22 Classification

22.1 Introduction

The problem of predicting a discrete random variable Y from another random variable X is called **classification**, **supervised learning**, **discrimination**, or **pattern recognition.**

Consider IID data $(X_1, Y_1), \ldots, (X_n, Y_n)$ where

$$X_i = (X_{i1}, \ldots, X_{id}) \in \mathcal{X} \subset \mathbb{R}^d$$

is a d-dimensional vector and Y_i takes values in some finite set $\mathcal{Y}$. A **classification rule** is a function $h : \mathcal{X} \to \mathcal{Y}$. When we observe a new X, we predict Y to be $h(X)$.

22.1 Example. Here is a an example with fake data. Figure 22.1 shows 100 data points. The covariate $X = (X_1, X_2)$ is 2-dimensional and the outcome $Y \in \mathcal{Y} = \{0, 1\}$. The Y values are indicated on the plot with the triangles representing $Y = 1$ and the squares representing $Y = 0$. Also shown is a linear classification rule represented by the solid line. This is a rule of the form

$$h(x) = \begin{cases} 1 & \text{if } a + b_1 x_1 + b_2 x_2 > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Everything above the line is classified as a 0 and everything below the line is classified as a 1. ■

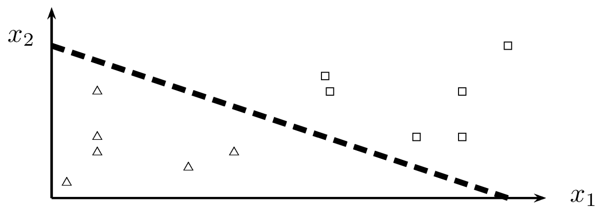

FIGURE 22.1. Two covariates and a linear decision boundary. △ means $Y = 1$. □ means $Y = 0$. These two groups are perfectly separated by the linear decision boundary; you probably won't see real data like this.

22.2 Example. Recall the the Coronary Risk-Factor Study (CORIS) data from Example 13.17. There are 462 males between the ages of 15 and 64 from three rural areas in South Africa. The outcome Y is the presence ($Y = 1$) or absence ($Y = 0$) of coronary heart disease and there are 9 covariates: systolic blood pressure, cumulative tobacco (kg), ldl (low density lipoprotein cholesterol), adiposity, famhist (family history of heart disease), typea (type-A behavior), obesity, alcohol (current alcohol consumption), and age. I computed a linear decision boundary using the LDA method based on two of the covariates, systolic blood pressure and tobacco consumption. The LDA method will be explained shortly. In this example, the groups are hard to tell apart. In fact, 141 of the 462 subjects are misclassified using this classification rule. ■

At this point, it is worth revisiting the Statistics/Data Mining dictionary:

Statistics	Computer Science	Meaning
classification	supervised learning	predicting a discrete Y from X
data	training sample	$(X_1, Y_1), \ldots, (X_n, Y_n)$
covariates	features	the X_i's
classifier	hypothesis	map $h : \mathcal{X} \to \mathcal{Y}$
estimation	learning	finding a good classifier

22.2 Error Rates and the Bayes Classifier

Our goal is to find a classification rule h that makes accurate predictions. We start with the following definitions:

22.3 Definition. *The* **true error rate**[1] *of a classifier h is*

$$L(h) = \mathbb{P}(\{h(X) \neq Y\}) \tag{22.1}$$

and the **empirical error rate** *or* **training error rate** *is*

$$\widehat{L}_n(h) = \frac{1}{n}\sum_{i=1}^{n} I(h(X_i) \neq Y_i). \tag{22.2}$$

First we consider the special case where $\mathcal{Y} = \{0, 1\}$. Let

$$r(x) = \mathbb{E}(Y|X = x) = \mathbb{P}(Y = 1|X = x)$$

denote the **regression function**. From Bayes' theorem we have that

$$\begin{aligned} r(x) &= \mathbb{P}(Y = 1|X = x) \\ &= \frac{f(x|Y=1)\mathbb{P}(Y=1)}{f(x|Y=1)\mathbb{P}(Y=1) + f(x|Y=0)\mathbb{P}(Y=0)} \\ &= \frac{\pi f_1(x)}{\pi f_1(x) + (1-\pi) f_0(x)} \end{aligned} \tag{22.3}$$

where

$$\begin{aligned} f_0(x) &= f(x|Y=0) \\ f_1(x) &= f(x|Y=1) \\ \pi &= \mathbb{P}(Y=1). \end{aligned}$$

22.4 Definition. *The* **Bayes classification rule** *h^* is*

$$h^*(x) = \begin{cases} 1 & \text{if } r(x) > \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases} \tag{22.4}$$

The set $\mathcal{D}(h) = \{x : \ \mathbb{P}(Y = 1|X = x) = \mathbb{P}(Y = 0|X = x)\}$ is called the **decision boundary.**

Warning! The Bayes rule has nothing to do with Bayesian inference. We could estimate the Bayes rule using either frequentist or Bayesian methods.

The Bayes rule may be written in several equivalent forms:

[1] One can use other loss functions. For simplicity we will use the error rate as our loss function.

$$h^*(x) = \begin{cases} 1 & \text{if } \mathbb{P}(Y=1|X=x) > \mathbb{P}(Y=0|X=x) \\ 0 & \text{otherwise} \end{cases} \tag{22.5}$$

and

$$h^*(x) = \begin{cases} 1 & \text{if } \pi f_1(x) > (1-\pi) f_0(x) \\ 0 & \text{otherwise.} \end{cases} \tag{22.6}$$

22.5 Theorem. *The Bayes rule is optimal, that is, if h is any other classification rule then $L(h^*) \le L(h)$.*

The Bayes rule depends on unknown quantities so we need to use the data to find some approximation to the Bayes rule. At the risk of oversimplifying, there are three main approaches:

1. Empirical Risk Minimization. Choose a set of classifiers $\mathcal{H}$ and find $\widehat{h} \in \mathcal{H}$ that minimizes some estimate of $L(h)$.
2. Regression. Find an estimate $\widehat{r}$ of the regression function r and define
$$\widehat{h}(x) = \begin{cases} 1 & \text{if } \widehat{r}(x) > \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$
3. Density Estimation. Estimate f_0 from the X_i's for which $Y_i = 0$, estimate f_1 from the X_i's for which $Y_i = 1$ and let $\widehat{\pi} = n^{-1} \sum_{i=1}^n Y_i$. Define
$$\widehat{r}(x) = \widehat{\mathbb{P}}(Y=1|X=x) = \frac{\widehat{\pi} \widehat{f}_1(x)}{\widehat{\pi} \widehat{f}_1(x) + (1-\widehat{\pi}) \widehat{f}_0(x)}$$
and
$$\widehat{h}(x) = \begin{cases} 1 & \text{if } \widehat{r}(x) > \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$

Now let us generalize to the case where Y takes on more than two values as follows.

22.6 Theorem. *Suppose that $Y \in \mathcal{Y} = \{1, \ldots, K\}$. The optimal rule is*

$$\begin{aligned} h(x) &= \operatorname{argmax}_k \mathbb{P}(Y=k|X=x) && (22.7) \\ &= \operatorname{argmax}_k \pi_k f_k(x) && (22.8) \end{aligned}$$

where

$$\mathbb{P}(Y=k|X=x) = \frac{f_k(x)\pi_k}{\sum_r f_r(x)\pi_r}, \tag{22.9}$$

$\pi_r = P(Y=r)$, $f_r(x) = f(x|Y=r)$ and argmax$_k$ *means "the value of k that maximizes that expression."*

22.3 Gaussian and Linear Classifiers

Perhaps the simplest approach to classification is to use the density estimation strategy and assume a parametric model for the densities. Suppose that $\mathcal{Y} = \{0, 1\}$ and that $f_0(x) = f(x|Y = 0)$ and $f_1(x) = f(x|Y = 1)$ are both multivariate Gaussians:

$$f_k(x) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left\{ -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) \right\}, \quad k = 0, 1.$$

Thus, $X|Y = 0 \sim N(\mu_0, \Sigma_0)$ and $X|Y = 1 \sim N(\mu_1, \Sigma_1)$.

22.7 Theorem. *If $X|Y = 0 \sim N(\mu_0, \Sigma_0)$ and $X|Y = 1 \sim N(\mu_1, \Sigma_1)$, then the Bayes rule is*

$$h^*(x) = \begin{cases} 1 & \text{if } r_1^2 < r_0^2 + 2\log\left(\frac{\pi_1}{\pi_0}\right) + \log\left(\frac{|\Sigma_0|}{|\Sigma_1|}\right) \\ 0 & \text{otherwise} \end{cases} \tag{22.10}$$

where

$$r_i^2 = (x - \mu_i)^T \Sigma_i^{-1}(x - \mu_i), \quad i = 1, 2 \tag{22.11}$$

is the **Manalahobis distance.** *An equivalent way of expressing the Bayes' rule is*

$$h^*(x) = \text{argmax}_k \delta_k(x)$$

where

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) + \log \pi_k \tag{22.12}$$

and $|A|$ denotes the determinant of a matrix A.

The decision boundary of the above classifier is quadratic so this procedure is called **quadratic discriminant analysis (QDA)**. In practice, we use sample estimates of $\pi, \mu_1, \mu_2, \Sigma_0, \Sigma_1$ in place of the true value, namely:

$$\begin{aligned}
\widehat{\pi}_0 &= \frac{1}{n}\sum_{i=1}^n (1 - Y_i), \quad \widehat{\pi}_1 = \frac{1}{n}\sum_{i=1}^n Y_i \\
\widehat{\mu}_0 &= \frac{1}{n_0}\sum_{i:\ Y_i=0} X_i, \quad \widehat{\mu}_1 = \frac{1}{n_1}\sum_{i:\ Y_i=1} X_i \\
S_0 &= \frac{1}{n_0}\sum_{i:\ Y_i=0} (X_i - \widehat{\mu}_0)(X_i - \widehat{\mu}_0)^T, \quad S_1 = \frac{1}{n_1}\sum_{i:\ Y_i=1} (X_i - \widehat{\mu}_1)(X_i - \widehat{\mu}_1)^T
\end{aligned}$$

where $n_0 = \sum_i (1 - Y_i)$ and $n_1 = \sum_i Y_i$.

A simplification occurs if we assume that $\Sigma_0 = \Sigma_0 = \Sigma$. In that case, the Bayes rule is

$$h^*(x) = \text{argmax}_k \delta_k(x) \tag{22.13}$$

where now

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} + \log \pi_k. \tag{22.14}$$

The parameters are estimated as before, except that the MLE of Σ is

$$S = \frac{n_0 S_0 + n_1 S_1}{n_0 + n_1}.$$

The classification rule is

$$h^*(x) = \begin{cases} 1 & \text{if } \delta_1(x) > \delta_0(x) \\ 0 & \text{otherwise} \end{cases} \tag{22.15}$$

where

$$\delta_j(x) = x^T S^{-1} \widehat{\mu}_j - \frac{1}{2} \widehat{\mu}_j^T S^{-1} \widehat{\mu}_j + \log \widehat{\pi}_j$$

is called the **discriminant function**. The decision boundary $\{x : \delta_0(x) = \delta_1(x)\}$ is linear so this method is called **linear discrimination analysis (LDA).**

22.8 Example. Let us return to the South African heart disease data. The decision rule in in Example 22.2 was obtained by linear discrimination. The outcome was

	classified as 0	classified as 1
$y = 0$	277	25
$y = 1$	116	44

The observed misclassification rate is $141/462 = .31$. Including all the covariates reduces the error rate to .27. The results from quadratic discrimination are

	classified as 0	classified as 1
$y = 0$	272	30
$y = 1$	113	47

which has about the same error rate $143/462 = .31$. Including all the covariates reduces the error rate to .26. In this example, there is little advantage to QDA over LDA. ■

Now we generalize to the case where Y takes on more than two values.

22.9 Theorem. *Suppose that $Y \in \{1, \ldots, K\}$. If $f_k(x) = f(x|Y = k)$ is Gaussian, the Bayes rule is*

$$h(x) = \text{argmax}_k \delta_k(x)$$

where

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log\pi_k. \tag{22.16}$$

If the variances of the Gaussians are equal, then

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1} + \log\pi_k. \tag{22.17}$$

We estimate $\delta_k(x)$ by by inserting estimates of μ_k, Σ_k and π_k. There is another version of linear discriminant analysis due to Fisher. The idea is to first reduce the dimension of covariates to one dimension by projecting the data onto a line. Algebraically, this means replacing the covariate $X = (X_1, \ldots, X_d)$ with a linear combination $U = w^T X = \sum_{j=1}^d w_j X_j$. The goal is to choose the vector $w = (w_1, \ldots, w_d)$ that "best separates the data." Then we perform classification with the one-dimensional covariate Z instead of X.

We need define what we mean by separation of the groups. We would like the two groups to have means that are far apart relative to their spread. Let μ_j denote the mean of X for Y_j and let Σ be the variance matrix of X. Then $\mathbb{E}(U|Y = j) = \mathbb{E}(w^T X|Y = j) = w^T\mu_j$ and $\mathbb{V}(U) = w^T\Sigma w$. [2] Define the separation by

$$\begin{aligned} J(w) &= \frac{(\mathbb{E}(U|Y=0) - \mathbb{E}(U|Y=1))^2}{w^T\Sigma w} \\ &= \frac{(w^T\mu_0 - w^T\mu_1)^2}{w^T\Sigma w} \\ &= \frac{w^T(\mu_0-\mu_1)(\mu_0-\mu_1)^T w}{w^T\Sigma w}. \end{aligned}$$

We estimate J as follows. Let $n_j = \sum_{i=1}^n I(Y_i = j)$ be the number of observations in group j, let $\overline{X}_j$ be the sample mean vector of the X's for group j, and let S_j be the sample covariance matrix in group j. Define

$$\widehat{J}(w) = \frac{w^T S_B w}{w^T S_W w} \tag{22.18}$$

[2]The quantity J arises in physics, where it is called the Rayleigh coefficient.

where

$$\begin{aligned} S_B &= (\overline{X}_0 - \overline{X}_1)(\overline{X}_0 - \overline{X}_1)^T \\ S_W &= \frac{(n_0 - 1)S_0 + (n_1 - 1)S_1}{(n_0 - 1) + (n_1 - 1)}. \end{aligned}$$

22.10 Theorem. *The vector*

$$w = S_W^{-1}(\overline{X}_0 - \overline{X}_1) \tag{22.19}$$

is a minimizer of $\widehat{J}(w)$. We call

$$U = w^T X = (\overline{X}_0 - \overline{X}_1)^T S_W^{-1} X \tag{22.20}$$

the **Fisher linear discriminant function**. *The midpoint m between $\overline{X}_0$ and $\overline{X}_1$ is*

$$m = \frac{1}{2}(\overline{X}_0 + \overline{X}_1) = \frac{1}{2}(\overline{X}_0 - \overline{X}_1)^T S_B^{-1}(\overline{X}_0 + \overline{X}_1) \tag{22.21}$$

Fisher's classification rule is

$$h(x) = \begin{cases} 0 & \text{if } w^T X \geq m \\ 1 & \text{if } w^T X < m. \end{cases}$$

Fisher's rule is the same as the Bayes linear classifier in equation (22.14) when $\widehat{\pi} = 1/2$.

22.4 Linear Regression and Logistic Regression

A more direct approach to classification is to estimate the regression function $r(x) = \mathbb{E}(Y|X = x)$ without bothering to estimate the densities f_k. For the rest of this section, we will only consider the case where $\mathcal{Y} = \{0, 1\}$. Thus, $r(x) = \mathbb{P}(Y = 1|X = x)$ and once we have an estimate $\widehat{r}$, we will use the classification rule

$$\widehat{h}(x) = \begin{cases} 1 & \text{if } \widehat{r}(x) > \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases} \tag{22.22}$$

The simplest regression model is the linear regression model

$$Y = r(x) + \epsilon = \beta_0 + \sum_{j=1}^{d} \beta_j X_j + \epsilon \tag{22.23}$$

where $\mathbb{E}(\epsilon) = 0$. This model can't be correct since it does not force $Y = 0$ or 1. Nonetheless, it can sometimes lead to a decent classifier.

Recall that the least squares estimate of $\beta = (\beta_0, \beta_1, \ldots, \beta_d)^T$ minimizes the residual sums of squares

$$\textsc{rss}(\beta) = \sum_{i=1}^{n} \left(Y_i - \beta_0 - \sum_{j=1}^{d} X_{ij}\beta_j \right)^2 .$$

Let $\mathbf{X}$ denote the $N \times (d+1)$ matrix of the form

$$\mathbf{X} = \begin{bmatrix} 1 & X_{11} & \ldots & X_{1d} \\ 1 & X_{21} & \ldots & X_{2d} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & X_{n1} & \ldots & X_{nd} \end{bmatrix} .$$

Also let $\mathbf{Y} = (Y_1, \ldots, Y_n)^T$. Then,

$$RSS(\beta) = (\mathbf{Y} - \mathbf{X}\beta)^T (\mathbf{Y} - \mathbf{X}\beta)$$

and the model can be written as

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon$$

where $\epsilon = (\epsilon_1, \ldots, \epsilon_n)^T$. From Theorem 13.13,

$$\widehat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y.$$

The predicted values are

$$\widehat{\mathbf{Y}} = \mathbf{X}\widehat{\beta}.$$

Now we use (22.22) to classify, where $\widehat{r}(x) = \widehat{\beta}_0 + \sum_j \widehat{\beta}_j x_j$.

An alternative is to use logistic regression which was also discussed in Chapter 13. The model is

$$r(x) = \mathbb{P}(Y = 1 | X = x) = \frac{e^{\beta_0 + \sum_j \beta_j x_j}}{1 + e^{\beta_0 + \sum_j \beta_j x_j}} \tag{22.24}$$

and the MLE $\widehat{\beta}$ is obtained numerically.

22.11 Example. Let us return to the heart disease data. The MLE is given in Example 13.17. The error rate, using this model for classification, is .27. The error rate from a linear regression is .26.

We can get a better classifier by fitting a richer model. For example, we could fit

$$\text{logit}\ \mathbb{P}(Y = 1 | X = x) = \beta_0 + \sum_j \beta_j x_j + \sum_{j,k} \beta_{jk} x_j x_k. \tag{22.25}$$

More generally, we could add terms of up to order r for some integer r. Large values of r give a more complicated model which should fit the data better. But there is a bias–variance tradeoff which we'll discuss later.

22.12 Example. If we use model (22.25) for the heart disease data with $r = 2$, the error rate is reduced to .22. ■

22.5 Relationship Between Logistic Regression and LDA

LDA and logistic regression are almost the same thing. If we assume that each group is Gaussian with the same covariance matrix, then we saw earlier that

$$\begin{aligned}\log\left(\frac{\mathbb{P}(Y=1|X=x)}{\mathbb{P}(Y=0|X=x)}\right) &= \log\left(\frac{\pi_0}{\pi_1}\right) - \frac{1}{2}(\mu_0+\mu_1)^T\Sigma^{-1}(\mu_1-\mu_0) \\ &\quad + x^T\Sigma^{-1}(\mu_1-\mu_0) \\ &\equiv \alpha_0 + \alpha^T x.\end{aligned}$$

On the other hand, the logistic model is, by assumption,

$$\log\left(\frac{\mathbb{P}(Y=1|X=x)}{\mathbb{P}(Y=0|X=x)}\right) = \beta_0 + \beta^T x.$$

These are the same model since they both lead to classification rules that are linear in x. The difference is in how we estimate the parameters.

The joint density of a single observation is $f(x,y) = f(x|y)f(y) = f(y|x)f(x)$. In LDA we estimated the whole joint distribution by maximizing the likelihood

$$\prod_i f(x_i,y_i) = \underbrace{\prod_i f(x_i|y_i)}_{\text{Gaussian}} \underbrace{\prod_i f(y_i)}_{\text{Bernoulli}}. \tag{22.26}$$

In logistic regression we maximized the conditional likelihood $\prod_i f(y_i|x_i)$ but we ignored the second term $f(x_i)$:

$$\prod_i f(x_i,y_i) = \underbrace{\prod_i f(y_i|x_i)}_{\text{logistic}} \underbrace{\prod_i f(x_i)}_{\text{ignored}}. \tag{22.27}$$

Since classification only requires knowing $f(y|x)$, we don't really need to estimate the whole joint distribution. Logistic regression leaves the marginal distribution $f(x)$ unspecified so it is more nonparametric than LDA. This is an advantage of the logistic regression approach over LDA.

To summarize: LDA and logistic regression both lead to a linear classification rule. In LDA we estimate the entire joint distribution $f(x, y) = f(x|y)f(y)$. In logistic regression we only estimate $f(y|x)$ and we don't bother estimating $f(x)$.

22.6 Density Estimation and Naive Bayes

The Bayes rule is $h(x) = \text{argmax}_k\ \pi_k\, f_k(x)$. If we can estimate π_k and f_k then we can estimate the Bayes classification rule. Estimating π_k is easy but what about f_k? We did this previously by assuming f_k was Gaussian. Another strategy is to estimate f_k with some nonparametric density estimator $\widehat{f}_k$ such as a kernel estimator. But if $x = (x_1, \ldots, x_d)$ is high-dimensional, nonparametric density estimation is not very reliable. This problem is ameliorated if we assume that $X_1, \ldots, X_d$ are independent, for then, $f_k(x_1, \ldots, x_d) = \prod_{j=1}^{d} f_{kj}(x_j)$. This reduces the problem to d one-dimensional density estimation problems, within each of the k groups. The resulting classifier is called **the naive Bayes classifier.** The assumption that the components of X are independent is usually wrong yet the resulting classifier might still be accurate. Here is a summary of the steps in the naive Bayes classifier:

The Naive Bayes Classifier

1. For each group k, compute an estimate $\widehat{f}_{kj}$ of the density f_{kj} for X_j, using the data for which $Y_i = k$.

2. Let
$$\widehat{f}_k(x) = \widehat{f}_k(x_1, \ldots, x_d) = \prod_{j=1}^{d} \widehat{f}_{kj}(x_j).$$

3. Let
$$\widehat{\pi}_k = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = k)$$
where $I(Y_i = k) = 1$ if $Y_i = k$ and $I(Y_i = k) = 0$ if $Y_i \neq k$.

4. Let
$$h(x) = \text{argmax}_k\ \widehat{\pi}_k\, \widehat{f}_k(x).$$

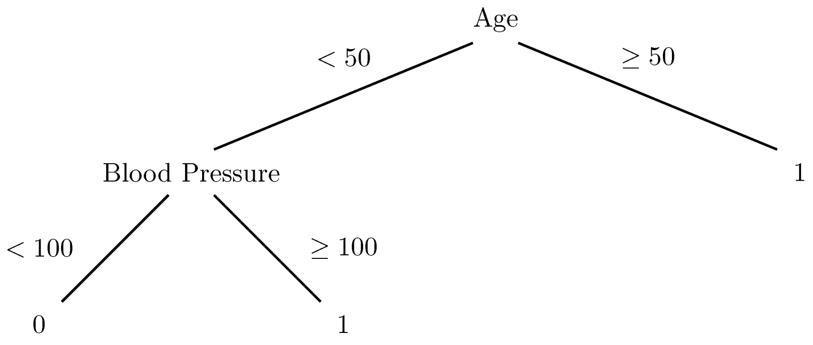

FIGURE 22.2. A simple classification tree.

The naive Bayes classifier is popular when x is high-dimensional and discrete. In that case, $\widehat{f}_{kj}(x_j)$ is especially simple.

22.7 Trees

Trees are classification methods that partition the covariate space $\mathcal{X}$ into disjoint pieces and then classify the observations according to which partition element they fall in. As the name implies, the classifier can be represented as a tree.

For illustration, suppose there are two covariates, $X_1 =$ age and $X_2 =$ blood pressure. Figure 22.2 shows a classification tree using these variables.

The tree is used in the following way. If a subject has Age ≥ 50 then we classify him as $Y = 1$. If a subject has Age < 50 then we check his blood pressure. If systolic blood pressure is < 100 then we classify him as $Y = 1$, otherwise we classify him as $Y = 0$. Figure 22.3 shows the same classifier as a partition of the covariate space.

Here is how a tree is constructed. First, suppose that $y \in \mathcal{Y} = \{0, 1\}$ and that there is only a single covariate X. We choose a split point t that divides the real line into two sets $A_1 = (-\infty, t]$ and $A_2 = (t, \infty)$. Let $\widehat{p}_s(j)$ be the proportion of observations in A_s such that $Y_i = j$:

$$\widehat{p}_s(j) = \frac{\sum_{i=1}^n I(Y_i = j, X_i \in A_s)}{\sum_{i=1}^n I(X_i \in A_s)} \tag{22.28}$$

for $s = 1, 2$ and $j = 0, 1$. The **impurity** of the split t is defined to be

$$I(t) = \sum_{s=1}^{2} \gamma_s \tag{22.29}$$

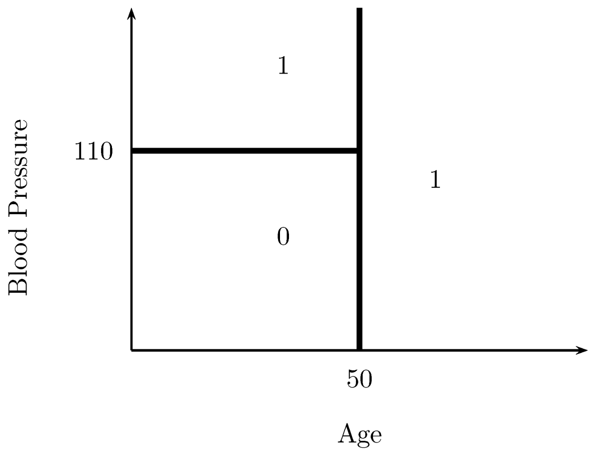

FIGURE 22.3. Partition representation of classification tree.

where

$$\gamma_s = 1 - \sum_{j=0}^{1} \widehat{p}_s(j)^2. \tag{22.30}$$

This particular measure of impurity is known as the **Gini index**. If a partition element A_s contains all 0's or all 1's, then $\gamma_s = 0$. Otherwise, $\gamma_s > 0$. We choose the split point t to minimize the impurity. (Other indices of impurity besides can be used besides the Gini index.)

When there are several covariates, we choose whichever covariate and split that leads to the lowest impurity. This process is continued until some stopping criterion is met. For example, we might stop when every partition element has fewer than n_0 data points, where n_0 is some fixed number. The bottom nodes of the tree are called the **leaves**. Each leaf is assigned a 0 or 1 depending on whether there are more data points with $Y = 0$ or $Y = 1$ in that partition element.

This procedure is easily generalized to the case where $Y \in \{1, \ldots, K\}$. We simply define the impurity by

$$\gamma_s = 1 - \sum_{j=1}^{k} \widehat{p}_s(j)^2 \tag{22.31}$$

where $\widehat{p}_i(j)$ is the proportion of observations in the partition element for which $Y = j$.

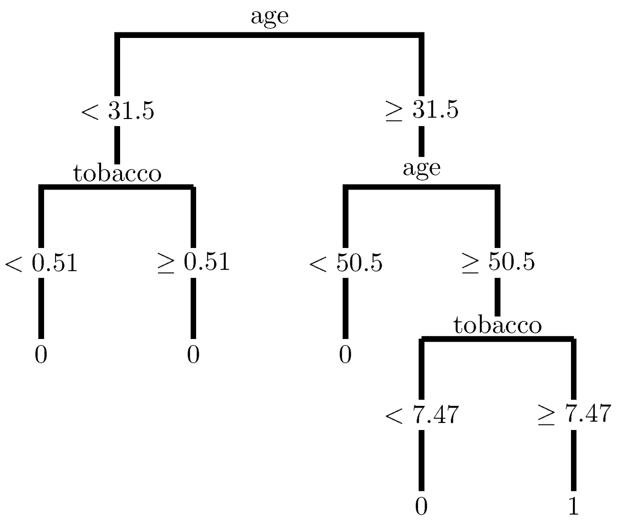

FIGURE 22.4. A classification tree for the heart disease data using two covariates.

22.13 Example. A classification tree for the heart disease data yields a misclassification rate of .21. If we build a tree using only tobacco and age, the misclassification rate is then .29. The tree is shown in Figure 22.4. ■

Our description of how to build trees is incomplete. If we keep splitting until there are few cases in each leaf of the tree, we are likely to overfit the data. We should choose the complexity of the tree in such a way that the estimated true error rate is low. In the next section, we discuss estimation of the error rate.

22.8 Assessing Error Rates and Choosing a Good Classifier

How do we choose a good classifier? We would like to have a classifier h with a low true error rate $L(h)$. Usually, we can't use the training error rate $\widehat{L}_n(h)$ as an estimate of the true error rate because it is biased downward.

22.14 Example. Consider the heart disease data again. Suppose we fit a sequence of logistic regression models. In the first model we include one covariate. In the second model we include two covariates, and so on. The ninth model includes all the covariates. We can go even further. Let's also fit a tenth model that includes all nine covariates plus the first covariate squared. Then we fit an eleventh model that includes all nine covariates plus the first covariate squared and the second covariate squared. Continuing this way we will get a sequence of 18 classifiers of increasing complexity. The solid line in Figure 22.5 shows the observed classification error which steadily decreases as we make the model more complex. If we keep going, we can make a model with zero observed classification error. The dotted line shows the **10-fold cross-validation estimate** of the error rate (to be explained shortly) which is a better estimate of the true error rate than the observed classification error. The estimated error decreases for a while then increases. This is essentially the bias–variance tradeoff phenomenon we have seen in Chapter 20. ■

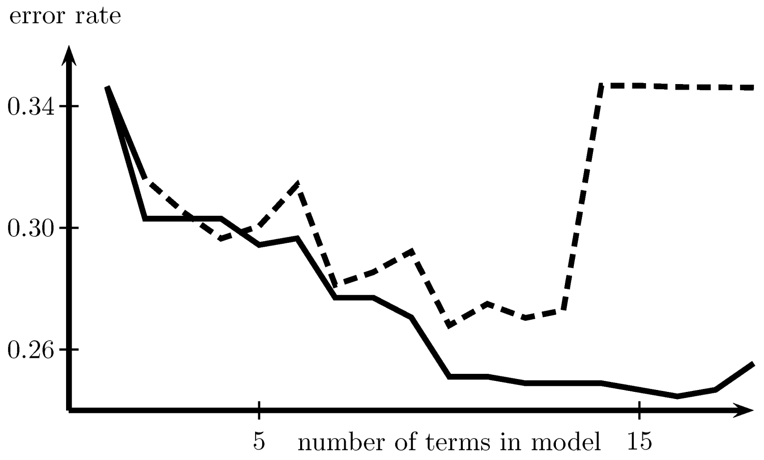

FIGURE 22.5. The solid line is the observed error rate and dashed line is the cross-validation estimate of true error rate.

There are many ways to estimate the error rate. We'll consider two: **cross-validation** and **probability inequalities.**

Cross-Validation. The basic idea of cross-validation, which we have already encountered in curve estimation, is to leave out some of the data when fitting a model. The simplest version of cross-validation involves randomly splitting the data into two pieces: the **training set** $\mathcal{T}$ and the **validation set** $\mathcal{V}$. Often, about 10 per cent of the data might be set aside as the validation set. The classifier h is constructed from the training set. We then estimate

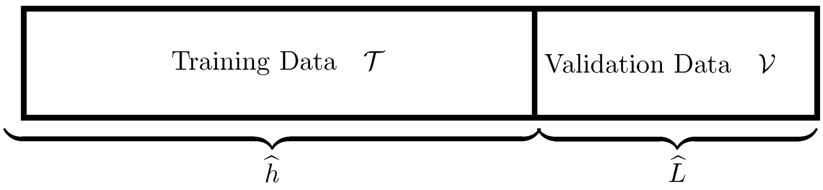

FIGURE 22.6. Cross-validation. The data are divided into two groups: the training data and the validation data. The training data are used to produce an estimated classifier $\widehat{h}$. Then, $\widehat{h}$ is applied to the validation data to obtain an estimate $\widehat{L}$ of the error rate of $\widehat{h}$.

the error by

$$\widehat{L}(h) = \frac{1}{m} \sum_{X_i \in \mathcal{V}} I(h(X_i) \neq Y_I). \tag{22.32}$$

where m is the size of the validation set. See Figure 22.6.

Another approach to cross-validation is **K-fold cross-validation** which is obtained from the following algorithm.

K-fold cross-validation.

1. Randomly divide the data into K chunks of approximately equal size. A common choice is $K = 10$.
2. For k = 1 to K, do the following:
   (a) Delete chunk k from the data.
   (b) Compute the classifier $\widehat{h}_{(k)}$ from the rest of the data.
   (c) Use $\widehat{h}_{(k)}$ to the predict the data in chunk k. Let $\widehat{L}_{(k)}$ denote the observed error rate.
3. Let

$$\widehat{L}(h) = \frac{1}{K} \sum_{k=1}^{K} \widehat{L}_{(k)}. \tag{22.33}$$

22.15 Example. We applied 10-fold cross-validation to the heart disease data. The minimum cross-validation error as a function of the number of leaves occurred at six. Figure 22.7 shows the tree with six leaves. ■

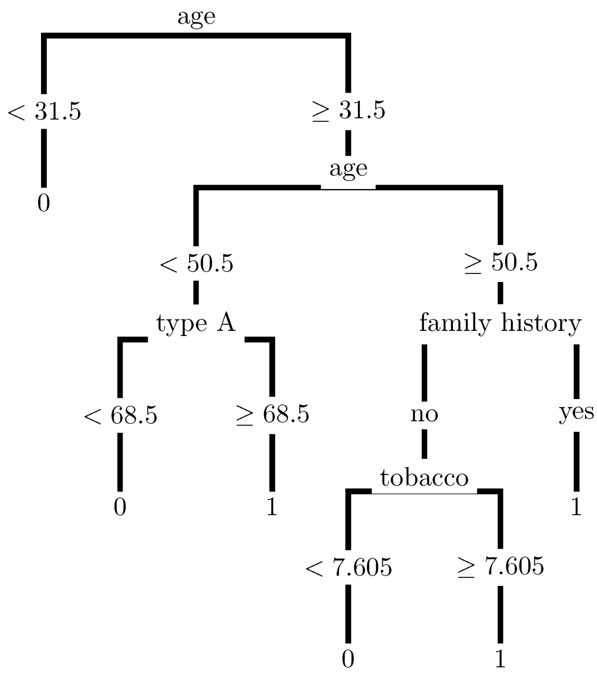

FIGURE 22.7. Smaller classification tree with size chosen by cross-validation.

Probability Inequalities. Another approach to estimating the error rate is to find a confidence interval for $\widehat{L}_n(h)$ using probability inequalities. This method is useful in the context of **empirical risk minimization**.

Let $\mathcal{H}$ be a set of classifiers, for example, all linear classifiers. Empirical risk minimization means choosing the classifier $\widehat{h} \in \mathcal{H}$ to minimize the training error $\widehat{L}_n(h)$, also called the empirical risk. Thus,

$$\widehat{h} = \text{argmin}_{h\in\mathcal{H}}\widehat{L}_n(h) = \text{argmin}_{h\in\mathcal{H}} \left(\frac{1}{n}\sum_i I(h(X_i) \neq Y_i)\right). \tag{22.34}$$

Typically, $\widehat{L}_n(\widehat{h})$ underestimates the true error rate $L(\widehat{h})$ because $\widehat{h}$ was chosen to make $\widehat{L}_n(\widehat{h})$ small. Our goal is to assess how much underestimation is taking place. Our main tool for this analysis is **Hoeffding's inequality** (Theorem 4.5). Recall that if $X_1, \ldots, X_n \sim \text{Bernoulli}(p)$, then, for any $\epsilon > 0$,

$$\mathbb{P}\left(|\widehat{p} - p| > \epsilon\right) \leq 2e^{-2n\epsilon^2} \tag{22.35}$$

where $\widehat{p} = n^{-1}\sum_{i=1}^n X_i$.

First, suppose that $\mathcal{H} = \{h_1, \ldots, h_m\}$ consists of finitely many classifiers. For any fixed h, $\widehat{L}_n(h)$ converges in almost surely to $L(h)$ by the law of large numbers. We will now establish a stronger result.

22.16 Theorem (Uniform Convergence). *Assume $\mathcal{H}$ is finite and has m elements. Then,*

$$\mathbb{P}\left(\max_{h\in\mathcal{H}} |\widehat{L}_n(h) - L(h)| > \epsilon\right) \leq 2me^{-2n\epsilon^2}.$$

Proof. We will use Hoeffding's inequality and we will also use the fact that if $A_1, \ldots, A_m$ is a set of events then $\mathbb{P}(\bigcup_{i=1}^m A_i) \leq \sum_{i=1}^m \mathbb{P}(A_i)$. Now,

$$\begin{aligned}\mathbb{P}\left(\max_{h\in\mathcal{H}} |\widehat{L}_n(h) - L(h)| > \epsilon\right) &= \mathbb{P}\left(\bigcup_{h\in\mathcal{H}} |\widehat{L}_n(h) - L(h)| > \epsilon\right)\\ &\leq \sum_{H\in\mathcal{H}} \mathbb{P}\left(|\widehat{L}_n(h) - L(h)| > \epsilon\right)\\ &\leq \sum_{H\in\mathcal{H}} 2e^{-2n\epsilon^2} = 2me^{-2n\epsilon^2}. \quad \blacksquare\end{aligned}$$

22.17 Theorem. *Let*

$$\epsilon = \sqrt{\frac{2}{n}\log\left(\frac{2m}{\alpha}\right)}.$$

Then $\widehat{L}_n(\widehat{h}) \pm \epsilon$ is a $1-\alpha$ confidence interval for $L(\widehat{h})$.

Proof. This follows from the fact that

$$\begin{aligned}\mathbb{P}(|\widehat{L}_n(\widehat{h}) - L(\widehat{h})| > \epsilon) &\leq \mathbb{P}\left(\max_{h\in\mathcal{H}} |\widehat{L}_n(\widehat{h}) - L(\widehat{h})| > \epsilon\right)\\ &\leq 2me^{-2n\epsilon^2} = \alpha. \quad \blacksquare\end{aligned}$$

When $\mathcal{H}$ is large the confidence interval for $L(\widehat{h})$ is large. The more functions there are in $\mathcal{H}$ the more likely it is we have "overfit" which we compensate for by having a larger confidence interval.

In practice we usually use sets $\mathcal{H}$ that are infinite, such as the set of linear classifiers. To extend our analysis to these cases we want to be able to say something like

$$\mathbb{P}\left(\sup_{h\in\mathcal{H}} |\widehat{L}_n(h) - L(h)| > \epsilon\right) \leq \text{something not too big.}$$

One way to develop such a generalization is by way of the **Vapnik-Chervonenkis** or **VC dimension.**

Let $\mathcal{A}$ be a class of sets. Give a finite set $F = \{x_1, \ldots, x_n\}$ let

$$N_{\mathcal{A}}(F) = \#\left\{F \bigcap A : \ A \in \mathcal{A}\right\} \tag{22.36}$$

be the number of subsets of F "picked out" by $\mathcal{A}$. Here $\#(B)$ denotes the number of elements of a set B. The **shatter coefficient** is defined by

$$s(\mathcal{A}, n) = \max_{F \in \mathcal{F}_n} N_{\mathcal{A}}(F) \tag{22.37}$$

where $\mathcal{F}_n$ consists of all finite sets of size n. Now let $X_1, \ldots, X_n \sim \mathbb{P}$ and let

$$\mathbb{P}_n(A) = \frac{1}{n} \sum_i I(X_i \in A)$$

denote the **empirical probability measure**. The following remarkable theorem bounds the distance between $\mathbb{P}$ and $\mathbb{P}_n$.

22.18 Theorem (Vapnik and Chervonenkis (1971)). *For any $\mathbb{P}$, n and $\epsilon > 0$,*

$$\mathbb{P}\left\{\sup_{A \in \mathcal{A}} |\mathbb{P}_n(A) - \mathbb{P}(A)| > \epsilon\right\} \leq 8 s(\mathcal{A}, n) e^{-n\epsilon^2/32}. \tag{22.38}$$

The proof, though very elegant, is long and we omit it. If $\mathcal{H}$ is a set of classifiers, define $\mathcal{A}$ to be the class of sets of the form $\{x : \ h(x) = 1\}$. We then define $s(\mathcal{H}, n) = s(\mathcal{A}, n)$.

22.19 Theorem.

$$\mathbb{P}\left\{\sup_{h \in \mathcal{H}} |\widehat{L}_n(h) - L(h)| > \epsilon\right\} \leq 8 s(\mathcal{H}, n) e^{-n\epsilon^2/32}.$$

A $1 - \alpha$ confidence interval for $L(\widehat{h})$ is $\widehat{L}_n(\widehat{h}) \pm \epsilon_n$ where

$$\epsilon_n^2 = \frac{32}{n} \log\left(\frac{8 s(\mathcal{H}, n)}{\alpha}\right).$$

These theorems are only useful if the shatter coefficients do not grow too quickly with n. This is where VC dimension enters.

22.20 Definition. *The VC (Vapnik-Chervonenkis) dimension of a class of sets $\mathcal{A}$ is defined as follows. If $s(\mathcal{A}, n) = 2^n$ for all n, set $VC(\mathcal{A}) = \infty$. Otherwise, define $VC(\mathcal{A})$ to be the largest k for which $s(\mathcal{A}, n) = 2^k$.*

Thus, the VC-dimension is the size of the largest finite set F that can be **shattered** by $\mathcal{A}$ meaning that $\mathcal{A}$ picks out each subset of F. If $\mathcal{H}$ is a set of classifiers we define $VC(\mathcal{H}) = VC(\mathcal{A})$ where $\mathcal{A}$ is the class of sets of the form $\{x : \ h(x) = 1\}$ as h varies in $\mathcal{H}$. The following theorem shows that if $\mathcal{A}$ has finite VC-dimension, then the shatter coefficients grow as a polynomial in n.

22.21 Theorem. *If $\mathcal{A}$ has finite VC-dimension v, then*

$$s(\mathcal{A}, n) \leq n^v + 1.$$

22.22 Example. Let $\mathcal{A} = \{(-\infty, a];\ a \in \mathcal{R}\}$. The $\mathcal{A}$ shatters every 1-point set $\{x\}$ but it shatters no set of the form $\{x, y\}$. Therefore, $VC(\mathcal{A}) = 1$. ∎

22.23 Example. Let $\mathcal{A}$ be the set of closed intervals on the real line. Then $\mathcal{A}$ shatters $S = \{x, y\}$ but it cannot shatter sets with 3 points. Consider $S = \{x, y, z\}$ where $x < y < z$. One cannot find an interval A such that $A \bigcap S = \{x, z\}$. So, $VC(\mathcal{A}) = 2$. ∎

22.24 Example. Let $\mathcal{A}$ be all linear half-spaces on the plane. Any 3-point set (not all on a line) can be shattered. No 4 point set can be shattered. Consider, for example, 4 points forming a diamond. Let T be the left and rightmost points. This can't be picked out. Other configurations can also be seen to be unshatterable. So $VC(\mathcal{A}) = 3$. In general, halfspaces in $\mathcal{R}^d$ have VC dimension $d + 1$. ∎

22.25 Example. Let $\mathcal{A}$ be all rectangles on the plane with sides parallel to the axes. Any 4 point set can be shattered. Let S be a 5 point set. There is one point that is not leftmost, rightmost, uppermost, or lowermost. Let T be all points in S except this point. Then T can't be picked out. So $VC(\mathcal{A}) = 4$. ∎

22.26 Theorem. *Let x have dimension d and let $\mathcal{H}$ be th set of linear classifiers. The VC-dimension of $\mathcal{H}$ is $d+1$. Hence, a $1-\alpha$ confidence interval for the true error rate is $\widehat{L}(\widehat{h}) \pm \epsilon$ where*

$$\epsilon_n^2 = \frac{32}{n} \log\left(\frac{8(n^{d+1}+1)}{\alpha}\right).$$

22.9 Support Vector Machines

In this section we consider a class of linear classifiers called **support vector machines**. Throughout this section, we assume that Y is binary. It will be convenient to label the outcomes as -1 and $+1$ instead of 0 and 1. A linear classifier can then be written as

$$h(x) = \text{sign}\Big(H(x)\Big)$$

where $x = (x_1, \ldots, x_d)$,

$$H(x) = a_0 + \sum_{i=1}^{d} a_i x_i$$

and

$$\text{sign}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ 1 & \text{if } z > 0. \end{cases}$$

First, suppose that the data are **linearly separable**, that is, there exists a hyperplane that perfectly separates the two classes.

22.27 Lemma. *The data can be separated by some hyperplane if and only if there exists a hyperplane* $H(x) = a_0 + \sum_{i=1}^{d} a_i x_i$ *such that*

$$Y_i H(x_i) \geq 1, \quad i = 1, \ldots, n. \tag{22.39}$$

PROOF. Suppose the data can be separated by a hyperplane $W(x) = b_0 + \sum_{i=1}^{d} b_i x_i$. It follows that there exists some constant c such that $Y_i = 1$ implies $W(X_i) \geq c$ and $Y_i = -1$ implies $W(X_i) \leq -c$. Therefore, $Y_i W(X_i) \geq c$ for all i. Let $H(x) = a_0 + \sum_{i=1}^{d} a_i x_i$ where $a_j = b_j / c$. Then $Y_i H(X_i) \geq 1$ for all i. The reverse direction is straightforward. ■

In the separable case, there will be many separating hyperplanes. How should we choose one? Intuitively, it seems reasonable to choose the hyperplane "furthest" from the data in the sense that it separates the +1s and -1s and maximizes the distance to the closest point. This hyperplane is called the **maximum margin hyperplane.** The margin is the distance to from the hyperplane to the nearest point. Points on the boundary of the margin are called **support vectors.** See Figure 22.8.

22.28 Theorem. *The hyperplane* $\widehat{H}(x) = \widehat{a}_0 + \sum_{i=1}^{d} \widehat{a}_i x_i$ *that separates the data and maximizes the margin is given by minimizing* $(1/2) \sum_{j=1}^{d} a_j^2$ *subject to (22.39).*

It turns out that this problem can be recast as a quadratic programming problem. Let $\langle X_i, X_k \rangle = X_i^T X_k$ denote the inner product of X_i and X_k.

22.29 Theorem. *Let* $\widehat{H}(x) = \widehat{a}_0 + \sum_{i=1}^{d} \widehat{a}_i x_i$ *denote the optimal (largest margin) hyperplane. Then, for* $j = 1, \ldots, d$,

$$\widehat{a}_j = \sum_{i=1}^{n} \widehat{\alpha}_i Y_i X_j(i)$$

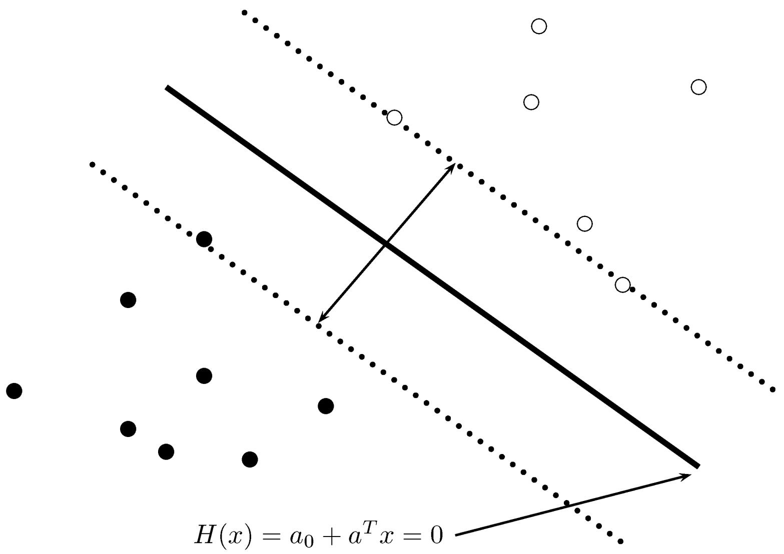

FIGURE 22.8. The hyperplane $H(x)$ has the largest margin of all hyperplanes that separate the two classes.

where $X_j(i)$ is the value of the covariate X_j for the i*th data point, and $\widehat{\alpha} = (\widehat{\alpha}_1, \ldots, \widehat{\alpha}_n)$ is the vector that maximizes*

$$\sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{k=1}^n \alpha_i \alpha_k Y_i Y_k \langle X_i, X_k \rangle \tag{22.40}$$

subject to

$$\alpha_i \geq 0$$

and

$$0 = \sum_i \alpha_i Y_i .$$

The points X_i for which $\widehat{\alpha} \neq 0$ are called **support vectors**. *$\widehat{a}_0$ can be found by solving*

$$\widehat{\alpha}_i \left(Y_i (X_i^T \widehat{a} + \widehat{\beta}_0 \right) = 0$$

for any support point X_i. $\widehat{H}$ may be written as

$$\widehat{H}(x) = \widehat{\alpha}_0 + \sum_{i=1}^n \widehat{\alpha}_i Y_i \langle x, X_i \rangle .$$

There are many software packages that will solve this problem quickly. If there is no perfect linear classifier, then one allows overlap between the groups by replacing the condition (22.39) with

$$Y_i H(x_i) \geq 1 - \xi_i, \quad \xi_i \geq 0, \ \ i = 1, \ldots, n. \tag{22.41}$$

The variables $\xi_1, \ldots, \xi_n$ are called **slack variables**.

We now maximize (22.40) subject to

$$0 \leq \xi_i \leq c, \quad i = 1, \ldots, n$$

and

$$\sum_{i=1}^{n} \alpha_i Y_i = 0.$$

The constant c is a tuning parameter that controls the amount of overlap.

22.10 Kernelization

There is a trick called **kernelization** for improving a computationally simple classifier h. The idea is to map the covariate X — which takes values in $\mathcal{X}$ — into a higher dimensional space $\mathcal{Z}$ and apply the classifier in the bigger space $\mathcal{Z}$. This can yield a more flexible classifier while retaining computationally simplicity.

The standard example of this idea is illustrated in Figure 22.9. The covariate $x = (x_1, x_2)$. The Y_is can be separated into two groups using an ellipse. Define a mapping ϕ by

$$z = (z_1, z_2, z_3) = \phi(x) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2).$$

Thus, ϕ maps $\mathcal{X} = \mathbb{R}^2$ into $\mathcal{Z} = \mathbb{R}^3$. In the higher-dimensional space $\mathcal{Z}$, the Y_i's are separable by a linear decision boundary. In other words,

> a linear classifier in a higher-dimensional space corresponds to a non-linear classifier in the original space.

The point is that to get a richer set of classifiers we do not need to give up the convenience of linear classifiers. We simply map the covariates to a higher-dimensional space. This is akin to making linear regression more flexible by using polynomials.

There is a potential drawback. If we significantly expand the dimension of the problem, we might increase the computational burden. For example, if x has dimension $d = 256$ and we wanted to use all fourth-order terms, then $z = \phi(x)$ has dimension 183,181,376. We are spared this computational

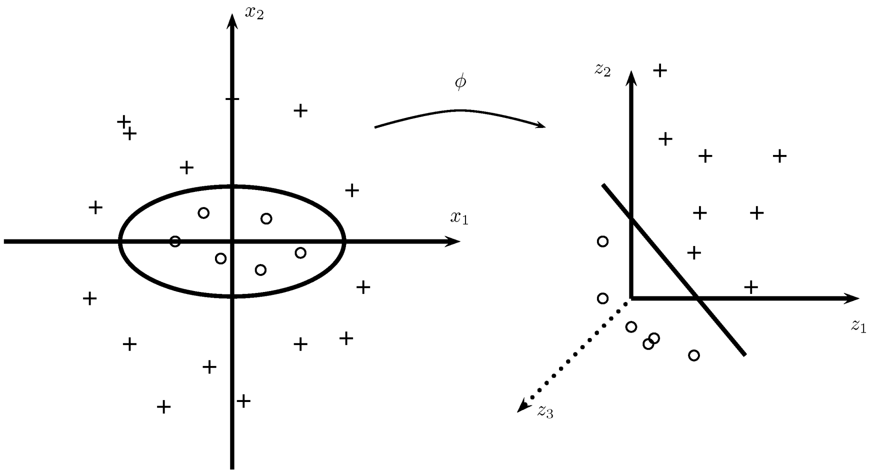

FIGURE 22.9. Kernelization. Mapping the covariates into a higher-dimensional space can make a complicated decision boundary into a simpler decision boundary.

nightmare by the following two facts. First, many classifiers do not require that we know the values of the individual points but, rather, just the inner product between pairs of points. Second, notice in our example that the inner product in $\mathcal{Z}$ can be written

$$\begin{aligned}\langle z, \widetilde{z}\rangle &= \langle \phi(x), \phi(\widetilde{x})\rangle \\ &= x_1^2\widetilde{x}_1^2 + 2x_1\widetilde{x}_1x_2\widetilde{x}_2 + x_2^2\widetilde{x}_2^2 \\ &= (\langle x, \widetilde{x}\rangle)^2 \equiv K(x, \widetilde{x}).\end{aligned}$$

Thus, we can compute $\langle z, \widetilde{z}\rangle$ without ever computing $Z_i = \phi(X_i)$.

To summarize, kernelization involves finding a mapping $\phi : \mathcal{X} \to \mathcal{Z}$ and a classifier such that:

1. $\mathcal{Z}$ has higher dimension than $\mathcal{X}$ and so leads a richer set of classifiers.
2. The classifier only requires computing inner products.
3. There is a function K, called a kernel, such that $\langle \phi(x), \phi(\widetilde{x})\rangle = K(x, \widetilde{x})$.
4. Everywhere the term $\langle x, \widetilde{x}\rangle$ appears in the algorithm, replace it with $K(x, \widetilde{x})$.

In fact, we never need to construct the mapping ϕ at all. We only need to specify a kernel $K(x, \widetilde{x})$ that corresponds to $\langle\phi(x), \phi(\widetilde{x})\rangle$ for some ϕ. This raises an interesting question: given a function of two variables $K(x, y)$, does there exist a function $\phi(x)$ such that $K(x, y) = \langle\phi(x), \phi(y)\rangle$? The answer is provided by **Mercer's theorem** which says, roughly, that if K is positive definite — meaning that

$$\int\int K(x, y)f(x)f(y)dxdy \geq 0$$

for square integrable functions f — then such a ϕ exists. Examples of commonly used kernels are:

$$\begin{array}{rrcl} \text{polynomial} & K(x, \widetilde{x}) & = & \Bigl(\langle x, \widetilde{x}\rangle + a\Bigr)^r \\ \text{sigmoid} & K(x, \widetilde{x}) & = & \tanh(a\langle x, \widetilde{x}\rangle + b) \\ \text{Gaussian} & K(x, \widetilde{x}) & = & \exp\Bigl(-\|x - \widetilde{x}\|^2/(2\sigma^2)\Bigr) \end{array}$$

Let us now see how we can use this trick in LDA and in support vector machines.

Recall that the Fisher linear discriminant method replaces X with $U = w^T X$ where w is chosen to maximize the Rayleigh coefficient

$$J(w) = \frac{w^T S_B w}{w^T S_W w},$$

$$S_B = (\overline{X}_0 - \overline{X}_1)(\overline{X}_0 - \overline{X}_1)^T$$

and

$$S_W = \left(\frac{(n_0 - 1)S_0}{(n_0 - 1) + (n_1 - 1)}\right) + \left(\frac{(n_1 - 1)S_1}{(n_0 - 1) + (n_1 - 1)}\right).$$

In the kernelized version, we replace X_i with $Z_i = \phi(X_i)$ and we find w to maximize

$$J(w) = \frac{w^T \widetilde{S}_B w}{w^T \widetilde{S}_W w} \tag{22.42}$$

where

$$\widetilde{S}_B = (\overline{Z}_0 - \overline{Z}_1)(\overline{Z}_0 - \overline{Z}_1)^T$$

and

$$S_W = \left(\frac{(n_0 - 1)\widetilde{S}_0}{(n_0 - 1) + (n_1 - 1)}\right) + \left(\frac{(n_1 - 1)\widetilde{S}_1}{(n_0 - 1) + (n_1 - 1)}\right).$$

Here, $\widetilde{S}_j$ is the sample of covariance of the Z_i's for which $Y = j$. However, to take advantage of kernelization, we need to re-express this in terms of inner products and then replace the inner products with kernels.

It can be shown that the maximizing vector w is a linear combination of the Z_i's. Hence we can write

$$w = \sum_{i=1}^{n} \alpha_i Z_i.$$

Also,

$$\overline{Z}_j = \frac{1}{n_j} \sum_{i=1}^{n} \phi(X_i) I(Y_i = j).$$

Therefore,

$$\begin{aligned} w^T \overline{Z}_j &= \left(\sum_{i=1}^{n} \alpha_i Z_i \right)^T \left(\frac{1}{n_j} \sum_{i=1}^{n} \phi(X_i) I(Y_i = j) \right) \\ &= \frac{1}{n_j} \sum_{i=1}^{n} \sum_{s=1}^{n} \alpha_i I(Y_s = j) Z_i^T \phi(X_s) \\ &= \frac{1}{n_j} \sum_{i=1}^{n} \alpha_i \sum_{s=1}^{n} I(Y_s = j) \phi(X_i)^T \phi(X_s) \\ &= \frac{1}{n_j} \sum_{i=1}^{n} \alpha_i \sum_{s=1}^{n} I(Y_s = j) K(X_i, X_s) \\ &= \alpha^T M_j \end{aligned}$$

where M_j is a vector whose i^{th} component is

$$M_j(i) = \frac{1}{n_j} \sum_{s=1}^{n} K(X_i, X_s) I(Y_i = j).$$

It follows that

$$w^T \widetilde{S}_B w = \alpha^T M \alpha$$

where $M = (M_0 - M_1)(M_0 - M_1)^T$. By similar calculations, we can write

$$w^T \widetilde{S}_W w = \alpha^T N \alpha$$

where

$$N = K_0 \left(I - \frac{1}{n_0} \mathbf{1} \right) K_0^T + K_1 \left(I - \frac{1}{n_1} \mathbf{1} \right) K_1^T,$$

I is the identity matrix, $\mathbf{1}$ is a matrix of all one's, and K_j is the $n \times n_j$ matrix with entries $(K_j)_{rs} = K(x_r, x_s)$ with x_s varying over the observations in group j. Hence, we now find α to maximize

$$J(\alpha) = \frac{\alpha^T M \alpha}{\alpha^T N \alpha}.$$

All the quantities are expressed in terms of the kernel. Formally, the solution is $\alpha = N^{-1}(M_0 - M_1)$. However, N might be non-invertible. In this case one replaces N by $N + bI$, for some constant b. Finally, the projection onto the new subspace can be written as

$$U = w^T \phi(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x).$$

The support vector machine can similarly be kernelized. We simply replace $\langle X_i, X_j \rangle$ with $K(X_i, X_j)$. For example, instead of maximizing (22.40), we now maximize

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_i \alpha_k Y_i Y_k K(X_i, X_j). \tag{22.43}$$

The hyperplane can be written as $\widehat{H}(x) = \widehat{a}_0 + \sum_{i=1}^{n} \widehat{\alpha}_i Y_i K(X, X_i)$.

22.11 Other Classifiers

There are many other classifiers and space precludes a full discussion of all of them. Let us briefly mention a few.

The **k-nearest-neighbors** classifier is very simple. Given a point x, find the k data points closest to x. Classify x using the majority vote of these k neighbors. Ties can be broken randomly. The parameter k can be chosen by cross-validation.

Bagging is a method for reducing the variability of a classifier. It is most helpful for highly nonlinear classifiers such as trees. We draw B bootstrap samples from the data. The b^{th} bootstrap sample yields a classifier h_b. The final classifier is

$$\widehat{h}(x) = \begin{cases} 1 & \text{if } \frac{1}{B}\sum_{b=1}^{B} h_b(x) \geq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$

Boosting is a method for starting with a simple classifier and gradually improving it by refitting the data giving higher weight to misclassified samples. Suppose that $\mathcal{H}$ is a collection of classifiers, for example, trees with only one split. Assume that $Y_i \in \{-1, 1\}$ and that each h is such that $h(x) \in \{-1, 1\}$. We usually give equal weight to all data points in the methods we have discussed. But one can incorporate unequal weights quite easily in most algorithms. For example, in constructing a tree, we could replace the impurity measure with a weighted impurity measure. The original version of boosting, called AdaBoost, is as follows.

1. Set the weights $w_i = 1/n$, $i = 1, \ldots, n$.

2. For $j = 1, \ldots, J$, do the following steps:

   (a) Constructing a classifier h_j from the data using the weights $w_1, \ldots, w_n$.

   (b) Compute the weighted error estimate:

   $$\widehat{L}_j = \frac{\sum_{i=1}^n w_i I(Y_i \neq h_j(X_i))}{\sum_{i=1}^n w_i}.$$

   (c) Let $\alpha_j = \log((1 - \widehat{L}_j)/\widehat{L}_j)$.

   (d) Update the weights:

   $$w_i \longleftarrow w_i e^{\alpha_j I(Y_i \neq h_j(X_i))}$$

3. The final classifier is

   $$\widehat{h}(x) = \text{sign}\left(\sum_{j=1}^J \alpha_j h_j(x)\right).$$

There is now an enormous literature trying to explain and improve on boosting. Whereas bagging is a variance reduction technique, boosting can be thought of as a bias reduction technique. We starting with a simple — and hence highly-biased — classifier, and we gradually reduce the bias. The disadvantage of boosting is that the final classifier is quite complicated.

Neural Networks are regression models of the form [3]

$$Y = \beta_0 + \sum_{j=1}^p \beta_j \sigma(\alpha_0 + \alpha^T X)$$

where σ is a smooth function, often taken to be $\sigma(v) = e^v/(1 + e^v)$. This is really nothing more than a nonlinear regression model. Neural nets were fashionable for some time but they pose great computational difficulties. In particular, one often encounters multiple minima when trying to find the least squares estimates of the parameters. Also, the number of terms p is essentially a smoothing parameter and there is the usual problem of trying to choose p to find a good balance between bias and variance.

[3] This is the simplest version of a neural net. There are more complex versions of the model.

22.12 Bibliographic Remarks

The literature on classification is vast and is growing quickly. An excellent reference is Hastie et al. (2001). For more on the theory, see Devroye et al. (1996) and Vapnik (1998). Two recent books on kernels are Scholkopf and Smola (2002) and Herbich (2002).

22.13 Exercises

1. Prove Theorem 22.5.
2. Prove Theorem 22.7.
3. Download the spam data from:

   http://www-stat.stanford.edu/~tibs/ElemStatLearn/index.html

   The data file can also be found on the course web page. The data contain 57 covariates relating to email messages. Each email message was classified as spam (Y=1) or not spam (Y=0). The outcome Y is the last column in the file. The goal is to predict whether an email is spam or not.

   (a) Construct classification rules using (i) LDA, (ii) QDA, (iii) logistic regression, and (iv) a classification tree. For each, report the observed misclassification error rate and construct a 2-by-2 table of the form

	$\widehat{h}(x) = 0$	$\widehat{h}(x) = 1$
$Y = 0$	??	??
$Y = 1$	??	??

   (b) Use 5-fold cross-validation to estimate the prediction accuracy of LDA and logistic regression.

   (c) Sometimes it helps to reduce the number of covariates. One strategy is to compare X_i for the spam and email group. For each of the 57 covariates, test whether the mean of the covariate is the same or different between the two groups. Keep the 10 covariates with the smallest p-values. Try LDA and logistic regression using only these 10 variables.

4. Let $\mathcal{A}$ be the set of two-dimensional spheres. That is, $A \in \mathcal{A}$ if $A = \{(x,y) : (x-a)^2+(y-b)^2 \le c^2\}$ for some a, b, c. Find the VC-dimension of $\mathcal{A}$.

5. Classify the spam data using support vector machines. Free software for the support vector machine is at http://svmlight.joachims.org/

6. Use VC theory to get a confidence interval on the true error rate of the LDA classifier for the iris data (from the book web site).

7. Suppose that $X_i \in \mathbb{R}$ and that $Y_i = 1$ whenever $|X_i| \le 1$ and $Y_i = 0$ whenever $|X_i| > 1$. Show that no linear classifier can perfectly classify these data. Show that the kernelized data $Z_i = (X_i, X_i^2)$ can be linearly separated.

8. Repeat question 5 using the kernel $K(x, \widetilde{x}) = (1 + x^T\widetilde{x})^p$. Choose p by cross-validation.

9. Apply the k nearest neighbors classifier to the "iris data." Choose k by cross-validation.

10. (Curse of Dimensionality.) Suppose that X has a uniform distribution on the d-dimensional cube $[-1/2, 1/2]^d$. Let R be the distance from the origin to the closest neighbor. Show that the median of R is

$$\left(\frac{\left(1-\left(\frac{1}{2}\right)^{1/n}\right)}{v_d(1)}\right)^{1/d}$$

where

$$v_d(r) = r^d \frac{\pi^{d/2}}{\Gamma((d/2)+1)}$$

is the volume of a sphere of radius r. For what dimension d does the median of R exceed the edge of the cube when $n = 100$, $n = 1,000$, $n = 10,000$? (Hastie et al. (2001), p. 22–27.)

11. Fit a tree to the data in question 3. Now apply bagging and report your results.

12. Fit a tree that uses only one split on one variable to the data in question 3. Now apply boosting.

13. Let $r(x) = \mathbb{P}(Y = 1|X = x)$ and let $\widehat{r}(x)$ be an estimate of $r(x)$. Consider the classifier

$$h(x) = \begin{cases} 1 & \text{if } \widehat{r}(x) \geq 1/2 \\ 0 & \text{otherwise.} \end{cases}$$

Assume that $\widehat{r}(x) \approx N(\overline{r}(x), \sigma^2(x))$ for some functions $\overline{r}(x)$ and $\sigma^2(x)$. Show that, for fixed x,

$$\begin{aligned}\mathbb{P}(Y \neq h(x)) &\approx \mathbb{P}(Y \neq h^*(x)) \\ &\quad + \left|2r(x) - 1\right| \times \left[1 - \Phi\left(\frac{\text{sign}\Big(r(x) - (1/2)\Big)(\overline{r}(x) - (1/2))}{\sigma(x)}\right)\right]\end{aligned}$$

where Φ is the standard Normal CDF and h^* is the Bayes rule. Regard $\text{sign}\Big((r(x) - (1/2))(\overline{r}(x) - (1/2))\Big)$ as a type of bias term. Explain the implications for the bias–variance tradeoff in classification (Friedman (1997)).

Hint: first show that

$$\mathbb{P}(Y \neq h(x)) = |2r(x) - 1|\mathbb{P}(h(x) \neq h^*(x)) + \mathbb{P}(Y \neq h^*(x)).$$