of f. These models need many more observations to work effectively. An example of a non-parametric model is a *thin-plate spline*, which can be either rough or smooth, where the roughness is analogous to overfitting data in a parametric model.

In choosing a model, there's a tradeoff between the *interpretability* of that model and the *flexibility* of that model. Generally, the more flexible a model is (bagging, boosting, SVMs, Deep Learning), the less interpretable the output is. Likewise, less flexible models (Subset Selection, Lasso, Least Squares) tend to be more interpretable.

- Choosing a more *restrictive* (less flexible) model is preferable in problems of inference, as the outcome needs to be interpretable.
- Conversely, more flexible models can be better for prediction problems when interpretability is not strictly necessary.

All examples from this past chapter have been examples of *supervised learning*. These techniques include linear regression, logistic regression, Generalized Additive Models (GAMs), boosting, Support Vector Machines (SVMs), and boosting. These all have in common that for a single observation x_i, there is an associated response measurement y_i.

Unsupervised learning tackles a different challenge, namely, that for each observation i, we have a set of measurements x_i with no associated response y_i. These types of problems are usually *cluster analysis* problems, where we try to find patterns, order, and/or groups in the data. More specifically, the goal is to figure whether or not the data fall into distinct groups, and if they do, which observations comprise those groups.

There do exist some problems which can be categorized as *semi-supervised learning* problems, namely scenarios in which our response data is incomplete. Namely, if we have both predictor and response measurements for m observations, where $m < n$, then for the remaining $m - n$ measurements we have predictor data but no response data.

Regression problems have a *quantitative response*. e.g. Least Squares.

Classification problems have a *categorical response*. e.g. Despite its name, logistic regression.

Some statistical methods, such as K-nearest neighbors (KNN) and boosting, can be used for either quantitative or categorical responses.

2.2: Assessing Model Accuracy

In evaluating statistical learning methods, we need some measure of how well its predictions match the observed data. In the case of regression, this is the *mean squared error* (MSE):

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

A "good prediction" will yield a small MSE, as the error term $(y_i - \hat{f}(x_i))$ will be small for each observation.

With the case of linear regression, though, it is quite possible to overfit the data to perfectly capture each response (thus the MSE would be zero). We don't care how our model performs on our *training data*, but rather how it performs when we feed it *test data*. Because of this, we care about fitting parameters for $\hat{f}$ that will minimize the *test MSE* rather than the *training MSE*.

Often when choosing a model, it is useful to plot the training MSE and test MSE as a function of flexibility. Typically, the most flexible models will can yield a very low training MSE but will fail to produce a low test MSE (overfitting). Similarly, the least flexible models cannot produce a low MSE in either the training nor the test data, so the best models have a training MSE and test MSE that are roughly equal.

The test MSE for a given value x_0 can always be decomposed into the sum of three fundamental quantities:

1. The *variance* of $\hat{f}(x_0)$
2. The squared *bias* of $\hat{f}(x_0)$
3. The variance of the error terms ϵ

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

In this case, $E\left(y_0 - \hat{f}(x_0)\right)^2$ is the *expected test MSE* at x_0. If our goal is to minimize the MSE, we need to estimate f such that the learning method achieves both a *low bias* and a *low variance*. This is known as the **bias-variance trade-off**.

The **variance** of a statistical learning method refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set. Ideally, the estimate for f should not vary much between training sets. This would be a *low variance*. In general, more flexible statistical methods have higher variance (overfitting to the training data).

The **bias** of a statistical learning method refers to the error that is introduced by approximating a real-life problem (usually very complicated) by a much simpler model. It's unlikely that a real-life problem actually has a linear relationship (as an example), so performing linear regression will result in some bias in estimating f.

It's very easy to achieve a model with low bias and high variance (draw a line that goes through every data point), as well as a model with high bias and low variance (fit a horizontal line through the data). Minimizing the test MSE is the key to finding the perfect bias-variance trade-off.

The bias-variance tradeoff also applies to the classification setting. In the case of classification, we assess our estimate of f with the training **error rate**, or the proportion of mistakes that are made if we apply our estimate to the training observations:

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i) \quad \text{where} \quad I = \begin{cases} 1 & \text{if } y_i \neq \hat{y}_i \\ 0 & \text{if } y_i = \hat{y}_i \end{cases}$$

In other words, I makes a binary indication of whether or not a classification error was made. Applying a classification method to test data, then, the *test error rate* with a set of test observations (x_0, y_0) is given by

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

A good classifier is one for which the test error is minimized.

The **Bayes Classifier** is one which utilizes conditional probability to classify points in a two-class problem. Particularly, it seeks to find $Pr(Y = j|X = x_0)$, or "given that the observation is x_0, what's the probability that it belongs to class j?"

- The *Bayes error rate* is given by $1 - \max_j Pr(Y = j|X = x_0)$
- In general, the overall Bayes error rate is given by $1 - E\left(\max_j \Pr(Y = j|X = x_0)\right)$

The K**-Nearest Neighbors Classifier** (KNN) allows us to construct a decision boundary based on real data. It accomplishes this by choosing a test observation x_0, identifying the K closest points ($\mathcal{N}_0$), then estimates the conditional probability that x_0 would be in class j as the fraction of points whose response values equal j:

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

Similar to regression methods from earlier sections, KNN can be very flexible with low values of K, and for any set of training data, it can be important to optimize the value of K selected to minimize the test error.

Chapter 4: Classification

Classification is a statistical method used with a *qualitative* response variable.

Some classifiers covered in this chapter include logistic regression, linear discriminant analysis, quadratic discriminant analysis, naive Bayes, and K-nearest neighbors. These topics are used to segue into Generalized Linear Models and Poisson Regression.

4.1: An Overview of Classification

The `Default` data set will be used extensively, in predicting whether an individual will default on their credit card payment, on the basis of their annual income and monthly credit card balance.

We'll be building a model to predict `default` from `balance` (X_1) and `income` (X_2).

4.2: Why Not Linear Regression?

Linear regression doesn't work for classification problems because in order to predict an outcome, we need to *numerically encode* the categories as a quantitative response variable. This doesn't work because the categories rarely ever have any logical order.

In the case of predicting medical diagnoses, a response variable might look like:

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

The situation improves slightly if we choose to use the *dummy variable* approach, where we code a response variable which looks like:

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

In this case, we could have some estimates outside of the $[0, 1]$ range, which leads to difficulty interpreting probabilities.

Overall: (1) regression cannot accommodate non-binary classification, and (b) regression methods will not provide meaningful estimates of $\Pr(Y|X)$, even with just two classes.

4.3: Logistic Regression

Consider the binary classification problem, namely with the `Default` data set.

Logistic regression models the *probability* that Y belongs to a particular class, rather than modeling the classification directly.

The probability of defaulting given a certain balance can be expressed as

$$\Pr(\texttt{balance} = \texttt{Yes}|\texttt{balance}) \equiv p(\texttt{balance})$$

In general, we notate that $p(X) = \Pr(Y = 1|X)$. In logistic regression, we use the logistic function, which we can rearrange to create a linear regression problem.

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \quad \text{Logistic Function}$$

$$\therefore \quad \frac{p(X)}{1 - p(X)} = e^{\beta_0+\beta_1 X} \quad \text{Odds}$$

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X \quad \text{Log Odds / Logit}$$

Odds close to zero indicate low probabilities of default, and values close to ∞ indicate high probabilities of default. Interpreting this final equation is a bit tricky, as a one-unit increase in X will yield a β_1 increase in the log odds.

The coefficients β_0 and β_1 must be estimated to best fit our training data. This is done using *maximum likelihood estimation*. The likelihood function is:

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

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

In a problem of inference looking for association between `default` and `balance`, our null hypothesis would be $H_0 : \beta_1 = 0$, which makes sense because a one-unit increase in X should not affect Y at all. So the null logistic function will be $p(X) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$. Similarly to linear regression, the z-statistic associated with β_1 is $z = \hat{\beta}_1/\text{SE}(\hat{\beta}_1)$.

For making predictions, we plug our estimates into the logistic function, so we have the equation:

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0+\hat{\beta}_1 X}}$$

Multiple Logistic Regression extends nicely out from simple logistic regression, where we can generalize our log odds equation from earlier:

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

Where we have p predictors. This also means that our logistic equation for making predictions becomes:

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

We would also use maximum likelihood estimation to estimate all the coefficients $\beta_0, \beta_1, \ldots, \beta_p$.

What happens when we want to classify something which has more than one outcome? In the previous section we dealt with the three classes being `stroke`, `drug overdose`, and `epileptic seizure`. This is a problem of **multinomial logistic regression**. We select a single class to act as the *baseline*, then we say that:

$$\Pr(Y = k|X = x) = \begin{cases} \dfrac{e^{\beta_{k0}+\beta_{k1}x_1+\cdots+\beta_{kp}x_p}}{1+\sum_{l=1}^{K-1} e^{\beta_{l0}+\beta_{l1}x_1+\cdots+\beta_{lp}x_p}} & \text{for } k = 1, \ldots, K-1, \\ \dfrac{1}{1+\sum_{l=1}^{K-1} e^{\beta_{l0}+\beta_{l1}x_1+\cdots+\beta_{lp}x_p}} & \text{for } k = K. \end{cases}$$

Additionally, it can be shown that

$$\log\left(\frac{\Pr(Y = k|X = x)}{\Pr(Y = K|X = x)}\right) = \beta_{k0} + \beta_{k1}x_1 + \cdots + \beta_{kp}x_p$$

An alternative coding for multinomial logistic regression is known as *softmax* coding, where, rather than selecting a baseline class, we treat all K classes symmetrically, and assume that for $k = 1, \ldots, K$,

$$\Pr(Y = k|X = x) = \frac{e^{\beta_{k0}+\beta_{k1}x_1+\cdots+\beta_{kp}x_p}}{1+\sum_{l=1}^{K} e^{\beta_{l0}+\beta_{l1}x_1+\cdots+\beta_{lp}x_p}}$$

So, we actually estimate coefficients for all K classes rather than just for $K-1$ classes. As a result of this, the log odds ratio between the kth and k'th classes is

$$\log\left(\frac{\Pr(Y = k|X = x)}{\Pr(Y = k'|X = x)}\right) = (\beta_{k0} - \beta_{k'0}) + (\beta_{k1} - \beta_{k'1})x_1 + \cdots + (\beta_{kp} - \beta_{k'p})x_p$$

4.4: Generative Models for Classification

An alternative model for classification makes use of *Bayes' Theorem*, which states that

$$\Pr(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

This is not how the textbook defines Bayes' Theorem, so we will derive the textbook definition from this representation to make sense of the book's definition.

When talking about multinomial classification, let us assume that we have K total classes, and we want to find the probability that an observation belongs to a class k given information about that observation X. We can express this with Bayes' Theorem as:

$$\Pr(Y = k|X = x) = \frac{\Pr(X = x|Y = k) \cdot \Pr(Y = k)}{\Pr(X = x)}$$

Cleaning this up a bit, we can rewrite some of the probabilities:

$$\Pr(k|x) = p_k(x) = \frac{\Pr(x|k) \cdot \Pr(k)}{\Pr(x)}$$

If we have classes $k = 1, 2, 3, \ldots, K$, we can use the law of total probability to expand this formula a bit in

order to figure out what $\Pr(x)$ is.

$$\begin{aligned} p_k(x) &= \frac{\Pr(x|k) \cdot \Pr(k)}{\Pr(x)} \\ &= \frac{\Pr(x|k) \cdot \Pr(k)}{\Pr(x \cap 1) + \Pr(x \cap 2) + \cdots + \Pr(x \cap K)} \\ &= \frac{\Pr(x|k) \cdot \Pr(k)}{\Pr(x|1) \cdot \Pr(1) + \Pr(x|2) \cdot \Pr(2) + \cdots + \Pr(x|K) \cdot \Pr(K)} \\ &= \frac{\Pr(x|k) \cdot \Pr(k)}{\sum_{l=1}^{K} \Pr(x|l) \cdot \Pr(l)} \\ &= \frac{\Pr(k) \cdot \Pr(x|k)}{\sum_{l=1}^{K} \Pr(l) \cdot \Pr(x|l)} \end{aligned}$$

If we let $f_k(x)$ be the *(probability) density function* of X for an observation which comes from class k, and we let π_k represent the probability that an observation comes from class k (also called the **prior probability**), then we have:

$$p_k(x) = \frac{\pi_k f_k(x)}{\sum_{\ell=1}^{K} \pi_\ell f_\ell(x)}$$

Rather than directly modeling $p_k(x)$ (also called the **posterior probability**) as we do with logistic regression, we can estimate the different prior probabilities $\pi_1, \ldots, \pi_K$, and the various density functions for the K classes $f_1(x), \ldots, f_K(x)$. The prior probabilities are relatively easy to estimate by taking a random sample, but estimating $f_k(x)$ proves to be a slightly bigger challenge.

We'll talk about three classifiers that all use different estimates of $f_k(x)$: *linear discriminant analysis*, *quadratic discriminant analysis*, and *naive Bayes*.

Let's assume the following:

- We have $p = 1$ predictors.
- $f_k(x)$ is *normal* or *Gaussian*. In order to approximate $f_k(x)$, we have to make some assumptions about its shape.
- The variances of all the classes are equal. That is, that there is a shared variance term among all the classes, σ^2.

Then,

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

and

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{\ell=1}^{K} \pi_\ell \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_\ell)^2\right)}$$

Our goal with classification is to choose the class k such that $p_k(x)$ is maximized. Maximizing $p_k(x)$ is equivalent to maximizing its logarithm, thus we want to assign x to a class k for which $\delta_k(x)$ is maximized, where $\delta_k(x)$ is given by:

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

In reality, we do not know the parameters of the Gaussian distribution from which we assume the observations come, we use *Linear Discriminant Analysis* to replace the parameters with statistics. We use the following estimations:

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

The LDA classifier then assigns an observation $X = x$ to the class for which the *discriminant function* $\hat{\delta}_k(x)$ is the largest:

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

Linear Disciminant Analysis is *linear* because the discriminant functions are linear functions of x.

When we have $p > 1$, we need to assume a multivariate Gaussian distribution with a class-specific mean vector and a common covariance matrix.

Without going through all the details, the Bayes classifier assigns an observation $X = x$ to the class for which $\delta_k(x)$ is the largest, where we have:

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

Here, x is an observation vector with p components, $\mathbf{\Sigma}$ is a common covariance matrix.

Quadratic Discriminant Analysis makes the same assumption as LDA about the Gaussian shape of f, but assumes that each class has its own covariance matrix. The discriminant function is then

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

Naive Bayes takes a different approach than LDA and QDA in estimating $f_k(x)$. Rather than worrying about the covariances between the different predictors, we can assume that they are independent of one another. This means, then, that for f_{kj} being the density function of the jth predictor among observations in the kth class,

$$f_k(x) = f_{k1}(x_1) \times f_{k2}(x_2) \times \cdots \times f_{kp}(x_p)$$

This is a big convenience for modeling, since we don't need to worry about including marginal or joint distributions of the predictors. This assumption doesn't always hold, but is extremely convenient much of the time.

Now, the posterior probability is given by:

$$\Pr(Y = k | X = x) = p_k(x) = \frac{\pi_k \times f_{k1}(x_1) \times f_{k2}(x_2) \times \cdots \times f_{kp}(x_p)}{\sum_{\ell=1}^{K} \pi_\ell \times f_{\ell 1}(x_1) \times f_{\ell 2}(x_2) \times \cdots \times f_{\ell p}(x_p)}$$

The question now becomes, how do we model or estimate f_{kj}? We have a few options.

- If X is quantitative, we could simpply assume that each X_j is normally distributed, such that $(X_j | Y = k) \sim N(\mu_{jk}, \sigma_{jk}^2)$. In other words, we would assume that within each class, the jth predictor is drawn from a univariate normal distribution with mean μ_{jk} and variance σ_{jk}^2.

- Another option for a quantitative X is to use a non-parametric method such as creating a histogram or using a kernel density estimator (essentially a smoothed histogram) as an estimate for $f_{kj}(x_j)$.
- For qualitative X_j, we can use a sample proportion according to each class to estimate f_{kj}. Let's say that $X_j \in \{1, 2, 3\}$, and we have 100 observations in the kth class. Suppose that the jth predictor takes on values of 1, 2, and 3 in 32, 55, and 13 of those observations, respectively. Then we have:

$$\hat{f}_{kj}(x_j) = \begin{cases} 0.32 & \text{if } x_j = 1 \\ 0.55 & \text{if } x_j = 2 \\ 0.13 & \text{if } x_j = 3 \end{cases}$$

To recap, with K classes and p predictors, we are estimating $K \times p$ density functions. In other words, for each class k, there will be p density estimates. Applying the observation X to each of these density estimates in a class k should yield a probability (posterior probability) that X belongs to that class. See Figure 4.10 and its caption for a good visual explanation.

With a low number of predictors, naive Bayes will not necessarily outperform LDA or QDA because the reduction in variance is not super important. In scenarios with many predictors or with few training examples, though, naive Bayes will outperform LDA or QDA because of the assumption of independence.

4.5: A Comparison of Classification Methods

Without writing out all the derivations, we can look at a mathematical comparison between the different classification settings. In a setting with K classes, we look to maximize

$$\log\left(\frac{\Pr(Y = k|X = x)}{\Pr(Y = K|X = x)}\right)$$

for $k = 1, \ldots, K$.

For the LDA setting, we end up with:

$$\begin{aligned} \log\left(\frac{\Pr(Y = k|X = x)}{\Pr(Y = K|X = x)}\right) &= \log\left(\frac{\pi_k f_k(x)}{\pi_K f_K(x)}\right) \\ &= \cdots \\ &= a_k + \sum_{j=1}^{p} b_{kj} x_j \end{aligned}$$

Where $a_k = \log\left(\frac{\pi_k}{\pi_K}\right) - \frac{1}{2}(\mu_k + \mu_K)^T \mathbf{\Sigma}^{-1}(\mu_k - \mu_K)$ and b_{kj} is the jth component of $\mathbf{\Sigma}^{-1}(\mu_k - \mu_K)$. So LDA assumes that the log odds of the posterior probabilities is linear in x, just like logistic regression.

In the QDA setting, we have:

$$\begin{aligned} \log\left(\frac{\Pr(Y = k|X = x)}{\Pr(Y = K|X = x)}\right) &= \log\left(\frac{\pi_k f_k(x)}{\pi_K f_K(x)}\right) \\ &= \cdots \\ &= a_k + \sum_{j=1}^{p} b_{kj} x_j + \sum_{j=1}^{p} \sum_{l=1}^{p} c_{kjl} x_j x_l \end{aligned}$$

where a_k, b_kj, and c_kjl are functions of π_k, π_K, μ_k, μ_K, $\mathbf{\Sigma}_k$, and $\mathbf{\Sigma}_K$

Finally, in the naive Bayes setting, we have:

$$\begin{aligned}\log\left(\frac{\Pr(Y=k|X=x)}{\Pr(Y=K|X=x)}\right) &= \log\left(\frac{\pi_k f_k(x)}{\pi_K f_K(x)}\right)\\ &= \cdots\\ &= a_k + \sum_{j=1}^{p} g_{kj}(x_j)\end{aligned}$$

where $a_k = \log\left(\frac{\pi_k}{\pi_K}\right)$ and $g_{kj}(x_j) = \log\left(\frac{f_{kj}(x_j)}{f_{Kj}(x_j)}\right)$. This takes the form of a *generalized additive model*, which is covered in chapter 7.

We can notice a few things about these results:

- LDA is a special case of QDA with $c_{kjl} = 0$ for all k, j, l.
- Any classifier with a linear decision boundary is a special case of naive Bayes with $g_{kj}(x_j) = b_{kj}x_j$. This also means that LDA is a specific case of naive Bayes.
  - This is not directly intuitive, as for LDA we assumed a shared within-class covariance matrix, and for naive Bayes we assumed independence across all features.
- Modeling f with naive Bayes using a one-dimensional Gaussian distribution gives us the LDA classifier where $\mathbf{\Sigma}$ is a diagonal matrix with the jth diagonal element is equal to σ_j^2.
- QDA and naive Bayes are completely separate from one another, but LDA is a special case of both. Because QDA has interaction terms, it has the potential to be more accurate in scenarios with high interactions between classes.

There is no one classifier that can do it all. In six scenarios, six different classifiers were tested on each scenario. In addition to Logistic Regression, LDA, QDA, and naive Bayes, K-nearest neighbors with $K = 1$ and with K chosen automatically from a cross-validation set were compared. When the decision boundary is highly non-linear and we have many observations, the nonparametric approaches such as KNN tend to work much better than parametric models.

4.6: Generalized Linear Models