CHAPTER 1

CLASSIFICATION

A classifier is designed and its error estimated from sample data taken from a population. Two fundamental issues arise. First, given a set of features (random variables), how does one design a classifier from the sample data that provides good classification over the general population? Second, how does one estimate the error of a classifier from the data? This chapter gives the essentials regarding the first question and the remainder of the book addresses the second.

1.1 CLASSIFIERS

Formally, classification involves a predictor vector $\mathbf{X} = (X_1, \ldots, X_d) \in R^d$, also known as a *feature vector*, where each $X_i \in R$ is a *feature*, for $i = 1, \ldots, d$. The feature vector $\mathbf{X}$ represents an individual from one of two populations Π_0 or Π_1.[1] The classification problem is to assign $\mathbf{X}$ correctly to its population of origin. Following the majority of the literature in pattern recognition, the populations are coded into a discrete *label* $Y \in \{0, 1\}$. Therefore, given a feature vector $\mathbf{X}$, classification attempts to predict the corresponding value of the label Y. The relationship between $\mathbf{X}$ and Y is assumed to be stochastic in nature; that is, we assume that there is a joint *feature–label distribution* $F_{\mathbf{X}Y}$ for the pair $(\mathbf{X}, Y)$.[2] The feature–label

[1] For simplicity and clarity, only the two-population case is discussed in the book; however, the extension to more than two populations is considered in the problem sections.

[2] A review of basic probability concepts used in the book is provided in Appendix A.

Error Estimation for Pattern Recognition, First Edition. Ulisses M. Braga-Neto and Edward R. Dougherty.

distribution completely characterizes the classification problem. It determines the probabilities $c_0 = P(\mathbf{X} \in \Pi_0) = P(Y = 0)$ and $c_1 = P(\mathbf{X} \in \Pi_1) = P(Y = 1)$, which are called the *prior probabilities*, as well as (if they exist) the probability densities $p_0(\mathbf{x}) = p(\mathbf{x} \mid Y = 0)$ and $p_1(\mathbf{x}) = p(\mathbf{x} \mid Y = 1)$, which are called the *class-conditional densities*. (In the discrete case, these are replaced by the corresponding probability mass functions.) To avoid trivialities, we shall always assume that $\min\{c_0, c_1\} \neq 0$. The *posterior probabilities* are defined by $\eta_0(\mathbf{x}) = P(Y = 0 \mid \mathbf{X} = \mathbf{x})$ and $\eta_1(\mathbf{x}) = P(Y = 1 \mid \mathbf{X} = \mathbf{x})$. Note that $\eta_0(\mathbf{x}) = 1 - \eta_1(\mathbf{x})$. If densities exist, then $\eta_0(\mathbf{x}) = c_0 p_0(\mathbf{x})/p(\mathbf{x})$ and $\eta_1(\mathbf{x}) = c_1 p_1(\mathbf{x})/p(\mathbf{x})$, where $p(\mathbf{x}) = c_0 p_0(\mathbf{x}) + c_1 p_1(\mathbf{x})$ is the density of $\mathbf{X}$ across the mixture of populations Π_0 and Π_1. Furthermore, in this binary-label setting, $\eta_1(\mathbf{x})$ is also the conditional expectation of Y given $\mathbf{X}$; that is, $\eta_1(\mathbf{x}) = E[Y|\mathbf{X} = \mathbf{x}]$.

A *classifier* is a measurable function $\psi : R^d \to \{0, 1\}$, which is to serve as a predictor of Y; that is, Y is to be predicted by $\psi(\mathbf{X})$. The error $\varepsilon[\psi]$ is the probability that the classification is erroneous, namely, $\varepsilon[\psi] = P(\psi(\mathbf{X}) \neq Y)$, which is a function both of ψ and the feature–label distribution $F_{\mathbf{X}Y}$. Owing to the binary nature of $\psi(\mathbf{X})$ and Y, we have that $\varepsilon[\psi] = E[|Y - \psi(\mathbf{X})|] = E[(Y - \psi(\mathbf{X}))^2]$, so that the classification error equals both the mean absolute deviation (MAD) and mean-square error (MSE) of predicting Y from $\mathbf{X}$ using ψ. If the class-conditional densities exist, then

$$\varepsilon[\psi] = \int_{\{\mathbf{x}|\psi(\mathbf{x})=0\}} \eta_1(\mathbf{x}) p(\mathbf{x})\, d\mathbf{x} + \int_{\{\mathbf{x}|\psi(\mathbf{x})=1\}} \eta_0(\mathbf{x}) p(\mathbf{x})\, d\mathbf{x}. \tag{1.1}$$

An optimal classifier ψ_{bay} is one having minimal classification error $\varepsilon_{\text{bay}} = \varepsilon[\psi_{\text{bay}}]$ — note that this makes ψ_{bay} also the minimum MAD and minimum MSE predictor of Y given $\mathbf{X}$. An optimal classifier ψ_{bay} is called a *Bayes classifier* and its error ε_{bay} is called the *Bayes error*. A Bayes classifier and the Bayes error depend on the feature–label distribution $F_{\mathbf{X}Y}$ — how well the labels are distributed among the variables being used to discriminate them, and how the variables are distributed in R^d. The Bayes error measures the inherent discriminability of the labels and is intrinsic to the feature–label distribution. Clearly, the right-hand side of (1.1) is minimized by the classifier

$$\psi_{\text{bay}}(\mathbf{x}) = \begin{cases} 1, & \eta_1(\mathbf{x}) \geq \eta_0(\mathbf{x}), \\ 0, & \text{otherwise.} \end{cases} \tag{1.2}$$

In other words, $\psi_{\text{bay}}(\mathbf{x})$ is defined to be 0 or 1 according to whether Y is more likely to be 0 or 1 given $\mathbf{X} = \mathbf{x}$. For this reason, a Bayes classifier is also known as a *maximum a posteriori* (MAP) classifier. The inequality can be replaced by strict inequality without affecting optimality; in fact, ties between the posterior probabilities can be broken arbitrarily, yielding a possibly infinite number of distinct Bayes classifiers. Furthermore, the derivation of a Bayes classifier does not require the existence of class-conditional densities, as has been assumed here; for an entirely general proof that ψ_{bay} in (1.2) is optimal, see Devroye et al. (1996, Theorem 2.1).

All Bayes classifiers naturally share the same Bayes error, which, from (1.1) and (1.2), is given by

$$\begin{aligned}\varepsilon_{\text{bay}} &= \int_{\{\mathbf{x}|\eta_1(\mathbf{x})<\eta_0(\mathbf{x})\}} \eta_1(\mathbf{x})p(\mathbf{x})\,d\mathbf{x} \;+ \int_{\{\mathbf{x}|\eta_1(\mathbf{x})\geq\eta_0(\mathbf{x})\}} \eta_0(\mathbf{x})p(\mathbf{x})\,d\mathbf{x} \\ &= E\left[\min\{\eta_0(\mathbf{X}),\eta_1(\mathbf{X})\}\right].\end{aligned} \tag{1.3}$$

By Jensen's inequality, it follows that $\varepsilon_{\text{bay}} \leq \min\left\{E[\eta_0(\mathbf{X})], E[\eta_1(\mathbf{X})]\right\}$. Therefore, if either posterior probability is uniformly small (for example, if one of the classes is much more likely *a priori* than the other), then the Bayes error is necessarily small.

1.2 POPULATION-BASED DISCRIMINANTS

Classifiers are often expressed in terms of a discriminant function, where the value of the classifier depends on whether or not the discriminant exceeds a threshold. Formally, a *discriminant* is a measurable function $D : R^d \to R$. A classifier ψ can be defined by

$$\psi(\mathbf{x}) = \begin{cases} 1\,, & D(\mathbf{x}) \leq k\,, \\ 0\,, & \text{otherwise,} \end{cases} \tag{1.4}$$

for a suitably chosen constant k. Such a classification procedure introduces three error rates: the population-specific error rates (assuming the existence of population densities)

$$\varepsilon^0 = P(D(\mathbf{X}) \leq k \mid Y = 0) = \int_{D(\mathbf{x})\leq k} p_0(\mathbf{x})\,d\mathbf{x}\,, \tag{1.5}$$

$$\varepsilon^1 = P(D(\mathbf{X}) > k \mid Y = 1) = \int_{D(\mathbf{x})>k} p_1(\mathbf{x})\,d\mathbf{x}\,, \tag{1.6}$$

and the overall error rate

$$\varepsilon = c_0\varepsilon^0 + c_1\varepsilon^1. \tag{1.7}$$

It is easy to show that this definition agrees with the previous definition of classification error.

It is well known (Anderson, 1984, Chapter 6) that a Bayes classifier is given by the likelihood ratio $p_0(\mathbf{x})/p_1(\mathbf{x})$ as the discriminant, with a threshold $k = c_1/c_0$. Since

log is a monotone increasing function, an equivalent optimal procedure is to use the log-likelihood ratio as the discriminant,

$$D_{\text{bay}}(\mathbf{x}) = \log \frac{p_0(\mathbf{x})}{p_1(\mathbf{x})}, \tag{1.8}$$

and define a Bayes classifier as

$$\psi_{\text{bay}}(\mathbf{x}) = \begin{cases} 1, & D_{\text{bay}}(\mathbf{x}) \leq \log \frac{c_1}{c_0}, \\ 0, & \text{otherwise.} \end{cases} \tag{1.9}$$

In the absence of knowledge about c_0 and c_1, a Bayes procedure cannot be completely determined, nor is the overall error rate known; however, one can still consider the error rates ε^0 and ε^1 individually. In this context, a classifier ψ_1 is *at least as good as* classifier ψ_2 if the error rates ε^0 and ε^1 corresponding to ψ_1 are smaller or equal to those corresponding to ψ_2; and ψ_1 is *better* than ψ_2 if ψ_1 is at least as good as ψ_2 and at least one of these errors is strictly smaller. A classifier is *an admissible procedure* if no other classifier is better than it.

Now assume that one uses the log-likelihood ratio as the discriminant and adjusts the threshold k to obtain different error rates ε^0 and ε^1. Increasing k increases ε^0 but decreases ε^1, while decreasing k decreases ε^0 but increases ε^1. This corresponds to a *family of Bayes procedures*, as each procedure in the family corresponds to a Bayes classifier for a choice of c_0 and c_1 that gives $\log c_1/c_0 = k$. Under minimal conditions on the population densities, it can be shown that every admissible procedure is a Bayes procedure and vice versa, so that the class of admissible procedures and the class of Bayes procedures are the same (Anderson, 1984).

It is possible to define a best classifier in terms of a criterion based only on the population error rates ε^0 and ε^1. One well-known example of such a criterion is the *minimax criterion*, which seeks to find the threshold k_{mm} and corresponding classifier

$$\psi_{\text{mm}}(\mathbf{x}) = \begin{cases} 1, & \log \frac{p_0(\mathbf{x})}{p_1(\mathbf{x})} \leq k_{\text{mm}} \\ 0, & \text{otherwise} \end{cases} \tag{1.10}$$

that minimize the maximum of ε^0 and ε^1. For convenience, we denote below the dependence of the error rates ε^0 and ε^1 on k explicitly. The minimax threshold can thus be written as

$$k_{\text{mm}} = \arg\min_k \max\{\varepsilon^0(k), \varepsilon^1(k)\}. \tag{1.11}$$

It follows easily from the fact that the family of admissible procedures coincides with the family of Bayes procedures that the minimax procedure is a Bayes procedure. Furthermore, we have the following result, which appears, in a slightly different form, in Esfahani and Dougherty (2014a).

Theorem 1.1 *(Esfahani and Dougherty, 2014a) If the class-conditional densities are strictly positive and $\varepsilon^0(k)$, $\varepsilon^1(k)$ are continuous functions of k, then*

(a) There exists a unique point k_{mm} such that

$$\varepsilon^0(k_{\text{mm}}) = \varepsilon^1(k_{\text{mm}}), \tag{1.12}$$

and this point corresponds to the minimax solution defined in (1.11).

(b) Let c_{mm} be the value of the prior probability c_0 that maximizes the Bayes error:

$$\begin{aligned} c_{\text{mm}} &= \arg\max_{c_0} \varepsilon_{\text{bay}} \\ &= \arg\max_{c_0} \left\{ c_0 \varepsilon^0 \left(\log \frac{1-c_0}{c_0} \right) + (1-c_0)\varepsilon^1 \left(\log \frac{1-c_0}{c_0} \right) \right\}. \end{aligned} \tag{1.13}$$

Then

$$k_{\text{mm}} = \log \left(\frac{1-c_{\text{mm}}}{c_{\text{mm}}} \right). \tag{1.14}$$

Proof: Part (a): Applying the monotone convergence theorem to (1.5) and (1.6) shows that $\varepsilon^0(k) \to 1$ and $\varepsilon^1(k) \to 0$ as $k \to \infty$ and $\varepsilon^0(k) \to 0$ and $\varepsilon^1(k) \to 1$ as $k \to -\infty$. Hence, since $\varepsilon^0(k)$ and $\varepsilon^1(k)$ are continuous functions of k, there is a point k_{mm} for which $\varepsilon^0(k_{\text{mm}}) = \varepsilon^1(k_{\text{mm}})$. Owing to $\varepsilon^0(k)$ and $\varepsilon^1(k)$ being strictly increasing and decreasing functions, respectively, this point is unique. We show that this point is the minimax solution via contradiction. Note that $\varepsilon^0(k_{\text{mm}}) = \varepsilon^1(k_{\text{mm}})$ implies that $\varepsilon_{\text{mm}} = c_0\varepsilon^0(k_{\text{mm}}) + c_1\varepsilon^1(k_{\text{mm}}) = \varepsilon^0(k_{\text{mm}}) = \varepsilon^1(k_{\text{mm}})$, regardless of c_0, c_1. Suppose there exists $k' \neq k_{\text{mm}}$ such that $\max\{\varepsilon^0(k'), \varepsilon^1(k')\} < \max\{\varepsilon^0(k_{\text{mm}}), \varepsilon^1(k_{\text{mm}})\} = \varepsilon_{\text{mm}}$. First, consider $k' > k_{\text{mm}}$. Since $\varepsilon^0(k)$ and $\varepsilon^1(k)$ are strictly increasing and decreasing functions, respectively, there are $\delta_0, \delta_1 > 0$ such that

$$\begin{cases} \varepsilon^0(k') = \varepsilon^0(k_{\text{mm}}) + \delta_0 = \varepsilon_{\text{mm}} + \delta_0, \\ \varepsilon^1(k') = \varepsilon^1(k_{\text{mm}}) - \delta_1 = \varepsilon_{\text{mm}} - \delta_1. \end{cases} \tag{1.15}$$

Therefore, $\max\{\varepsilon^0(k'), \varepsilon^1(k')\} = \varepsilon_{\text{mm}} + \delta_0 > \varepsilon_{\text{mm}}$, a contradiction. Similarly, if $k' < k_{\text{mm}}$, there are $\delta_0, \delta_1 > 0$ such that $\varepsilon^0(k') = \varepsilon_{\text{mm}} - \delta_0$ and $\varepsilon^1(k') = \varepsilon_{\text{mm}} + \delta_1$, so that $\max\{\varepsilon^0(k'), \varepsilon^1(k')\} = \varepsilon_{\text{mm}} + \delta_1 > \varepsilon_{\text{mm}}$, again a contradiction.

Part (b): Define

$$\varepsilon(c, c_0) \stackrel{\Delta}{=} \left\{ c_0 \varepsilon^0 \left(\log \frac{1-c}{c} \right) + (1-c_0)\varepsilon^1 \left(\log \frac{1-c}{c} \right) \right\}. \tag{1.16}$$

Let $k_{\text{mm}} = \log(1-c_{\text{mm}})/c_{\text{mm}}$, where at this point c_{mm} is just the value that makes the equality true (given k_{mm}). We need to show that c_{mm} is given by the expression in

part (b). We have that $\varepsilon_{\text{mm}} = \varepsilon(c_{\text{mm}}, c_0)$ is independent of c_0. Also, for a given value of c_0, the Bayes error is given by $\varepsilon_{\text{bay}} = \varepsilon(c_0, c_0)$. We need to show that $\varepsilon(c_{\text{mm}}, c_{\text{mm}}) = \max_{c_0} \varepsilon(c_0, c_0)$. The inequality $\varepsilon(c_{\text{mm}}, c_{\text{mm}}) \leq \max_{c_0} \varepsilon(c_0, c_0)$ is trivial. To establish the converse inequality, notice that $\varepsilon_{\text{mm}} = \varepsilon(c_{\text{mm}}, c_{\text{mm}}) \geq \varepsilon(c_0, c_0) = \varepsilon_{\text{bay}}$, for any value of c_0, by definition of the Bayes error, so that $\varepsilon(c_{\text{mm}}, c_{\text{mm}}) \geq \max_{c_0} \varepsilon(c_0, c_0)$. ■

The previous theorem has two immediate and important consequences: (1) The error $\varepsilon_{\text{mm}} = c_0\varepsilon^0(k_{\text{mm}}) + c_1\varepsilon^1(k_{\text{mm}})$ of the minimax classifier is such that $\varepsilon_{\text{mm}} = \varepsilon^0(k_{\text{mm}}) = \varepsilon^1(k_{\text{mm}})$, regardless of c_0, c_1. (2) The error of the minimax classifier is equal to the maximum Bayes error as c_0 varies: $\varepsilon_{\text{mm}} = \max_{c_0} \varepsilon_{\text{bay}}$.

Consider now the important special case of multivariate Gaussian populations $\Pi_0 \sim N_d(\boldsymbol{\mu}_0, \Sigma_0)$ and $\Pi_1 \sim N_d(\boldsymbol{\mu}_1, \Sigma_1)$ (see Section A.11 for basic facts about the multivariate Gaussian). The class-conditional densities take the form

$$p_i(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma_i)}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)\right], \quad i = 0, 1. \quad (1.17)$$

It follows from (1.8) that the optimal discriminant (using "D_Q" for quadratic discriminant) is given by

$$D_Q(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_0)^T \Sigma_0^{-1} (\mathbf{x} - \boldsymbol{\mu}_0) + \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^T \Sigma_1^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) + \log \frac{\det(\Sigma_1)}{\det(\Sigma_0)}. \quad (1.18)$$

This discriminant has the form $\mathbf{x}^T A \mathbf{x} - b^T \mathbf{x} + c = 0$, which describes a *hyperquadric* decision boundary in R^d (Duda and Hart, 2001). If $\Sigma_0 = \Sigma_1 = \Sigma$, then the optimal discriminant (using "D_L" for linear discriminant) reduces to

$$D_L(\mathbf{x}) = \left(\mathbf{x} - \frac{\boldsymbol{\mu}_0 + \boldsymbol{\mu}_1}{2}\right)^T \Sigma^{-1} \left(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\right). \quad (1.19)$$

This discriminant has the form $A\mathbf{x}^T + b = 0$, which describes a hyperplane in R^d. The optimal classifiers based on discriminants $D_Q(\mathbf{x})$ and $D_L(\mathbf{x})$ define population-based *Quadratic Discriminant Analysis* (QDA) and population-based *Linear Discriminant Analysis* (LDA), respectively.

Using the properties of Gaussian random vectors (c.f. Section A.11), it follows from (1.19) that $D_L(\mathbf{X}) \mid Y = 0 \sim N\left(\frac{1}{2}\delta^2, \delta^2\right)$, whereas $D_L(\mathbf{X}) \mid Y = 1 \sim N\left(-\frac{1}{2}\delta^2, \delta^2\right)$, where

$$\delta = \sqrt{(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \Sigma^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)} \quad (1.20)$$

is the *Mahalanobis distance* between the populations. It follows that

$$\varepsilon_L^0(k) = P(D_L(\mathbf{X}) \leq k \mid Y = 0) = \Phi\left(\frac{k - \frac{1}{2}\delta^2}{\delta}\right), \tag{1.21}$$

$$\varepsilon_L^1(k) = P(D_L(\mathbf{X}) > k \mid Y = 1) = \Phi\left(\frac{-k - \frac{1}{2}\delta^2}{\delta}\right), \tag{1.22}$$

where $\Phi(x)$ is the cumulative distribution function of a standard $N(0,1)$ Gaussian random variable. Regardless of the value of k, $\varepsilon_L^0(k)$ and $\varepsilon_L^1(k)$, and thus the overall error rate $\varepsilon_L(k) = c_0\varepsilon_L^0(k) + c_1\varepsilon_L^1(k)$, are decreasing functions of the Mahalanobis distance δ and $\varepsilon_L^0(k), \varepsilon_L^1(k), \varepsilon_L(k) \to 0$ as $\delta \to \infty$. The optimal classifier corresponds to the special case $k = k_{\text{bay}} = \log c_1/c_0$. Therefore, the Bayes error rates $\varepsilon_{L,\text{bay}}^0, \varepsilon_{L,\text{bay}}^1$, and $\varepsilon_{L,\text{bay}}$ are decreasing in δ, and converge to zero as $\delta \to \infty$. If the classes are equally likely, $c_0 = c_1 = 0.5$, then $k_{\text{bay}} = 0$, so that the Bayes error rates can be written simply as

$$\varepsilon_{L,\text{bay}}^0 = \varepsilon_{L,\text{bay}}^1 = \varepsilon_{L,\text{bay}} = \Phi\left(-\frac{\delta}{2}\right). \tag{1.23}$$

If the prior probabilities are not known, then, as seen in Theorem 1.1, the minimax rule is obtained by thresholding the discriminant $D_L(\mathbf{x})$ at a point k_{mm} that makes the two population-specific error rates ε_L^0 and ε_L^1 equal. From (1.21) and (1.22), we need to find the value k_{mm} for which

$$\Phi\left(\frac{k_{\text{mm}} - \frac{1}{2}\delta^2}{\delta}\right) = \Phi\left(\frac{-k_{\text{mm}} - \frac{1}{2}\delta^2}{\delta}\right). \tag{1.24}$$

Since Φ is monotone, the only solution to this equation is $k_{\text{mm}} = 0$. The minimax LDA classifier is thus given by

$$\psi_{L,\text{mm}}(\mathbf{x}) = \begin{cases} 1, & \left(\mathbf{x} - \frac{\boldsymbol{\mu}_0 + \boldsymbol{\mu}_1}{2}\right)^T \Sigma^{-1}\left(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\right) \leq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{1.25}$$

Clearly, the minimax LDA classifier is a Bayes LDA classifier when $c_0 = c_1$.

Under the general Gaussian model, the minimax threshold can differ substantially from 0. This is illustrated in Figure 1.1, which displays plots of the Bayes error as a function of the prior probability c_0 for different Gaussian models, including the case of distinct covariance matrices. Notice that, in accordance with Theorem 1.1, $k_{\text{mm}} = \log(1 - c_{\text{mm}})/c_{\text{mm}}$, where c_{mm} is the point where the plots reach the maximum. For the *homoskedastic* (equal-covariance) LDA model, $c_{\text{mm}} = 0.5$ and $k_{\text{mm}} = 0$, but k_{mm} becomes nonzero under the two *heteroskedastic* (unequal-covariance) models.

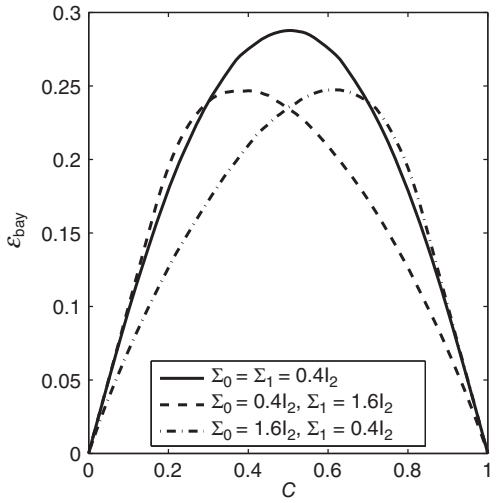


FIGURE 1.1 Bayes error as a function of c_0 under different Gaussian models, with $\boldsymbol{\mu}_0 = (0.3, 0.3)$ and $\boldsymbol{\mu}_1 = (0.8, 0.8)$ and indicated covariance matrices. Reproduced from Esfahani, S. and Dougherty (2014) with permission from Oxford University Press.

1.3 CLASSIFICATION RULES

In practice, the feature–label distribution is typically unknown and a classifier must be designed from sample data. An obvious approach would be to estimate the posterior probabilities from data, but often there is insufficient data to obtain good estimates. Fortunately, good classifiers can be obtained even when there is insufficient data for satisfactory density estimation.

We assume that we have a random sample $S_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ of feature vector–label pairs drawn from the feature–label distribution, meaning that the pairs $(\mathbf{X}_i, Y_i)$ are independent and identically distributed according to $F_{\mathbf{X}Y}$. The number of sample points from populations Π_0 and Π_1 are denoted by N_0 and N_1, respectively. Note that $N_0 = \sum_{i=1}^{n} I_{Y_i=0}$ and $N_1 = \sum_{i=1}^{n} I_{Y_i=1}$ are random variables such that $N_0 \sim$ Binomial(n, c_0) and $N_1 \sim$ Binomial(n, c_1), with $N_0 + N_1 = n$. A *classification rule* is a mapping $\Psi_n : S_n \mapsto \psi_n$, that is, given a sample S_n, a designed classifier $\psi_n = \Psi_n(S_n)$ is obtained according to the rule Ψ_n. Note that what we call a classification rule is really a sequence of classification rules depending on n. For simplicity, we shall not make the distinction formally.

We would like the error $\varepsilon_n = \varepsilon[\psi_n]$ of the designed classifier to be close to the the Bayes error. The *design error,*

$$\Delta_n = \varepsilon_n - \varepsilon_{\text{bay}}, \tag{1.26}$$

quantifies the degree to which a designed classifier fails to achieve this goal, where ε_n and Δ_n are sample-dependent random variables. The expected design error is $E[\Delta_n]$,

the expectation being relative to all possible samples. The expected error of ψ_n is decomposed according to

$$E[\varepsilon_n] = \varepsilon_{\text{bay}} + E[\Delta_n]. \tag{1.27}$$

The quantity $E[\varepsilon_n]$, or alternatively $E[\Delta_n]$, measures the performance of the classification rule relative to the feature–label distribution, rather than the performance of an individual designed classifier. Nonetheless, a classification rule for which $E[\varepsilon_n]$ is small will also tend to produce designed classifiers possessing small errors.

Asymptotic properties of a classification rule concern behavior as sample size grows to infinity. A rule is said to be *consistent* for a feature–label distribution of $(\mathbf{X}, Y)$ if $\varepsilon_n \to \varepsilon_{\text{bay}}$ in probability as $n \to \infty$. The rule is said to be *strongly consistent* if convergence is with probability one. A classification rule is said to be *universally consistent* (resp. *universally strongly consistent*) if the previous convergence modes hold for any distribution of $(\mathbf{X}, Y)$. Note that the random sequence $\{\varepsilon_n; n = 1, 2, \ldots\}$ is uniformly bounded in the interval $[0, 1]$, by definition; hence, $\varepsilon_n \to \varepsilon_{\text{bay}}$ in probability implies that $E[\varepsilon_n] \to \varepsilon_{\text{bay}}$ as $n \to \infty$ (c.f. Corollary A.1). Furthermore, it is easy to see that the converse implication also holds in this case, because $\varepsilon_n - \varepsilon_{\text{bay}} = |\varepsilon_n - \varepsilon_{\text{bay}}|$, and thus $E[\varepsilon_n] \to \varepsilon_{\text{bay}}$ implies $E[|\varepsilon_n - \varepsilon_{\text{bay}}|] \to 0$, and hence L_1-convergence. This in turn is equivalent to convergence in probability (c.f. Theorem A.1). Therefore, a rule is consistent if and only if $E[\Delta_n] = E[\varepsilon_n] - \varepsilon_{\text{bay}} \to 0$ as $n \to \infty$, that is, the expected design error becomes arbitrarily small for a sufficiently large sample. If this condition holds for any distribution of $(\mathbf{X}, Y)$, then the rule is universally consistent, according to the previous definition.

Universal consistency is a useful property for large samples, but it is virtually of no consequence for small samples. In particular, the design error $E[\Delta_n]$ cannot be made universally small for fixed n. For any $\tau > 0$, fixed n, and classification rule Ψ_n, there exists a distribution of $(\mathbf{X}, Y)$ such that $\varepsilon_{\text{bay}} = 0$ and $E[\varepsilon_n] > \frac{1}{2} - \tau$ (Devroye et al., 1996, Theorem 7.1). Put another way, for fixed n, absent constraint on the feature–label distribution, a classification rule may have arbitrarily bad performance.

Furthermore, in addition to consistency of a classification rule, the rate at which $E[\Delta_n] \to 0$ is also critical. In other words, we would like a statement of the following form: for any $\tau > 0$, there exists $n(\tau)$ such that $E[\Delta_{n(\tau)}] < \tau$ for any distribution of $(\mathbf{X}, Y)$. Such a proposition is not possible; even if a classification rule is universally consistent, the design error converges to 0 arbitrarily slowly relative to all possible distributions. Indeed, it can be shown (Devroye et al., 1996, Theorem 7.2) that if $\{a_n; n = 1, 2, \ldots\}$ is a decreasing sequence of positive numbers with $a_1 \leq \frac{1}{16}$, then for any classification rule Ψ_n, there exists a distribution of $(\mathbf{X}, Y)$ such that $\varepsilon_{\text{bay}} = 0$ and $E[\varepsilon_n] > a_n$. The situation changes if distributional assumptions are made — a scenario that will be considered extensively in this book.

A classification rule can yield a classifier that makes very few errors, or none, on the sample data from which it is designed, but performs poorly on the distribution as a whole, and therefore on new data to which it is applied. This situation, typically referred to as *overfitting*, is exacerbated by complex classifiers and small samples.

The basic point is that a classification rule should not cut up the space in a manner too complex for the amount of sample data available. This might improve the *apparent error rate* (i.e., the number of errors committed by the classifier using the training data as testing points), also called the *resubstitution error rate* (see the next chapter), but at the same time it will most likely worsen the true error of the classifier on independent future data (also called the *generalization error* in this context). The problem is not necessarily mitigated by applying an error-estimation rule — perhaps more sophisticated than the apparent error rate — to the designed classifier to see if it "actually" performs well, since when there is only a small amount of data available, error-estimation rules are usually inaccurate, and the inaccuracy tends to be worse for complex classification rules. Hence, a low error estimate is not sufficient to overcome our expectation of a large expected error when using a complex classifier with a small data set. These issues can be studied with the help of Vapnik–Chervonenkis theory, which is reviewed in Appendix B. In the remainder of this section, we apply several of the results in Appendix B to the problem of efficient classifier design.

Constraining classifier design means restricting the functions from which a classifier can be chosen to a class $\mathcal{C}$. This leads to trying to find an optimal *constrained classifier* $\psi_{\mathcal{C}} \in \mathcal{C}$ having error $\varepsilon_{\mathcal{C}}$. Constraining the classifier can reduce the expected design error, but at the cost of increasing the error of the best possible classifier. Since optimization in $\mathcal{C}$ is over a subclass of classifiers, the error $\varepsilon_{\mathcal{C}}$ of $\psi_{\mathcal{C}}$ will typically exceed the Bayes error, unless a Bayes classifier happens to be in $\mathcal{C}$. This *cost of constraint* is

$$\Delta_{\mathcal{C}} = \varepsilon_{\mathcal{C}} - \varepsilon_{\text{bay}}. \tag{1.28}$$

A classification rule yields a classifier $\psi_{n,\mathcal{C}} \in \mathcal{C}$, with error $\varepsilon_{n,\mathcal{C}}$, and $\varepsilon_{n,\mathcal{C}} \geq \varepsilon_{\mathcal{C}} \geq \varepsilon_{\text{bay}}$. The *design error* for constrained classification is

$$\Delta_{n,\mathcal{C}} = \varepsilon_{n,\mathcal{C}} - \varepsilon_{\mathcal{C}}. \tag{1.29}$$

For small samples, this can be substantially less than Δ_n, depending on $\mathcal{C}$ and the classification rule. The error of a designed constrained classifier is decomposed as

$$\varepsilon_{n,\mathcal{C}} = \varepsilon_{\text{bay}} + \Delta_{\mathcal{C}} + \Delta_{n,\mathcal{C}}. \tag{1.30}$$

Therefore, the expected error of the designed classifier from $\mathcal{C}$ can be decomposed as

$$E[\varepsilon_{n,\mathcal{C}}] = \varepsilon_{\text{bay}} + \Delta_{\mathcal{C}} + E[\Delta_{n,\mathcal{C}}]. \tag{1.31}$$

The constraint is beneficial if and only if $E[\varepsilon_{n,\mathcal{C}}] < E[\varepsilon_n]$, that is, if

$$\Delta_{\mathcal{C}} < E[\Delta_n] - E[\Delta_{n,\mathcal{C}}]. \tag{1.32}$$

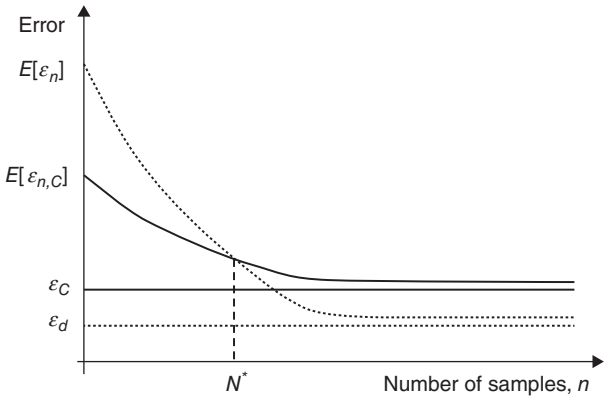


FIGURE 1.2 "Scissors Plot." Errors for constrained and unconstrained classification as a function of sample size.

If the cost of constraint is less than the decrease in expected design error, then the expected error of $\psi_{n,C}$ is less than that of ψ_n. The dilemma: strong constraint reduces $E[\Delta_{n,C}]$ at the cost of increasing ε_C, and vice versa. Figure 1.2 illustrates the design problem; this is known in pattern recognition as a "scissors plot" (Kanal and Chandrasekaran, 1971; Raudys, 1998). If n is sufficiently large, then $E[\varepsilon_n] < E[\varepsilon_{n,C}]$; however, if n is sufficiently small, then $E[\varepsilon_n] > E[\varepsilon_{n,C}]$. The point N^* at which the decreasing lines cross is the cut-off: for $n > N^*$, the constraint is detrimental; for $n < N^*$, it is beneficial. When $n < N^*$, the advantage of the constraint is the difference between the decreasing solid and dashed lines.

Clearly, how much constraint is applied to a classification rule is related to the size of $\mathcal{C}$. The *Vapnik–Chervonenkis dimension* $V_{\mathcal{C}}$ yields a quantitative measure of the size of $\mathcal{C}$. A larger VC dimension corresponds to a larger $\mathcal{C}$ and a greater potential for overfitting, and vice versa. The VC dimension is defined in terms of the *shatter coefficients* $\mathcal{S}(\mathcal{C}, n)$, $n = 1, 2, \ldots$ The reader is referred to Appendix B for the formal definitions of VC dimension and shatter coefficients of a class $\mathcal{C}$.

The Vapnik–Chervonenkis theorem (c.f. Theorem B.1) states that, regardless of the distribution of $(\mathbf{X}, Y)$, for all sample sizes and for all $\tau > 0$,

$$P\left(\sup_{\psi \in \mathcal{C}} |\hat{\varepsilon}[\psi] - \varepsilon[\psi]| > \tau\right) \leq 8\mathcal{S}(\mathcal{C}, n)e^{-n\tau^2/32}, \tag{1.33}$$

where $\varepsilon[\psi]$ and $\hat{\varepsilon}[\psi]$ are respectively the true and apparent error rates of a classifier $\psi \in \mathcal{C}$. Since this holds for all $\psi \in \mathcal{C}$, it holds in particular for the designed classifier $\psi = \psi_{n,C}$, in which case $\varepsilon[\psi] = \varepsilon_{n,C}$, the designed classifier error, and $\hat{\varepsilon}[\psi] = \hat{\varepsilon}_{n,C}$, the apparent error of the designed classifier. Thus, we may drop the sup to obtain

$$P\left(|\hat{\varepsilon}_{n,C} - \varepsilon_{n,C}| > \tau\right) \leq 8\mathcal{S}(\mathcal{C}, n)e^{-n\tau^2/32}, \tag{1.34}$$

for all $\tau > 0$. Now, suppose that the classifier $\psi_{n,\mathcal{C}}$ is designed by choosing the classifier in $\mathcal{C}$ that minimizes the apparent error. This is called the *empirical risk minimization* (ERM) principle (Vapnik, 1998). Many classification rules of interest satisfy this property, such as histogram rules, perceptrons, and certain neural network and classification tree rules. In this case, we can obtain the following inequality for the design error $\Delta_{n,\mathcal{C}}$:

$$\begin{aligned}\Delta_{n,\mathcal{C}} &= \varepsilon_{n,\mathcal{C}} - \varepsilon_{\mathcal{C}} = \varepsilon_{n,\mathcal{C}} - \hat{\varepsilon}_{n,\mathcal{C}} + \hat{\varepsilon}_{n,\mathcal{C}} - \inf_{\psi\in\mathcal{C}} \varepsilon[\psi] \\ &= \varepsilon[\psi_{n,\mathcal{C}}] - \hat{\varepsilon}[\psi_{n,\mathcal{C}}] + \inf_{\psi\in\mathcal{C}} \hat{\varepsilon}[\psi] - \inf_{\psi\in\mathcal{C}} \varepsilon[\psi] \\ &\leq \varepsilon[\psi_{n,\mathcal{C}}] - \hat{\varepsilon}[\psi_{n,\mathcal{C}}] + \sup_{\psi\in\mathcal{C}} |\hat{\varepsilon}[\psi] - \varepsilon[\psi]| \\ &\leq 2 \sup_{\psi\in\mathcal{C}} |\hat{\varepsilon}[\psi] - \varepsilon[\psi]|\,,\end{aligned} \tag{1.35}$$

where we used $\hat{\varepsilon}_{n,\mathcal{C}} = \inf_{\psi\in\mathcal{C}} \hat{\varepsilon}[\psi]$ (because of ERM) and the fact that $\inf f(x) - \inf g(x) \leq \sup(f(x) - g(x)) \leq \sup |f(x) - g(x)|$. Therefore,

$$P(\Delta_{n,\mathcal{C}} > \tau) \leq P\left(\sup_{\psi\in\mathcal{C}} |\hat{\varepsilon}[\psi] - \varepsilon[\psi]| > \frac{\tau}{2}\right). \tag{1.36}$$

We can now apply (1.34) to get

$$P(\Delta_{n,\mathcal{C}} > \tau) \leq 8\mathcal{S}(\mathcal{C}, n) e^{-n\tau^2/128}, \tag{1.37}$$

for all $\tau > 0$. We can also apply Lemma A.3 to (1.37) to obtain a bound on the expected design error:

$$E[\Delta_{n,\mathcal{C}}] \leq 16\sqrt{\frac{\log \mathcal{S}(\mathcal{C}, n) + 4}{2n}}. \tag{1.38}$$

If $V_{\mathcal{C}} < \infty$, then we can use the inequality $\mathcal{S}(\mathcal{C}, n) \leq (n+1)^{V_{\mathcal{C}}}$ to replace $\mathcal{S}(\mathcal{C}, n)$ by $(n+1)^{V_{\mathcal{C}}}$ in the previous equations. For example, from (1.34),

$$P(\Delta_{n,\mathcal{C}} > \tau) \leq 8(n+1)^{V_{\mathcal{C}}} e^{-n\tau^2/128}, \tag{1.39}$$

for all $\tau > 0$. Note that in this case $\sum_{n=1}^{\infty} P(\Delta_{n,\mathcal{C}} > \tau) < \infty$, for all $\tau > 0$, and all distributions, so that, by the First Borel–Cantelli lemma (c.f. Lemma A.1), $\Delta_{n,\mathcal{C}} \to 0$ with probability 1, regardless of the distribution. In addition, (1.38) yields

$$E[\Delta_{n,\mathcal{C}}] \leq 16\sqrt{\frac{V_{\mathcal{C}} \log(n+1) + 4}{2n}}. \tag{1.40}$$

Hence, $E[\Delta_{n,C}] = O(\sqrt{V_C \log n/n})$, and one must have $n \gg V_C$ for the bound to be small. In other words, sample size has to grow faster than the complexity of the classification rule in order to guarantee good performance. One may think that by not using the ERM principle in classifier design, one is not impacted by the VC dimension of the classification rule. In fact, independently of how to pick ψ_n from C, both empirical and analytical evidence suggest that this is not true. For example, it can be shown that, under quite general conditions, there is a distribution of the data that demands one to have $n \gg V_C$ for acceptable performance, regardless of the classification rule (Devroye et al., 1996, Theorem. 14.5)

A note of caution: all bounds in VC theory are worst-case, as there are no distributional assumptions, and thus they can be very loose for a particular feature–label distribution and in small-sample cases. Nonetheless, in the absence of distributional knowledge, the theory prescribes that increasing complexity is counterproductive unless there is a large sample available. Otherwise, one could easily end up with a very bad classifier whose error estimate is very small.

1.4 SAMPLE-BASED DISCRIMINANTS

A *sample-based discriminant* is a real function $W(S_n, \mathbf{x})$, where S_n is a given sample and $\mathbf{x} \in R^d$. According to this definition, in fact we have a sequence of discriminants indexed by n. For simplicity, we shall not make this distinction formally, nor shall we add a subscript "n" to the discriminant; what is intended will become clear by the context. Many useful classification rules are defined via sample-based discriminants. Formally, a sample-based discriminant W defines a designed classifier (and a classification rule, implicitly) via

$$\Psi_n(S_n)(\mathbf{x}) = \begin{cases} 1, & W(S_n, \mathbf{x}) \leq k, \\ 0, & \text{otherwise}, \end{cases} \tag{1.41}$$

where S_n is the sample and k is a threshold parameter, the choice of which can be made in different ways.

A *plug-in discriminant* is a discriminant obtained from the population discriminant of (1.8) by plugging in sample estimates. If the discriminant is a plug-in discriminant and the prior probabilities are known, then one should follow the optimal classification procedure (1.9) and set $k = \log c_1/c_0$. On the other hand, if the prior probabilities are not known, one can set $k = \log N_1/N_0$, which converges to $\log c_1/c_0$ as $n \to \infty$ with probability 1. However, with small sample sizes, this choice can be problematic, as it may introduce significant variance in classifier design. An alternative in these cases is to use a fixed parameter, perhaps the minimax choice based on a model assumption (see Section 1.2) — this choice is risky, as the model assumption may be incorrect (Esfahani and Dougherty, 2014a). In any case, the use of a fixed threshold parameter is fine provided that it is reasonably close to $\log c_1/c_0$.

1.4.1 Quadratic Discriminants

Recall the population-based QDA discriminant in (1.18). If part or all of the population parameters are not known, sample estimators can be plugged in to yield the sample-based QDA discriminant

$$\begin{aligned} W_Q(S_n, \mathbf{x}) \;=\; & -\frac{1}{2}(\mathbf{x} - \bar{\mathbf{X}}_0)^T \hat{\Sigma}_0^{-1}(\mathbf{x} - \bar{\mathbf{X}}_0) \\ & + \frac{1}{2}(\mathbf{x} - \bar{\mathbf{X}}_1)^T \hat{\Sigma}_1^{-1}(\mathbf{x} - \bar{\mathbf{X}}_1) + \log \frac{|\hat{\Sigma}_1|}{|\hat{\Sigma}_0|}\,, \end{aligned} \tag{1.42}$$

where

$$\bar{\mathbf{X}}_0 \;=\; \frac{1}{N_0} \sum_{i=1}^{n} \mathbf{X}_i I_{Y_i=0} \quad \text{and} \quad \bar{\mathbf{X}}_1 \;=\; \frac{1}{N_1} \sum_{i=1}^{n} \mathbf{X}_i I_{Y_i=1} \tag{1.43}$$

are the sample means for each population, and

$$\hat{\Sigma}_0 \;=\; \frac{1}{N_0 - 1} \sum_{i=1}^{n} (\mathbf{X}_i - \bar{\mathbf{X}}_0)(\mathbf{X}_i - \bar{\mathbf{X}}_0)^T I_{Y_i=0}\,, \tag{1.44}$$

$$\hat{\Sigma}_1 \;=\; \frac{1}{N_1 - 1} \sum_{i=1}^{n} (\mathbf{X}_i - \bar{\mathbf{X}}_1)(\mathbf{X}_i - \bar{\mathbf{X}}_1)^T I_{Y_i=1}\,, \tag{1.45}$$

are the sample covariance matrices for each population.

The QDA discriminant is a special case of the general *quadratic discriminant*, which takes the form

$$W_{\text{quad}}(S_n, \mathbf{x}) \;=\; \mathbf{x}^T A(S_n)\mathbf{x} + \mathbf{b}(S_n)^T \mathbf{x} + c(S_n)\,, \tag{1.46}$$

where the sample-dependent parameters are a $d \times d$ matrix $A(S_n)$, a d-dimensional vector $\mathbf{b}(S_n)$, and a scalar $c(S_n)$. The QDA discriminant results from the choices

$$\begin{aligned} A(S_n) &= -\frac{1}{2}\left(\hat{\Sigma}_1^{-1} - \hat{\Sigma}_0^{-1}\right), \\ \mathbf{b}(S_n) &= \hat{\Sigma}_1^{-1}\bar{\mathbf{X}}_1 - \hat{\Sigma}_0^{-1}\bar{\mathbf{X}}_0, \\ c(S_n) &= -\frac{1}{2}\left(\bar{\mathbf{X}}_1{}^T \hat{\Sigma}_1^{-1}\bar{\mathbf{X}}_1 - \bar{\mathbf{X}}_0{}^T \hat{\Sigma}_0^{-1}\bar{\mathbf{X}}_0\right) - \frac{1}{2}\log\frac{|\hat{\Sigma}_1|}{|\hat{\Sigma}_0|}\,. \end{aligned} \tag{1.47}$$

1.4.2 Linear Discriminants

Recall now the population-based LDA discriminant in (1.19). Replacing the population parameters by sample estimators yields the sample-based LDA discriminant

$$W_L(S_n, \mathbf{x}) = \left(\mathbf{x} - \frac{\bar{\mathbf{X}}_0 + \bar{\mathbf{X}}_1}{2}\right)^T \hat{\Sigma}^{-1} \left(\bar{\mathbf{X}}_0 - \bar{\mathbf{X}}_1\right), \tag{1.48}$$

where $\bar{\mathbf{X}}_0$ and $\bar{\mathbf{X}}_1$ are the sample means and

$$\hat{\Sigma} = \frac{1}{n-2} \sum_{i=1}^{n} \left[(\mathbf{X}_i - \bar{\mathbf{X}}_0)(\mathbf{X}_i - \bar{\mathbf{X}}_0)^T I_{Y_i=0} + (\mathbf{X}_i - \bar{\mathbf{X}}_1)(\mathbf{X}_i - \bar{\mathbf{X}}_1)^T I_{Y_1=1} \right] \tag{1.49}$$

is the *pooled sample covariance matrix*. This discriminant is also known as *Anderson's LDA discriminant* (Anderson, 1984).

If Σ is assumed to be known, it can be used in placed of $\hat{\Sigma}$ in (1.48) to obtain the discriminant

$$W_{\bar{L}}(S_n, \mathbf{x}) = \left(\mathbf{x} - \frac{\bar{\mathbf{X}}_0 + \bar{\mathbf{X}}_1}{2}\right)^T \Sigma^{-1} \left(\bar{\mathbf{X}}_0 - \bar{\mathbf{X}}_1\right). \tag{1.50}$$

This discriminant is also known as *John's LDA discriminant* (John, 1961; Moran, 1975). Anderson's and John's discriminants are studied in detail in Chapters 6 and 7.

If the common covariance matrix can be assumed to be diagonal, then only its diagonal elements, that is, the univariate feature variances, need to be estimated. This leads to the *diagonal LDA discriminant* (DLDA), an application of the "Naive Bayes" principle (Dudoit et al., 2002).

Finally, a simple form of LDA discriminant arises from assuming that $\Sigma = \sigma^2 I$, a scalar multiple of the identity matrix, in which case one obtains the *nearest-mean classifier* (NMC) discriminant (Duda and Hart, 2001).

All variants of LDA discriminants discussed above are special cases of the general *linear discriminant*, which takes the form

$$W_{\text{lin}}(S_n, \mathbf{x}) = \mathbf{a}(S_n)^T \mathbf{x} + b(S_n), \tag{1.51}$$

where the sample-dependent parameters are a d-dimensional vector $\mathbf{a}(S_n)$ and a scalar $b(S_n)$. The basic LDA discriminant results from the choices

$$\begin{aligned} \mathbf{a}(S_n) &= \hat{\Sigma}^{-1}(\bar{\mathbf{X}}_0 - \bar{\mathbf{X}}_1), \\ b(S_n) &= -\frac{1}{2}(\bar{\mathbf{X}}_0 + \bar{\mathbf{X}}_1)^T \hat{\Sigma}^{-1}(\bar{\mathbf{X}}_0 - \bar{\mathbf{X}}_1). \end{aligned} \tag{1.52}$$

Besides LDA, other examples of linear discriminants include perceptrons (Duda and Hart, 2001) and linear SVMs (c.f. Section 1.6.2). For any linear discriminant, the decision surface $W(\mathbf{x}) = k$ defines a hyperplane in the feature space R^d.

The QDA and LDA classification rules are derived from a Gaussian assumption; nevertheless, in practice they can perform well so long as the underlying populations are approximately unimodal, and one either knows the relevant covariance matrices or can obtain good estimates of them. In large-sample scenarios, QDA is preferable; however, LDA has been reported to be more robust relative to the underlying Gaussian assumption than QDA (Wald and Kronmal, 1977). Moreover, even when the covariance matrices are not identical, LDA can outperform QDA under small sample sizes owing to the difficulty of estimating the individual covariance matrices.

1.4.3 Kernel Discriminants

Kernel discriminants take the form

$$W_{\text{ker}}(S_n, \mathbf{x}) = \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{X}_i}{h_n}\right)(I_{Y_i=0} - I_{Y_i=1}), \tag{1.53}$$

where the *kernel* $K(\mathbf{x})$ is a real function on R^d, which is usually (but not necessarily) nonnegative, and $h_n = H_n(S_n) > 0$ is the sample-based kernel bandwidth, defined by a rule H_n. From (1.53), we can see that the class label of a given point $\mathbf{x}$ is determined by the sum of the "contributions" $K((\mathbf{x} - \mathbf{X}_i)/h_n)$ of each training sample point $\mathbf{X}_i$, with positive sign if $\mathbf{X}_i$ is from class 0 and negative sign, otherwise. According to (1.41), with $k = 0$, $\psi_n(\mathbf{x})$ is equal to 0 or 1 according to whether the sum is positive or negative, respectively. The *Gaussian kernel* is defined by $K(\mathbf{x}) = e^{-\|\mathbf{x}\|^2}$, the *Epanechnikov kernel* is given by $K(\mathbf{x}) = (1 - ||\mathbf{x}||^2)I_{||\mathbf{x}||\leq 1}$, and the *uniform kernel* is given by $I_{||\mathbf{x}||\leq 1}$. Since the Gaussian kernel is never 0, all sample points get some weight. The Epanechnikov and uniform kernels possess bounded support, being 0 for sample points at a distance more than $h(S_n)$ from $\mathbf{x}$, so that only training points sufficiently close to $\mathbf{x}$ contribute to the definition of $\psi_n(\mathbf{x})$. The uniform kernel produces the *moving-window rule*, which takes the majority label among all sample points within distance $h(S_n)$ of $\mathbf{x}$. The decision surface $W_{\text{ker}}(S_n, \mathbf{x}) = k$ generally defines a complicated nonlinear manifold in R^d. The classification rules corresponding to the Gaussian, Epanechnikov, and uniform kernels are universally consistent (Devroye et al., 1996). Kernel discriminant rules also include as special cases Parzen-window classification rules (Duda and Hart, 2001) and nonlinear SVMs (c.f. Section 1.6.2).

1.5 HISTOGRAM RULE

Suppose that R^d is partitioned into cubes of equal side length r_n. For each point $\mathbf{x} \in R^d$, the *histogram rule* defines $\psi_n(\mathbf{x})$ to be 0 or 1 according to which is the

majority among the labels for points in the cube containing $\mathbf{x}$. If the cubes are defined so that $r_n \to 0$ and $nr_n^d \to \infty$ as $n \to \infty$, then the rule is universally consistent (Gordon and Olshen, 1978).

An important special case of the histogram rule occurs when the feature space is finite. This discrete histogram rule is also referred to as *multinomial discrimination*. In this case, we can establish a bijection between all possible values taken on by $\mathbf{X}$ and a finite set of integers, so that there is no loss in generality in assuming a single feature X taking values in the set $\{1, \ldots, b\}$. The value b provides a direct measure of the complexity of discrete classification — in fact, the VC dimension of the discrete histogram rule can be shown to be $V_C = b$ (c.f. Section B.3). The properties of the discrete classification problem are completely determined by the a priori class probabilities $c_0 = P(Y = 0)$, $c_1 = P(Y = 1)$, and the class-conditional probability mass functions $p_i = P(X = i|Y = 0)$, $q_i = P(X = i|Y = 1)$, for $i = 1, \ldots, b$. Since $c_1 = 1 - c_0$, $p_b = 1 - \sum_{i=1}^{b-1} p_i$, and $q_b = 1 - \sum_{i=1}^{b-1} q_i$, the classification problem is determined by a $(2b-1)$-dimensional vector $(c_0, p_1, \ldots, p_{b-1}, q_1, \ldots, q_{b-1}) \in R^{2b-1}$.

For a given a probability model, the error of a discrete classifier $\psi : \{1, \ldots, b\} \to \{0, 1\}$ is given by

$$\begin{aligned} \varepsilon[\psi] &= \sum_{i=1}^{b} P(X = i,\ Y = 1 - \psi(i)) \\ &= \sum_{i=1}^{b} P(X = i|Y = 1 - \psi(i))P(Y = 1 - \psi(i)) \\ &= \sum_{i=1}^{b} \left[p_i c_0 I_{\psi(i)=1} + q_i c_1 I_{\psi(i)=0} \right] . \end{aligned} \tag{1.54}$$

From this, it is clear that a Bayes classifier is given by

$$\psi_{\text{bay}}(i) = \begin{cases} 1\,, & p_i c_0 < q_i c_1\,, \\ 0\,, & \text{otherwise,} \end{cases} \tag{1.55}$$

for $i = 1, \ldots, b$, with Bayes error

$$\varepsilon_{\text{bay}} = \sum_{i=1}^{b} \min\{c_0 p_i, c_1 q_i\}\,. \tag{1.56}$$

Given a sample $S_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, the *bin counts* U_i and V_i are defined as the observed number of points with $X = i$ for class 0 and 1, respectively, for $i = 1, \ldots, b$. The *discrete histogram classification rule* is given by majority voting between the bin counts over each bin:

$$\Psi_n(S_n)(i) = I_{U_i < V_i} = \begin{cases} 1\,, & U_i < V_i \\ 0\,, & \text{otherwise,} \end{cases} \tag{1.57}$$

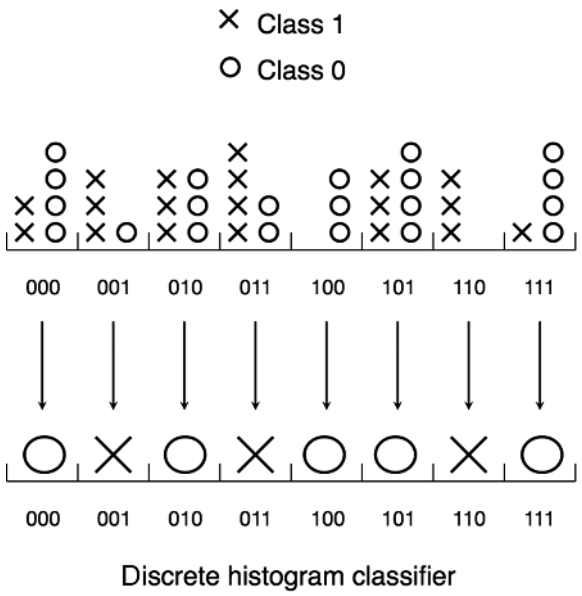


FIGURE 1.3 Discrete histogram rule: top shows distribution of the sample data in the bins; bottom shows the designed classifier.

for $i = 1, \ldots, b$. The rule is illustrated in Figure 1.3 where the top and bottom rows represent the sample data and the designed classifier, respectively. Note that in (1.57) the classifier is defined so that ties go to class 0, as happens in the figure.

The maximum-likelihood estimators of the distributional parameters are

$$\widehat{c}_0 = \frac{N_0}{n}, \ \widehat{c}_1 = \frac{N_1}{n}, \quad \text{and} \quad \widehat{p}_i = \frac{U_i}{N_0}, \ \widehat{q}_i = \frac{V_i}{N_1}, \ \text{for } i = 1, \ldots, b. \tag{1.58}$$

Plugging these into the expression of a Bayes classifier yields a histogram classifier, so that the histogram rule is the plug-in rule for discrete classification. For this reason, the fundamental rule is also called the *discrete-data plug-in rule.*

The classification error of the discrete histogram rule is given by

$$\varepsilon_n = \sum_{i=1}^{b} P(X = i, \, Y = 1 - \psi_n(i)) = \sum_{i=1}^{b} \left[c_0 p_i \, I_{V_i > U_i} + c_1 q_i \, I_{U_i \geq V_i} \right]. \tag{1.59}$$

Taking expectation leads to a remarkably simple expression for the expected classification error rate:

$$E[\varepsilon_n] = \sum_{i=1}^{b} \left[c_0 p_i \, P(V_i > U_i) + c_1 q_i \, P(U_i \geq V_i) \right]. \tag{1.60}$$

The probabilities in the previous equation can be easily computed by using the fact that the vector of bin counts $(U_1, \ldots, U_b, V_1, \ldots, V_b)$ is distributed multinomially with parameters $(n, c_0 p_1, \ldots, c_0 p_b, c_1 q_1, \ldots, c_1 q_b)$:

$$P(U_1 = u_1, \ldots, U_b = u_b, V_1 = v_1, \ldots, V_b = v_b) = \frac{n!}{u_1! \cdots u_b!\, v_1! \cdots v_b!}\, c_0^{\sum u_i} c_1^{\sum v_i} \prod_{i=1}^{b} p_i^{u_i} q_i^{v_i}\,. \tag{1.61}$$

The marginal distribution of the vector (U_i, V_i) is multinomial with parameters $(n, c_0 p_i, c_1 q_i, 1 - c_0 p_i - c_1 q_i)$, so that

$$\begin{aligned} P(V_i > U_i) &= \sum_{\substack{0 \le k+l \le n \\ k<l}} P(U_i = k, V_i = l) \\ &= \sum_{\substack{0 \le k+l \le n \\ k<l}} \frac{n!}{k!\, l!\, (n-k-l)!}\, (c_0 p_i)^k (c_1 q_i)^l (1 - c_0 p_i - c_1 q_i)^{n-k-l}\,, \end{aligned} \tag{1.62}$$

with $P(U_i \ge V_i) = 1 - P(V_i > U_i)$.

Using the Law of Large Numbers, it is easy to show that $\varepsilon_n \to \varepsilon_{\text{bay}}$ as $n \to \infty$ (for fixed b), with probability 1, for any distribution, that is, the discrete histogram rule is universally strongly consistent. As was mentioned in Section 1.3, this implies that $E[\varepsilon_n] \to \varepsilon_{\text{bay}}$.

As the number of bins b becomes comparable to the sample size n, the performance of the classifier degrades due to the increased likelihood of empty bins. Such bins have to be assigned an arbitrary label (here, the label zero), which clearly is suboptimal. This fact can be appreciated by examining (1.60), from which it follows that the expected error is bounded below in terms of the probabilities of empty bins:

$$E[\varepsilon_n] \ge \sum_{i=1}^{b} c_1 q_i\, P(U_i = 0, V_i = 0) = \sum_{i=1}^{b} c_1 q_i\, (1 - c_0 p_i - c_1 q_i)^n\,. \tag{1.63}$$

To appreciate the impact of the lower bound in (1.63), consider the following distribution, for even b: $c_0 = c_1 = \frac{1}{2}$, $q_i = \frac{2}{b}$, for $i = 1, \ldots, \frac{b}{2}$, $p_i = \frac{2}{b}$, for $i = \frac{b}{2} + 1, \ldots, b$, while q_i and p_i are zero, otherwise. In this case, it follows from (1.56) that $\varepsilon_{\text{bay}} = 0$. Despite an optimal error of zero, (1.63) yields the lower bound $E[\varepsilon_n] \ge \frac{1}{2}(1 - \frac{1}{b})^n$. With $b = n$ and $n \ge 4$, this bound gives $E[\varepsilon_n] > 0.15$, despite the classes being completely separated. This highlights the issue of small sample size n compared to the complexity b. Having n comparable to b results in a substantial increase in the expected error. But since $V_C = b$ for the discrete histogram rule, this simply reflects the need to have $n \gg V_C$ for acceptable performance, as argued in Section 1.3. While

it is true that the previous discussion is based on a distribution-dependent bound, it is shown in Devroye et al. (1996) that, regardless of the distribution,

$$E[\varepsilon_n] \leq \varepsilon_{\text{bay}} + 1.075\sqrt{\frac{b}{n}}. \tag{1.64}$$

This confirms that n must greatly exceed b to guarantee good performance.

Another issue concerns the rate of convergence, that is, how fast the bounds converge to zero. This is not only a theoretical question, as the usefulness in practice of such results may depend on how large the sample size needs to be to guarantee that performance is close enough to optimality. The rate of convergence guaranteed by (1.64) is polynomial. It is shown next that exponential rates of convergence can be obtained, if one is willing to drop the distribution-free requirement. Provided that ties in bin counts are assigned a class randomly (with equal probability), the following exponential bound on the convergence of $E[\varepsilon_n]$ to ε_{bay} applies (Glick, 1973, Theorem A):

$$E[\varepsilon_n] - \varepsilon_{\text{bay}} \leq \left(\frac{1}{2} - \varepsilon_{\text{bay}}\right) e^{-cn}, \tag{1.65}$$

where the constant $c > 0$ is distribution-dependent:

$$c = \log \frac{1}{1 - \min_{\{i: c_0 p_i \neq c_1 q_i\}} |\sqrt{c_0 p_i} - \sqrt{c_1 q_i}|^2}. \tag{1.66}$$

Interestingly, the number of bins does not figure in this bound. The speed of convergence of the bound is determined by the minimum (nonzero) difference between the probabilities $c_0 p_i$ and $c_1 q_i$ over any one cell. The larger this difference is, the larger c is, and the faster convergence is. Conversely, the presence of a single cell where these probabilities are close slows down convergence of the bound.

1.6 OTHER CLASSIFICATION RULES

This section briefly describes other commonly employed classification rules that will be used in the text to illustrate the behavior of error estimators.

1.6.1 *k*-Nearest-Neighbor Rules

For the basic *nearest-neighbor* (NN) classification rule, ψ_n is defined for each $\mathbf{x} \in R^d$ by letting $\psi_n(\mathbf{x})$ take the label of the sample point closest to $\mathbf{x}$. For the NN classification rule, no matter the feature–label distribution of $(\mathbf{X}, Y)$, $\varepsilon_{\text{bay}} \leq \lim_{n\to\infty} E[\varepsilon_n] \leq 2\,\varepsilon_{\text{bay}}$ (Cover and Hart, 1967). It follows that $\lim_{n\to\infty} E[\Delta_n] \leq \varepsilon_{\text{bay}}$. Hence, the asymptotic expected design error is small if the Bayes error is small; however, this result does

not give consistency. More generally, for the *k-nearest-neighbor* (*k*NN) classification rule, with k odd, the k points closest to $\mathbf{x}$ are selected and $\psi_n(\mathbf{x})$ is defined to be 0 or 1 according to which the majority is among the labels of these points. The 1NN rule is the NN rule defined previously. The Cover-Hart Theorem can be readily extended to the *k*NN case, which results in tighter bounds as k increases; for example, it can be shown that, for the 3NN rule, $\varepsilon_{\text{bay}} \leq \lim_{n\to\infty} E[\varepsilon_n] \leq 1.316\,\varepsilon_{\text{bay}}$, regardless of the distribution of $(\mathbf{X}, Y)$ (Devroye et al., 1996). The *k*NN rule is universally consistent if $k \to \infty$ and $k/n \to 0$ as $n \to \infty$ (Stone, 1977).

1.6.2 Support Vector Machines

The *support vector machine* (*SVM*) is based on the idea of a *maximal-margin hyperplane* (*MMH*) (Vapnik, 1998). Figure 1.4 shows a linearly separable data set and three hyperplanes (lines). The outer lines, which pass through points in the sample data, are the *margin hyperplanes*, whereas the third is the MMH, which is equidistant between the margin hyperplanes. The sample points closest to the MMH are on the margin hyperplanes and are called *support vectors* (the circled sample points in Figure 1.4). The distance from the MMH to any support vector is called the *margin*. The MMH defines a *linear SVM* classifier.

Let the equation for the MMH be $\mathbf{a}^T\mathbf{x} + b = 0$, where $\mathbf{a} \in R^d$ and $b \in R$ are the parameters of the linear SVM. It can be shown that the margin is given by $1/||\mathbf{a}||$ (see

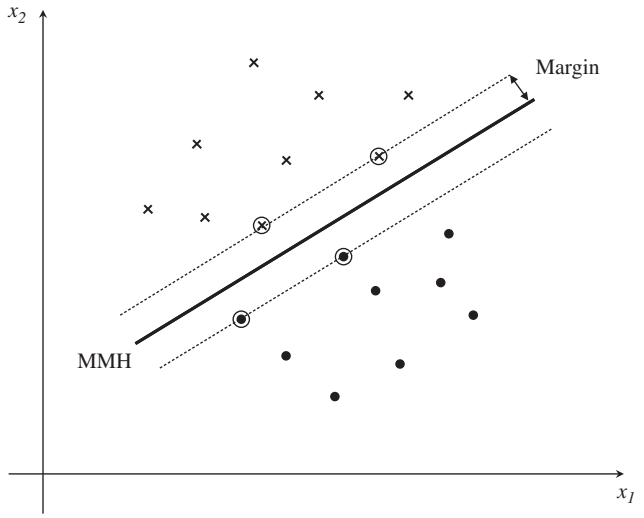


FIGURE 1.4 Maximal-margin hyperplane for linearly-separable data. The support vectors are the circled sample points.

Exercise 1.11). Since the goal is to maximize the margin, one finds the parameters of the MMH by solving the following quadratic optimization problem:

$$\begin{aligned} &\min \frac{1}{2}||a||^2, \text{ subject to} \\ &\begin{cases} a^T\mathbf{X}_i + b \geq 1\,, & \text{if } Y_i = 1\,, \\ a^T\mathbf{X}_i + b \leq -1\,, & \text{if } Y_i = 0\,, \end{cases} \quad \text{for } i = 1, \ldots, n\,. \end{aligned} \tag{1.67}$$

If the sample is not linearly separable, then one has two choices: find a reasonable linear classifier or find a nonlinear classifier. In the first case, the preceding method can be modified by making appropriate changes to the optimization problem (1.67); in the second case, one can map the sample points into a higher dimensional space where they are linearly separable, find a hyperplane in that space, and then map back into the original space; see (Vapnik, 1998; Webb, 2002) for the details.

1.6.3 Neural Networks

A (feed-forward) two-layer *neural network* has the form

$$\psi(\mathbf{x}) = \mathrm{T}\left[c_0 + \sum_{i=1}^{k} c_i \sigma[\phi_i(\mathbf{x})]\right], \tag{1.68}$$

where T thresholds at 0, σ is a *sigmoid function* (that is, a nondecreasing function with limits -1 and $+1$ at $-\infty$ and ∞ , respectively), and

$$\phi_i(\mathbf{x}) = a_{i0} + \sum_{j=1}^{d} a_{ij} x_j. \tag{1.69}$$

Each operator in the sum of (1.68) is called a *neuron*. These form the hidden layer. We consider neural networks with the threshold sigmoid: $\sigma(t) = -1$ if $t \leq 0$ and $\sigma(t) = 1$ if $t > 0$. If $k \to \infty$ such that $(k \log n)/n \to 0$ as $n \to \infty$, then, as a class, neural networks are universally consistent (Farago and Lugosi, 1993), but one should beware of the increasing number of neurons required.

Any absolutely integrable function can be approximated arbitrarily closely by a sufficiently complex neural network. While this is theoretically important, there are limitations to its practical usefulness. Not only does one not know the function, in this case a Bayes classifier, whose approximation is desired, but even were we to know the function and how to find the necessary coefficients, a close approximation can require an extremely large number of model parameters. Given the neural-network structure, the task is to estimate the optimal weights. As the number of model parameters grows, use of the model for classifier design becomes increasingly intractable owing to the increasing amount of data required for estimation of the model parameters. Since the number of hidden units must be kept relatively small, thereby requiring significant

constraint, when data are limited, there is no assurance that the optimal neural network of the prescribed form closely approximates the Bayes classifier. Model estimation is typically done by some iterative procedure, with advantages and disadvantages being associated with different methods (Bishop, 1995).

1.6.4 Classification Trees

The histogram rule partitions the space without reference to the actual data. One can instead partition the space based on the data, either with or without reference to the labels. Tree classifiers are a common way of performing data-dependent partitioning. Since any tree can be transformed into a binary tree, we need only consider binary classification trees. A tree is constructed recursively. If S represents the set of all data, then it is partitioned according to some rule into $S = S_1 \cup S_2$. There are four possibilities: (i) S_1 is split into $S_1 = S_{11} \cup S_{12}$ and S_2 is split into $S_2 = S_{21} \cup S_{22}$; (ii) S_1 is split into $S_1 = S_{11} \cup S_{12}$ and splitting of S_2 is terminated; (iii) S_2 is split into $S_2 = S_{21} \cup S_{22}$ and splitting of S_1 is terminated; and (iv) splitting of both S_1 and S_2 is terminated. In the last case, splitting is complete; in any of the others, it proceeds recursively until all branches end in termination, at which point the leaves on the tree represent the partition of the space. On each cell (subset) in the final partition, the designed classifier is defined to be 0 or 1, according to which the majority is among the labels of the points in the cell.

For *Classification and Regression Trees* (*CART*), splitting is based on an *impurity function*. For any rectangle R, let $N_0(R)$ and $N_1(R)$ be the numbers of 0-labeled and 1-labeled points, respectively, in R, and let $N(R) = N_0(R) + N_1(R)$ be the total number of points in R. The *impurity* of R is defined by

$$\kappa(R) = \zeta(p_R, 1 - p_R)\,, \tag{1.70}$$

where $p_R = N_0(R)/N(R)$ is the proportion of 0 labels in R, and where $\zeta(p, 1-p)$ is a nonnegative function satisfying the following conditions: (1) $\zeta(0.5, 0.5) \geq \zeta(p, 1-p)$ for any $p \in [0, 1]$; (2) $\zeta(0, 1) = \zeta(1, 0) = 0$; and (3) as a function of p, $\zeta(p, 1-p)$ increases for $p \in [0, 0.5]$ and decreases for $p \in [0.5, 1]$. Several observations follow from the definition of ζ : (1) $\kappa(R)$ is maximum when the proportions of 0-labeled and 1-labeled points in R are equal (corresponding to maximum impurity); (2) $\kappa(R) = 0$ if R is pure; and (3) $\kappa(R)$ increases for greater impurity.

We mention three possible choices for ζ:

1. $\zeta_e(p, 1-p) = -p \log p - (1-p)\log(1-p)$ (*entropy impurity*)
2. $\zeta_g(p, 1-p) = p(1-p)$ (*Gini impurity*)
3. $\zeta_m(p, 1-p) = \min(p, 1-p)$ (*misclassification impurity*)

Regarding these three impurities: $\zeta_e(p, 1-p)$ provides an entropy estimate, $\zeta_g(p, 1-p)$ provides a variance estimate for a binomial distribution, and $\zeta_m(p, 1-p)$ provides a misclassification-rate estimate.

A splitting regimen is determined by the manner in which a split will cause an overall decrease in impurity. Let i be a coordinate, α be a real number, R be a rectangle to be split along the ith coordinate, $R^i_{\alpha,-}$ be the sub-rectangle resulting from the ith coordinate being less than or equal to α, and $R^i_{\alpha,+}$ be the sub-rectangle resulting from the ith coordinate being greater than α. Define the *impurity decrement* by

$$\Delta_i(R,\alpha) \stackrel{\Delta}{=} \kappa(R) - \frac{N\left(R^i_{\alpha,-}\right)}{N(R)}\kappa\left(R^i_{\alpha,-}\right) - \frac{N\left(R^i_{\alpha,+}\right)}{N(R)}\kappa\left(R^i_{\alpha,+}\right). \tag{1.71}$$

A good split will result in impurity reductions in the sub-rectangles. In computing $\Delta_i(R, \alpha)$, the new impurities are weighted by the proportions of points going into the sub-rectangles. CART proceeds iteratively by splitting a rectangle at $\hat{\alpha}$ on the ith coordinate if $\Delta_i(R, \alpha)$ is maximized for $\alpha = \hat{\alpha}$. There exist two splitting strategies: (i) the coordinate i is given and $\Delta_i(R, \alpha)$ is maximized over all α; (ii) the coordinate is not given and $\Delta_i(R, \alpha)$ is maximized over all i, α. Notice that only a finite number of candidate α need to be considered; say, the midpoints between adjacent training points. The search can therefore be exhaustive. Various stopping strategies are possible — for instance, stopping when maximization of $\Delta_i(R, \alpha)$ yields a value below a preset threshold, or when there are fewer than a specified number of sample points assigned to the node. One may also continue to grow the tree until all leaves are pure and then prune.

1.6.5 Rank-Based Rules

Rank-based classifiers utilize only the relative order between feature values, that is, their ranks. This produces classification rules that are very simple and thus are likely to avoid overfitting. The simplest example, the *Top-Scoring Pair* (TSP) classification rule, was introduced in Geman et al. (2004). Given two feature indices $1 \leq i, j \leq d$, with $i \neq j$, the TSP classifier is given by

$$\psi(\mathbf{x}) = \begin{cases} 1\,, & x_i < x_j\,, \\ 0\,, & \text{otherwise.} \end{cases} \tag{1.72}$$

Therefore, the TSP classifier depends only on the feature ranks, and not on their magnitudes. In addition, given (i, j), the TSP classifier is independent of the sample data; that is, there is minimum adjustment of the decision boundary. However, the pair (i, j) is selected by using the sample data $S_n = \{(X_{11}, \ldots, X_{1d}, Y_1), \ldots, (X_{n1}, \ldots, X_{nd}, Y_n)\}$, as follows:

$$\begin{aligned}(i,j) &= \arg\max_{1\leq i,j\leq d}\left[\hat{P}(X_i < X_j \mid Y = 1) - \hat{P}(X_i < X_j \mid Y = 0)\right] \\ &= \arg\max_{1\leq i,j\leq d}\left[\frac{1}{N_0}\sum_{k=1}^{n} I_{X_{ki}<X_{kj}} I_{Y_k=1} - \frac{1}{N_1}\sum_{k=1}^{n} I_{X_{ki}<X_{kj}} I_{Y_k=0}\right].\end{aligned} \tag{1.73}$$

This approach is easily extended to k pairs of features, $k = 1, 2, \ldots$, yielding a *kTSP* classification rule, and additional operations on ranks, such as the median, yielding a *Top Scoring Median* (TSM) classification rule (Afsari et al., 2014). If d is large, then the search in (1.73) is conducted using an efficient greedy algorithm; see Afsari et al. (2014) for the details.

1.7 FEATURE SELECTION

The motivation for increasing classifier complexity is to achieve finer discrimination, but as we have noted, increasing the complexity of classification rules can be detrimental because it can result in overfitting during the design process. One way of achieving greater discrimination is to utilize more features, but too many features when designing from sample data can result in increased expected classification error. This can be seen in the VC dimension of a linear classification rule, which has VC dimension $d + 1$ in R^d (c.f. Appendix B). With the proliferation of sensing technologies capable of measuring thousands or even millions of features (also known as "big data") at typically small sample sizes, this has become an increasingly serious issue. In addition to improving classification performance, other reasons to apply feature selection include the reduction of computational load, both in terms of execution time and data storage, and scientific interpretability of the designed classifiers.

A key issue is monotonicity of the expected classification error relative to the dimensionality of the feature vector. Let $\mathbf{X}' = \upsilon(\mathbf{X})$, where $\upsilon : R^D \to R^d$, with $D > d$, is a *feature selection transformation*, that is, it is an orthogonal projection from the larger dimensional space to the smaller one, which simply discards components from vector $\mathbf{X}$ to produce $\mathbf{X}'$. It can be shown that the Bayes error is monotone: $\varepsilon_{\text{bay}}(\mathbf{X}, Y) \leq \varepsilon_{\text{bay}}(\mathbf{X}', Y)$; for a proof, see Devroye et al (1996, Theorem 3.3). In other words, the optimal error rate based on the larger feature vector is smaller than or equal to the one based on the smaller one (this property in fact also applies to more general dimensionality reduction transformations). However, for the expected error rate of classifiers designed on sample data, this is not true: $E[\varepsilon_n(\mathbf{X}, Y)]$ could be smaller, equal, or larger than $E[\varepsilon_n(\mathbf{X}', Y)]$. Indeed, it is common for the expected error to decrease at first and then increase for increasingly large feature sets. This behavior is known as the *peaking phenomenon* or the *curse of dimensionality* (Hughes, 1968; Jain and Chandrasekaran, 1982). Peaking is illustrated in Figure 1.5, where the expected classification error rate is plotted against sample size and number of features, for different classification rules and Gaussian populations with blocked covariance matrices. The black line indicates where the error rate peaks (bottoms out) for each fixed sample size. For this experiment, the features are listed in a particular order, and they are added in that order to increase the size of the feature set — see Hua et al. (2005a) for the details. In fact, the peaking phenomenon can be quite complex. In Figure 1.5a, peaking occurs with very few features for sample sizes below 30, but exceeds 30 features for sample sizes above 90. In Figure 1.5b, even with a sample size of 200, the optimal number of features is only eight. The concave behavior and increasing number of optimal features in parts (a) and (b) of the figure,

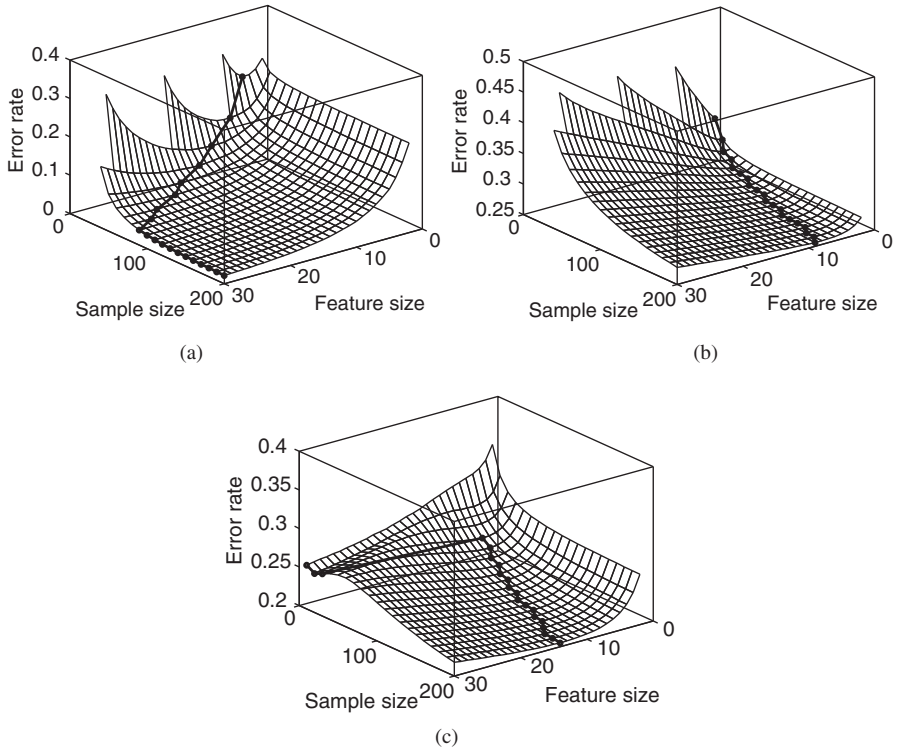


FIGURE 1.5 Expected classification error as a function of sample size and number of features, illustrating the peaking phenomenon. The black line indicates where the error rate peaks (bottoms out) for each fixed sample size. (a) LDA in model with common covariance matrix and slightly correlated features; (b) LDA in model with common covariance matrix and highly correlated features; (c) linear support vector machine in model with unequal covariance matrices. Reproduced from Hua et al. (2005) with permission from Oxford University Press.

which employs the LDA classification rule, corresponds to the usual understanding of peaking. But in Figure 1.5c, which employs a linear SVM, not only does the optimal number of features not increase as a function of sample size, but for fixed sample size the error curve is not concave. For some sizes the error decreases, increases, and then decreases again as the feature-set size grows, thereby forming a ridge across the expected-error surface. This behavior is not rare and applies to other classification rules. We can also see in Figure 1.5c what appears to be a peculiarity of the linear SVM classification rule: for small sample sizes, the error rate does not seem to peak at all. This odd phenomenon is supported by experimental evidence from other studies on linear SVMs (Afra and Braga-Neto, 2011).

Let S_n^D be the sample data including all D original features, and let Ψ_n^d be a classification rule defined on the low-dimensional sample space. A *feature selection rule* is a mapping $\Upsilon_n : (\Psi_n^d, S_n^D) \mapsto \upsilon_n$, that is, given a classification rule Ψ_n^d and sample

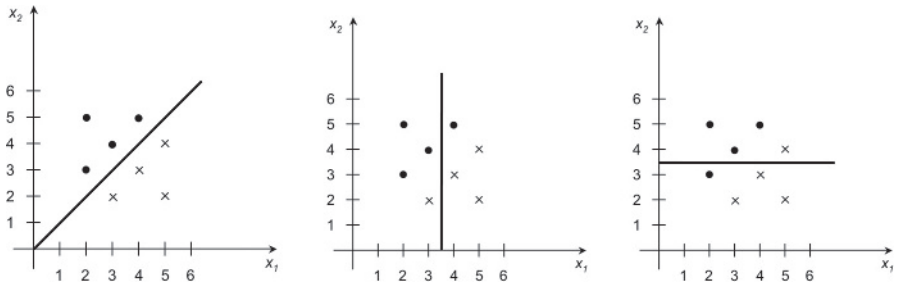


FIGURE 1.6 Constraint introduced by feature selection on a linear classification rule.

S_n^D, a feature selection transformation $\upsilon_n = \Upsilon_n(\Psi_n^d, S_n^D)$ is obtained according to the rule Υ_n. The reduced sample in the smaller space is $S_n^d = \{(\mathbf{X}_1', Y_1), \ldots, (\mathbf{X}_n', Y_n)\}$, where $\mathbf{X}_i' = \upsilon_n(\mathbf{X}_i)$, for $i = 1, \ldots, n$. Typically, feature selection is followed by classifier design using the selected feature set, according to the classification rule Ψ_n^d. This composition of feature selection and classifier design defines a new classification rule Ψ_n^D on the original high-dimensional feature space. If ψ_n^d denotes the classifier designed by Ψ_n^d on the smaller space, then the classifier designed by Ψ_n^D on the larger space is given by $\psi_n^D(\mathbf{X}) = \psi_n^d(\upsilon_n(\mathbf{X}))$. This means that feature selection becomes part of the classification rule in higher-dimensional space. This will be very important in Chapter 2 when we discuss error estimation rules in the presence of feature selection.

Since υ_n is an orthogonal projection, the decision boundary produced by ψ_n^D is an extension at right angles of the one produced by ψ_n^d. If the decision boundary in the lower-dimensional space is a hyperplane, then it is still a hyperplane in the higher-dimensional space, but a hyperplane that is not allowed to be oriented along any direction, but must be orthogonal to the lower-dimensional space. See Figure 1.6 for an illustration. Therefore, feature selection corresponds to a constraint on the classification rule in the sense of Section 1.3, and not just a reduction in dimensionality. As discussed in Section 1.3, there is a cost of constraint, which the designer is willing to incur if it is offset by a sufficient decrease in the expected design error.

Feature selection rules can be categorized into two broad classes: if the feature selection rule does not depend on Ψ_n^d, then it is called a *filter* rule, in which case $\upsilon_n = \Upsilon_n(S_n^D)$; otherwise, it is called a *wrapper* method. One could yet have a hybrid between the two, in case two rounds of feature selection are applied consecutively. In addition, feature selection rules can be *exhaustive*, which evaluate all $\binom{D}{d}$ possible projections from the higher-dimensional space to the lower-dimensional space, or *nonexhaustive*, otherwise. A typical exhaustive wrapper feature selection rule applies the classification rule Ψ_n^d on each of the $\binom{D}{d}$ possible feature sets of size d, obtains an estimate of the error of the resulting classifier using one of the techniques discussed in the next chapter, and then selects the feature set with the minimum estimated error. This creates a complex interaction between feature selection and error estimation (Sima et al., 2005a).

If D, and especially d, are large enough, then exhaustive feature selection is not possible due to computational reasons, and "greedy" nonexhaustive methods that only explore a subset of the search space must be used. Filter methods are typically of this kind. A simple and very popular example is based on finding individual features for which the class means are far apart relative to their pooled variance. The *t-test* statistic is typically employed for that purpose. An obvious problem with choosing features individually is that, while the selected features may perform individually the best, the approach is subject to choosing a feature set with a large number of redundant features and, in addition, features that perform poorly individually may do well in combination with other features; these will not be picked up. A classical counter example that illustrates this is *Toussaint's Counterexample*, where there are three original features and the best pair of individual features selected individually is the worst pair when considered together, according to the Bayes error; see Devroye et al. (1996) for the details.

As for nonexhaustive wrapper methods, the classical approaches are *branch-and-bound* algorithms, *sequential forward and backward selection*, and their variants (Webb, 2002). For example, sequential forward selection begins with a small set of features, perhaps one, and iteratively builds the feature set by adding more features as long as the estimated classification error is reduced. If features are allowed to be removed from the current feature set, in order to avoid bad features to be frozen in place, then one has a *floating* method. The standard greedy wrapper feature selection methods are *sequential forward selection* (SFS) and *sequential floating forward selection* (SFFS) (Pudil et al., 1994). The efficacy of greedy methods, that is, how fast and how close they get to the solution found by the corresponding exhaustive method, depends on the feature–label distribution, the classification rule and error estimation rule used, as well as the sample size and the original and target dimensionalities. A fundamental "no-free-lunch" result in Pattern Recognition specifies that, without distributional knowledge, no greedy method can always avoid the full complexity of the exhaustive search; that is, there will always be a feature–label distribution for which the algorithm must take as long as exhaustive search to find the best feature set. This result is known as the Cover–Campenhout Theorem (Cover and van Campenhout, 1977). Evaluation of methods is generally comparative and often based on simulations (Hua et al., 2009; Jain and Zongker, 1997; Kudo and Sklansky, 2000). Moreover, feature selection affects peaking. The classical way of examining peaking is to list the features in some order and then adjoin one at a time. This is how Figure 1.5 has been constructed. Peaking can be much more complicated when feature selection is involved (Sima and Dougherty, 2008).

EXERCISES

1.1 Suppose the class-conditional densities $p_0(\mathbf{x})$ and $p_1(\mathbf{x})$ are uniform distributions over a unit-radius disk centered at the origin and a unit-area square centered at $(\frac{1}{2}, \frac{1}{2})$, respectively. Find the Bayes classifier and the Bayes error.

1.2 Prove that $\varepsilon_{\text{bay}} = \frac{1}{2}(1 - E[|2\eta_1(\mathbf{X}) - 1|])$.

1.3 Consider Cauchy class-conditional densities:

$$p_i(x) = \frac{1}{\pi b} \frac{1}{1 + \left(\frac{x - a_i}{b}\right)^2}, \quad i = 0, 1,$$

where $-\infty < a_0 < a_1 < \infty$ are location parameters (the Cauchy distribution does not have a mean), and $b > 0$ is a dispersion parameter. Assume that the classes are equally likely: $c_0 = c_1 = \frac{1}{2}$.

(a) Determine the Bayes classifier.

(b) Write the Bayes error as a function of the parameters a_0, a_1, and b.

(c) Plot the Bayes error as a function of $(a_1 - a_0)/b$ and explain what you see. In particular, what are the maximum and minimum (infimum) values of the curve and what do they correspond to?

1.4 This problem concerns the extension to the multiple-class case of concepts developed in Section 1.1. Let $Y \in \{0, 1, \ldots, c-1\}$, where c is the number of classes, and let

$$\eta_i(\mathbf{x}) = P(Y = i \mid \mathbf{X} = \mathbf{x}), \quad i = 0, 1, \ldots, c-1,$$

for each $\mathbf{x} \in R^d$. We need to remember that these probabilities are not independent, but satisfy $\eta_0(\mathbf{x}) + \eta_1(\mathbf{x}) + \cdots + \eta_{c-1}(\mathbf{x}) = 1$, for each $\mathbf{x} \in R^d$, so that one of the functions is redundant. Hint: you should answer the following items in sequence, using the previous answers in the solution of the following ones.

(a) Given a classifier $\psi : R^d \to \{0, 1, \ldots, c-1\}$, show that its *conditional error* at a point $\mathbf{x} \in R^d$ is given by

$$\begin{aligned} \varepsilon[\psi \mid \mathbf{X} = \mathbf{x}] &= P(\psi(\mathbf{X}) \neq Y \mid \mathbf{X} = \mathbf{x}) \\ &= 1 - \sum_{i=0}^{c-1} I_{\psi(\mathbf{x})=i}\, \eta_i(\mathbf{x}) = 1 - \eta_{\psi(\mathbf{x})}(\mathbf{x}). \end{aligned}$$

(b) Show that the classification error is given by

$$\varepsilon[\psi] = 1 - \sum_{i=0}^{c-1} \int_{\{\mathbf{x} \mid \psi(\mathbf{x})=i\}} \eta_i(\mathbf{x}) p(\mathbf{x})\, d\mathbf{x}.$$

Hint: $\varepsilon[\psi] = E[\varepsilon[\psi \mid \mathbf{X}]]$.

(c) Prove that the Bayes classifier is given by

$$\psi_{\text{bay}}(\mathbf{x}) = \arg\max_{i=0,1,\ldots,c-1} \eta_i(\mathbf{x}), \quad \mathbf{x} \in R^d.$$

Hint: Start by considering the difference between conditional errors $P(\psi(\mathbf{X}) \neq Y \mid \mathbf{X} = \mathbf{x}) - P(\psi_{\text{bay}}(\mathbf{X}) \neq Y \mid \mathbf{X} = \mathbf{x})$.

(d) Show that the Bayes error is given by

$$\varepsilon_{\text{bay}} = 1 - E\left[\max_{i=0,1,\ldots,c-1} \eta_i(\mathbf{X})\right].$$

(e) Show that the results in parts (b), (c), and (d) reduce to (1.1), (1.2), and (1.3), respectively, in the two-class case $c = 2$.

1.5 Supose $\hat{\eta}_{0,n}$ and $\hat{\eta}_{1,n}$ are estimates of η_0 and η_1, respectively, for a sample of size n. The *plug-in rule* is defined by using $\hat{\eta}_{0,n}$ and $\hat{\eta}_{1,n}$ in place of η_0 and η_1, respectively, in (1.2). Derive the design error for the plug-in rule and show that it satisfies the bounds $\Delta_n \leq 2E[|\eta_1(\mathbf{X}) - \hat{\eta}_{1,n}(\mathbf{X})|]$ and $\Delta_n \leq 2E[|\eta_1(\mathbf{X}) - \hat{\eta}_{1,n}(\mathbf{X})|^2]^{1/2}$, thereby providing two criteria for consistency.

1.6 This problem concerns the Gaussian case, (1.17).

(a) Given a general linear discriminant $W_{\text{lin}}(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b$, where $\mathbf{a} \in R^d$ and $b \in R$ are arbitrary parameters, show that the error of the associated classifier

$$\psi(\mathbf{x}) = \begin{cases} 1, & W_{\text{lin}}(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b < 0 \\ 0, & \text{otherwise} \end{cases}$$

is given by

$$\varepsilon[\psi] = c_0\,\Phi\left(\frac{\mathbf{a}^T\boldsymbol{\mu}_0 + b}{\sqrt{\mathbf{a}^T\Sigma_0\mathbf{a}}}\right) + c_1\,\Phi\left(-\frac{\mathbf{a}^T\boldsymbol{\mu}_1 + b}{\sqrt{\mathbf{a}^T\Sigma_1\mathbf{a}}}\right), \quad (1.74)$$

where Φ is the cumulative distribution of a standard normal random variable.

(b) Using the result from part (a) and the expression for the optimal linear discriminant in (1.19), show that if $\Sigma_0 = \Sigma_1 = \Sigma$ and $c_0 = c_1 = \frac{1}{2}$, then the Bayes error for the problem is given by (1.24):

$$\varepsilon_{\text{bay}} = \Phi\left(-\frac{\delta}{2}\right),$$

where $\delta = \sqrt{(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T\Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}$ is the Mahalanobis distance between the populations. Hence, there is a tight relationship (in fact, one-to-one) between the Mahalanobis distance and the Bayes error. What is the maximum and minimum (infimum) Bayes errors and when do they happen?

1.7 Determine the shatter coefficients and the VC dimension for

(a) The 1NN classification rule.

(b) A binary tree classification rule with data-independent splits and a fixed depth of k levels of splitting nodes.

(c) A classification rule that partitions the feature space into a finite number of b cells.

1.8 In this exercise we define two models to be used to generate synthetic data. In both cases, the dimension is d, the means are at the points $\boldsymbol{\mu}_0 = (0, \ldots, 0)$ and $\boldsymbol{\mu}_1 = (u, \ldots, u)$, and $\Sigma_0 = I$. For model M1,

$$\Sigma_1 = \sigma^2 \begin{pmatrix} 1 & \rho & \rho & \cdots & \rho \\ \rho & 1 & \rho & \cdots & \rho \\ \rho & \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \vdots & \ddots & \rho \\ \rho & \rho & \rho & \cdots & 1 \end{pmatrix},$$

where Σ_1 is $d \times d$, and for model M2,

$$\Sigma_1 = \sigma^2 \begin{pmatrix} 1 & \rho & \rho & \cdots & 0 & 0 & 0 \\ \rho & 1 & \rho & \cdots & 0 & 0 & 0 \\ \rho & \rho & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \rho & \rho \\ 0 & 0 & 0 & \cdots & \rho & 1 & \rho \\ 0 & 0 & 0 & \cdots & \rho & \rho & 1 \end{pmatrix},$$

where Σ_1 is a $d \times d$ blocked matrix, consisting of 3×3 blocks along the diagonal. The prior class probabilities are $c_0 = c_1 = \frac{1}{2}$. For each model, the Bayes error is a function of d, u, ρ, and σ. For $d = 6$, $\sigma = 1, 2$, and $\rho = 0, 0.2, 0.8$, find for each model the value of u so that the Bayes error is $0.05, 0.15, 0.25, 0.35$.

1.9 For the models M1 and M2 of Exercise 1.8 with $d = 6$, $\sigma = 1, 2$, $\rho = 0, 0.2, 0.8$, and $n = 25, 50, 200$, plot $E[\varepsilon_n]$ against the Bayes error values $0.05, 0.15, 0.25, 0.35$, for the LDA, QDA, 3NN, and linear SVM classification rules. Hint: you will plot $E[\varepsilon_n]$ against the values of u corresponding to the Bayes error values $0.05, 0.15, 0.25, 0.35$.

1.10 The sampling mechanism considered so far is called "mixture sampling." It is equivalent to selecting Y_i from a Bernoulli random variable with mean $c_1 = P(Y = 1)$ and then sampling $\mathbf{X}_i$ from population Π_j if $Y_i = j$, for $i = 1, \ldots, n$, $j = 0, 1$. The numbers of sample points N_0 and N_1 from each population are therefore random. Consider the case where the sample sizes n_0 and n_1 are not random, but fixed before sampling occurs. This mechanism is called "separate sampling," and will be considered in detail in Chapters 5–7. For the model M1 of Exercise 1.8, the LDA and 3NN classification rules, $d = 6$, $c_0 = c_1 = \frac{1}{2}$, $\sigma = 1$, $\rho = 0$, and Bayes error 0.25, find $E[\varepsilon_{n_0,n_1}]$ using separate sampling with $n_0 = 50$ and $n_1 = 50$, and compare the result to mixture sampling with $n = 100$. Repeat for $n_0 = 25$, $n_1 = 75$ and again compare to mixture sampling.

1.11 Consider a general linear discriminant $W_{\text{lin}}(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b$, where $\mathbf{a} \in R^d$ and $b \in R$ are arbitrary parameters.

(a) Show that the Euclidean distance of a point $\mathbf{x}_0$ to the hyperplane $W_{\text{lin}}(\mathbf{x}) = 0$ is given by $|W_{\text{lin}}(\mathbf{x}_0)|/||\mathbf{a}||$. Hint: Minimize the distance $||\mathbf{x} - \mathbf{x}_0||$ of a point $\mathbf{x}$ on the hyperplane to the point $\mathbf{x}_0$.

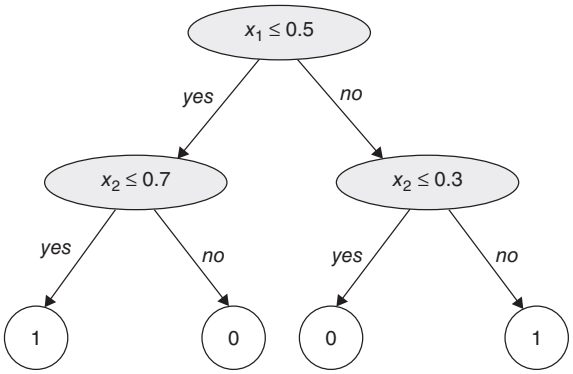
(b) Use part(a) to show that the margin for a linear SVM is $1/||\mathbf{a}||$.

1.12 This problem concerns a parallel between discrete classification and Gaussian classification. Let $\mathbf{X} = (X_1, \ldots, X_d) \in \{0, 1\}^d$ be a discrete feature vector, that is, all individual features are binary. Assume furthermore that the features are conditionally independent given $Y = 0$ and given $Y = 1$ — compare this to spherical class-conditional Gaussian densities, where the features are also conditionally independent given $Y = 0$ and given $Y = 1$.

(a) As in the Gaussian case with equal covariance matrices, prove that the Bayes classifier is linear, that is, show that $\psi_{\text{bay}}(\mathbf{x}) = I_{D_L(\mathbf{x})>0}$, for $\mathbf{x} \in \{0, 1\}^d$, where $D_L(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b$. Give the values of $\mathbf{a}$ and b in terms of the class-conditional distribution parameters $p_i = P(\mathbf{X}_i = 1|Y = 0)$ and $q_i = P(\mathbf{X}_i = 1|Y = 1)$, for $i = 1, \ldots, d$ (notice that these are different than the parameters p_i and q_i defined in Section 1.5), and the prior probabilities $c_0 = P(Y = 0)$ and $c_1 = P(Y = 1)$.

(b) Suppose that sample data $S_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ is available, where $\mathbf{X}_i = (X_{i1}, \ldots, X_{id}) \in \{0, 1\}^d$, for $i = 1, \ldots, d$. Just as is done for LDA in the Gaussian case, obtain a sample discriminant $W_{\text{lin}}(\mathbf{x})$ from $D_L(\mathbf{x})$ in the previous item, by plugging in maximum-likelihood (ML) estimators $\hat{p}_i$, $\hat{q}_i$, $\hat{c}_0$, and $\hat{c}_1$ for the unknown parameters p_i, q_i, c_0 and c_1. The maximum-likelihood estimators in this case are the empirical frequencies (you may use this fact without showing it). Show that $W_{\text{lin}}(S_n, \mathbf{x}) = \mathbf{a}(S_n)^T\mathbf{x} + b(S_n)$, where $\mathbf{a}(S_n) \in R^d$ and $b(S_n) \in R$. Hence, this is a sample-based linear discriminant, just as in the case of sample-based LDA. Give the values of $\mathbf{a}(S_n)$ and $b(S_n)$ in terms of $S_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$.

1.13 Consider the simple CART classifier in R^2 depicted below.



Design an equivalent two-hidden-layer neural network with threshold sigmoids, with three neurons in the first hidden layer and four neurons in the second hidden layer. Hint: Note the correspondence between the number of neurons and the number of splitting and leaf nodes.

1.14 Let $\mathbf{X} \in R^d$ be a feature set of size d, and let $X_0 \in R$ be an additional feature. Define the augmented feature set $\mathbf{X}' = (\mathbf{X}, X_0) \in R^{d+1}$. It was mentioned in Section 1.7 that the Bayes error satisfies the monotonicity property $\varepsilon_{\text{bay}}(\mathbf{X}, Y) \leq \varepsilon_{\text{bay}}(\mathbf{X}', Y)$. Show that a sufficient condition for the undesirable case where there is no improvement, that is, $\varepsilon_{\text{bay}}(\mathbf{X}, Y) = \varepsilon_{\text{bay}}(\mathbf{X}', Y)$, is that X_0 be independent of $(\mathbf{X}, Y)$ (in which case, the feature X_0 is redundant or "noise").