Chapter 5
Discriminant Analysis

This chapter considers discriminant analysis: given p measurements $\boldsymbol{w}$, we want to correctly classify $\boldsymbol{w}$ into one of G groups or populations. The maximum likelihood, Bayesian, and Fisher's discriminant rules are used to show why methods like linear and quadratic discriminant analysis can work well for a wide variety of group distributions.

5.1 Introduction

Definition 5.1. In *supervised classification*, there are G known groups and m test cases to be classified. Each test case is assigned to exactly one group based on its measurements $\boldsymbol{w}_i$.

Suppose there are G populations or groups or classes where $G \geq 2$. Assume that for each population there is a probability density function (pdf) $f_j(\boldsymbol{z})$ where $\boldsymbol{z}$ is a $p \times 1$ vector and $j = 1, ..., G$. Hence if the random vector $\boldsymbol{x}$ comes from population j, then $\boldsymbol{x}$ has pdf $f_j(\boldsymbol{z})$. Assume that there is a random sample of n_j cases $\boldsymbol{x}_{1,j}, ..., \boldsymbol{x}_{n_j,j}$ for each group. Let $(\overline{\boldsymbol{x}}_j, \boldsymbol{S}_j)$ denote the sample mean and covariance matrix for each group. Let $\boldsymbol{w}_i$ be a new $p \times 1$ (observed) random vector from one of the G groups, but the group is unknown. Usually there are many $\boldsymbol{w}_i$, and *discriminant analysis* (DA) or *classification* attempts to allocate the $\boldsymbol{w}_i$ to the correct groups. The $\boldsymbol{w}_1, ..., \boldsymbol{w}_m$ are known as the *test data*. Let π_k = the (prior) probability that a randomly selected case $\boldsymbol{w}_i$ belongs to the kth group. If $\boldsymbol{x}_{1,1}..., \boldsymbol{x}_{n_G,G}$ are a random sample of cases from the collection of G populations, then $\hat{\pi}_k = n_k/n$ where $n = \sum_{i=1}^{G} n_i$. Often the *training data* $\boldsymbol{x}_{1,1}, ..., \boldsymbol{x}_{n_G,G}$ is not collected in this manner. Often the n_k are fixed numbers such that n_k/n does not estimate π_k. For example, suppose $G = 2$ where $n_1 = 100$ and $n_2 = 100$ where patients in group 1 have a deadly disease and patients in group 2 are healthy, but an attempt has been made to match the sick patients with healthy patients on p variables such as

age, weight, height, an indicator for smoker or nonsmoker, and gender. Then using $\hat{\pi}_j = 0.5$ does not make sense because π_1 is much smaller than π_2. Here the indicator variable is qualitative, so the p variables do not have a pdf.

Let $\boldsymbol{W}_i$ be the random vector and $\boldsymbol{w}_i$ be the observed random vector. Let $Y = j$ if $\boldsymbol{w}_i$ comes from the jth group for $j = 1, ..., G$. Then $\pi_j = P(Y = j)$ and the *posterior probability* that $Y = k$ or that $\boldsymbol{w}_i$ belongs to group k is

$$p_k(\boldsymbol{w}_i) = P(Y = k|\boldsymbol{W}_i = \boldsymbol{w}_i) = \frac{\pi_k f_k(\boldsymbol{w}_i)}{\sum_{j=1}^{G} \pi_j f_j(\boldsymbol{w}_i)}. \tag{5.1}$$

Definition 5.2. a) The *maximum likelihood discriminant rule* allocates case $\boldsymbol{w}_i$ to group a if $\hat{f}_a(\boldsymbol{w}_i)$ maximizes $\hat{f}_j(\boldsymbol{w}_i)$ for $j = 1, ..., G$.

b) The *Bayesian discriminant rule* allocates case $\boldsymbol{w}_i$ to group a if $\hat{p}_a(\boldsymbol{w}_i)$ maximizes

$$\hat{p}_k(\boldsymbol{w}_i) = \frac{\hat{\pi}_k \hat{f}_k(\boldsymbol{w}_i)}{\sum_{j=1}^{G} \hat{\pi}_j \hat{f}_j(\boldsymbol{w}_i)}$$

for $k = 1, ..., G$.

c) The (population) *Bayes classifier* allocates case $\boldsymbol{w}_i$ to group a if $p_a(\boldsymbol{w}_i)$ maximizes $p_k(\boldsymbol{w}_i)$ for $k = 1, ..., G$.

Note that the above rules are robust to nonnormality of the G groups. Following James et al. (2013, pp. 38-39, 139), the Bayes classifier has the lowest possible expected test error rate out of all classifiers using the same p predictor variables $\boldsymbol{w}$. Of course typically the π_j and f_j are unknown. Note that the maximum likelihood rule and the Bayesian discriminant rule are equivalent if $\hat{\pi}_j \equiv 1/G$ for $j = 1, ..., G$. If p is large, or if there is multicollinearity among the predictors, or if some of the predictor variables are noise variables (useless for prediction), then there is likely a subset $\boldsymbol{z}$ of d of the p variables $\boldsymbol{w}$ such that the Bayes classifier using $\boldsymbol{z}$ has lower error rate than the Bayes classifier using $\boldsymbol{w}$.

Several of the discriminant rules in this chapter can be modified to incorporate π_j and costs of correct and incorrect allocation. See Johnson and Wichern (1988, ch. 11). We will assume that costs of correct allocation are unknown or equal to 0, and that costs of incorrect allocation are unknown or equal. Unless stated otherwise, assume that the probabilities π_j that $\boldsymbol{w}_i$ is in group j are unknown or equal: $\pi_j = 1/G$ for $j = 1, ..., G$. Some rules can handle discrete predictors.

5.2 LDA and QDA

Often it is assumed that the G groups have the same covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{x}}$. Then the pooled covariance matrix estimator is

$$\boldsymbol{S}_{pool} = \frac{1}{n-G}\sum_{j=1}^{G}(n_j - 1)\boldsymbol{S}_j \tag{5.2}$$

where $n = \sum_{j=1}^{G} n_j$. The pooled estimator $\boldsymbol{S}_{pool}$ can also be useful if some of the n_i are small so that the $\boldsymbol{S}_j$ are not good estimators. Let $(\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j)$ be the estimator of multivariate location and dispersion for the jth group, e.g. the sample mean and sample covariance matrix $(\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j) = (\overline{\boldsymbol{x}}_j, \boldsymbol{S}_j)$. Then a pooled estimator of dispersion is

$$\hat{\boldsymbol{\Sigma}}_{pool} = \frac{1}{k-G}\sum_{j=1}^{G}(k_j - 1)\hat{\boldsymbol{\Sigma}}_j \tag{5.3}$$

where often $k = \sum_{j=1}^{G} k_j$ and often k_j is the number of cases used to compute $\hat{\boldsymbol{\Sigma}}_j$.

LDA is especially useful if the population dispersion matrices are equal: $\boldsymbol{\Sigma}_j \equiv \boldsymbol{\Sigma}$ for $j = 1, ..., G$. Then $\hat{\boldsymbol{\Sigma}}_{pool}$ is an estimator of $c\boldsymbol{\Sigma}$ for some constant $c > 0$ if each $\hat{\boldsymbol{\Sigma}}_j$ is a consistent estimator of $c_j\boldsymbol{\Sigma}$ where $c_j > 0$ for $j = 1, ..., G$. If LDA does not work well with predictors $\boldsymbol{x} = (X_1, ..., X_p)$, try adding squared terms X_i^2 and possibly two way interaction terms X_iX_j. If all squared terms and two way interactions are added, LDA will often perform like QDA.

Definition 5.3. Let $\hat{\boldsymbol{\Sigma}}_{pool}$ be a pooled estimator of dispersion. Then the *linear discriminant rule* is allocate $\boldsymbol{w}$ to the group with the largest value of

$$d_j(\boldsymbol{w}) = \hat{\boldsymbol{\mu}}_j^T \hat{\boldsymbol{\Sigma}}_{pool}^{-1}\boldsymbol{w} - \frac{1}{2}\hat{\boldsymbol{\mu}}_j^T \hat{\boldsymbol{\Sigma}}_{pool}^{-1}\hat{\boldsymbol{\mu}}_j = \hat{\alpha}_j + \hat{\boldsymbol{\beta}}_j^T \boldsymbol{w}$$

where $j = 1, ..., G$. *Linear discriminant analysis* (LDA) uses $(\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_{pool}) = (\overline{\boldsymbol{x}}_j, \boldsymbol{S}_{pool})$.

Definition 5.4. The *quadratic discriminant rule* is allocate $\boldsymbol{w}$ to the group with the largest value of

$$Q_j(\boldsymbol{w}) = \frac{-1}{2}\log(|\hat{\boldsymbol{\Sigma}}_j|) - \frac{1}{2}(\boldsymbol{w} - \hat{\boldsymbol{\mu}}_j)^T \hat{\boldsymbol{\Sigma}}_j^{-1}(\boldsymbol{w} - \hat{\boldsymbol{\mu}}_j)$$

where $j = 1, ..., G$. *Quadratic discriminant analysis* (QDA) uses $(\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j) = (\overline{\boldsymbol{x}}_j, \boldsymbol{S}_j)$.

Definition 5.5. The *distance discriminant rule* allocates $\boldsymbol{w}$ to the group with the smallest squared distance $D^2_{\boldsymbol{w}}(\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j) = (\boldsymbol{w} - \hat{\boldsymbol{\mu}}_j)^T \hat{\boldsymbol{\Sigma}}_j^{-1} (\boldsymbol{w} - \hat{\boldsymbol{\mu}}_j)$ where $j = 1, ..., G$.

Examining some of the rules for $G = 2$ and one predictor w is informative. First, assume group 2 has a uniform(−10,10) distribution and group 1 has a uniform$(a - 1, a + 1)$ distribution. If $a = 0$ is known, then the maximum likelihood discriminant rule assigns w to group 1 if $-1 < w < 1$ and assigns w to group 2, otherwise. This occurs since $f_2(w) = 1/20$ for $-10 < w < 10$ and $f_2(w) = 0$, otherwise, while $f_1(w) = 1/2$ for $-1 < w < 1$ and $f_1(w) = 0$, otherwise. For the distance rule, the distances are basically the absolute value of the z-score. Hence $D_1(w) \approx 1.732|w - a|$ and $D_2(w) \approx 0.1732|w|$. If w is from group 1, then w will not be classified very well unless $|a| \geq 10$ or if w is very close to a. In particular, if $a = 0$ then expect nearly all w to be classified to group 2 if w is used to classify the groups. On the other hand, if $a = 0$, then $D_1(w)$ is small for w in group 1 but large for w in group 2. Hence using $z = D_1(w)$ in the distance rule would result in classification with low error rates.

Similarly if group 2 comes from a $N_p(\boldsymbol{0}, 10\boldsymbol{I}_p)$ distribution and group 1 comes from a $N_p(\boldsymbol{\mu}, \boldsymbol{I}_p)$ distribution, the maximum likelihood rule will tend to classify $\boldsymbol{w}$ in group 1 if $\boldsymbol{w}$ is close to $\boldsymbol{\mu}$ and to classify $\boldsymbol{w}$ in group 2 otherwise. The two misclassification error rates should both be low. For the distance rule, the distances D_i have an approximate χ^2_p distribution if $\boldsymbol{w}$ is from group i. If covering ellipsoids from the two groups have little overlap, then the distance rule does well. If $\boldsymbol{\mu} = \boldsymbol{0}$, then expect nearly all of the $\boldsymbol{w}$ to be classified to group 2 with the distance rule, but $D_1(\boldsymbol{w})$ will be small for $\boldsymbol{w}$ from group 1 and large for $\boldsymbol{w}$ from group 2, so using the single predictor $z = D_1(\boldsymbol{w})$ in the distance rule would result in classification with low error rates. More generally, if group 1 has a covering hyperellipsoid that has little overlap with the observations from group 2, using the single predictor $z = D_1(\boldsymbol{w})$ in the distance rule should result in classification with low error rates even if the observations from group 2 do not fall in an hyperellipsoidal region.

Now suppose the G groups come from the same family of elliptically contoured $EC(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, g)$ distributions where g is a continuous decreasing function that does not depend on j for $j = 1, ..., G$. For example, the jth distribution could have $\boldsymbol{w} \sim N_p(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$. Using Equation (1.16), $\log(f_j(\boldsymbol{w})) =$

$$\log(k_p) - \frac{1}{2}\log(|\boldsymbol{\Sigma}_j|) + \log(g[(\boldsymbol{w} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{w} - \boldsymbol{\mu}_j)]) =$$

$$\log(k_p) - \frac{1}{2}\log(|\boldsymbol{\Sigma}_j|) + \log(g[D^2_{\boldsymbol{w}}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)]).$$

Hence the maximum likelihood rule leads to the quadratic rule if the k groups have $N_p(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ distributions where $g(z) = \exp(-z/2)$, and the maximum likelihood rule leads to the distance rule if the groups have dispersion matrices

that have the same determinant: $\det(\boldsymbol{\Sigma}_j) = |\boldsymbol{\Sigma}_j| \equiv |\boldsymbol{\Sigma}|$ for $j = 1, ..., k$. This result is true since then maximizing $f_j(\boldsymbol{w})$ is equivalent to minimizing $D^2_{\boldsymbol{w}}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$. Plugging in estimators leads to the distance rule. The same determinant assumption is a much weaker assumption than that of equal dispersion matrices. For example, let $c_X \boldsymbol{\Sigma}_j$ be the covariance matrix of $\boldsymbol{x}$, and let $\boldsymbol{\Gamma}_j$ be an orthogonal matrix. Then $\boldsymbol{y} = \boldsymbol{\Gamma}_j \boldsymbol{x}$ corresponds to rotating $\boldsymbol{x}$, and $c_X \boldsymbol{\Gamma}_j \boldsymbol{\Sigma}_j \boldsymbol{\Gamma}_j^T$ is the covariance matrix of $\boldsymbol{y}$ with $|\mathrm{Cov}(\boldsymbol{x})| = |\mathrm{Cov}(\boldsymbol{y})|$.

Note that if the G groups come from the same family of elliptically contoured $EC(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, g)$ distributions with nonsingular covariance matrices $c_X \boldsymbol{\Sigma}_j$, then $D^2_{\boldsymbol{w}}(\overline{\boldsymbol{x}}_j, \boldsymbol{S}_j)$ is a consistent estimator of $D^2_{\boldsymbol{w}}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)/c_X$. Hence the distance rule using $(\overline{\boldsymbol{x}}_j, \boldsymbol{S}_j)$ is a maximum likelihood rule if the $\boldsymbol{\Sigma}_j$ have the same determinant. The constant c_X is given below Equation (1.19).

Now $D^2_{\boldsymbol{w}}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \boldsymbol{w}^T \boldsymbol{\Sigma}_j^{-1} \boldsymbol{w} - \boldsymbol{w}^T \boldsymbol{\Sigma}_j^{-1} \boldsymbol{\mu}_j - \boldsymbol{\mu}_j^T \boldsymbol{\Sigma}_j^{-1} \boldsymbol{w} + \boldsymbol{\mu}_j^T \boldsymbol{\Sigma}_j^{-1} \boldsymbol{\mu}_j = \boldsymbol{w}^T \boldsymbol{\Sigma}_j^{-1} \boldsymbol{w} - 2\boldsymbol{\mu}_j^T \boldsymbol{\Sigma}_j^{-1} \boldsymbol{w} + \boldsymbol{\mu}_j^T \boldsymbol{\Sigma}_j^{-1} \boldsymbol{\mu}_j = \boldsymbol{w}^T \boldsymbol{\Sigma}_j^{-1} \boldsymbol{w} + \boldsymbol{\mu}_j^T \boldsymbol{\Sigma}_j^{-1} (-2\boldsymbol{w} + \boldsymbol{\mu}_j)$. Hence if $\boldsymbol{\Sigma}_j \equiv \boldsymbol{\Sigma}$ for $j = 1, ..., G$, then we want to minimize $\boldsymbol{\mu}_j^T \boldsymbol{\Sigma}^{-1} (-2\boldsymbol{w} + \boldsymbol{\mu}_j)$ or maximize $\boldsymbol{\mu}_j^T \boldsymbol{\Sigma}^{-1} (2\boldsymbol{w} - \boldsymbol{\mu}_j)$. Plugging in estimators leads to the linear discriminant rule.

The maximum likelihood rule is robust to nonnormality, but it is difficult to estimate $\hat{f}_j(\boldsymbol{w})$ if $p > 2$. The linear discriminant rule and distance rule are robust to nonnormality, as is the logistic regression discriminant rule if $G = 2$. The distance rule tends to work well when the ellipsoidal covering regions of the G groups have little overlap. The distance rule can be very poor if the groups overlap and have very different variability.

Rule of thumb 5.1. It is often useful to use predictor transformations from Section 1.2 to remove nonlinearities from the predictors. The log rule is especially useful for highly skewed predictors. After making transformations, assume that there are $1 \leq k \leq p$ continuous predictors $X_1, ..., X_k$ where no terms like $X_2 = X_1^2$ or $X_3 = X_1 X_2$ are included. If $n_j \geq 10k$ for $j = 1, ..., G$, then make the G DD plots using the k predictors from each group to check for outliers, which could be cases that were incorrectly classified. Then use p predictors which could include squared terms, interactions, and categorical predictors. Try several discriminant rules. For a given rule, the error rates computed using the training data $\boldsymbol{x}_{i,j}$ with known groups give a lower bound on the error rates for the test data $\boldsymbol{w}_i$. That is, the error rates computed on the training data $\boldsymbol{x}_{i,j}$ are optimistic. When the discriminant rule is applied to the m $\boldsymbol{w}_i$ where the groups for the test data $\boldsymbol{w}_i$ are unknown, the error rates will be higher. If equal covariance matrices are assumed, plot $D_i(\overline{\boldsymbol{x}}_j, \boldsymbol{S}_j)$ versus $D_i(\overline{\boldsymbol{x}}_j, \boldsymbol{\Sigma}_{pool})$ for each of the G groups, where the $\boldsymbol{x}_{i,j}$ are used for $i = 1, ..., n_j$. If all of the n_j are large, say $n_j \geq 30p$, then the plotted points should cluster tightly about the identity line in each of the G plots if the assumption of equal covariance matrices is reasonable. The linear discriminant rule has some robustness against the assumption of equal covariance matrices. See Remark 5.3.

5.2.1 Regularized Estimators

A regularized estimator reduces the degrees of freedom d of the estimator. We want $n \geq 10d$, say. Often regularization is done by reducing the number of parameters in the model. For MLR, lasso and ridge regression were regularized if $\lambda > 0$. A covariance matrix of a $p \times 1$ vector $\boldsymbol{x}$ is symmetric with $p + (p-1) + \cdots + 2 + 1 = p(p+1)/2$ parameters. A correlation matrix has $p(p-1)/2$ parameters. We want $n \geq 10p$ for the sample covariance and correlation matrices $\boldsymbol{S}$ and $\boldsymbol{R}$. If $n < 5p$, then these matrices are being overfit: the degrees of freedom is too large for the sample size n.

Hence QDA needs $n_i \geq 10p$ for $i = 1, ..., G$. LDA need $n \geq 10p$ where $\sum_{i=1}^{G} n_i = n$. Hence the pooled covariance matrix can be regarded as a regularized estimator of the $\boldsymbol{\Sigma}_i$. Hence LDA can be regarded as a regularized version of QDA. See Friedman (1989, p. 167). Adding squared terms and interactions to LDA can make LDA perform more like QDA if the $n_i \geq 10p$, but increases the LDA degrees of freedom.

For QDA, Friedman (1989) suggested using $\hat{\boldsymbol{\Sigma}}(\lambda) = \boldsymbol{S}_k(\lambda)/n_k(\lambda)$ where $\boldsymbol{S}_k(\lambda) = (1-\lambda)\boldsymbol{S}_k + \lambda \boldsymbol{S}_{pool}$, $0 \leq \lambda \leq 1$, and $n_k(\lambda) = (1-\lambda)n_k + \lambda n$. Then $\lambda = 0$ gives QDA, while $\lambda = 1$ gives LDA if the covariance matrices are computed using slightly different divisors such as n_k instead of $n_k - 1$. This regularized QDA method needs n large enough so LDA is useful with $\boldsymbol{S}_{pool}$. If further regularization is needed and $0 \leq \gamma \leq 1$, then use

$$\boldsymbol{S}_k(\lambda, \gamma) = (1-\lambda)\boldsymbol{S}_k(\lambda) + \frac{\gamma}{p} tr[\boldsymbol{S}_k(\lambda)]\boldsymbol{I}_p.$$

If $n < 5p$, the LDA should not be used with $\boldsymbol{S}_{pool}$, and more regularization is needed. An extreme amount of regularization would replace $\boldsymbol{S}_{pool}$ by the identity matrix $\boldsymbol{I}_p$. Hopefully better estimators are discussed in Chapter 6.

5.3 LR

Definition 5.6. Assume that $G = 2$ and that there is a group 0 and a group 1. Let $\rho(\boldsymbol{w}) = P(\boldsymbol{w} \in$ group 1). Let $\hat{\rho}(\boldsymbol{w})$ be the logistic regression (LR) estimate of $\rho(\boldsymbol{w})$. The *logistic regression discriminant rule* allocates $\boldsymbol{w}$ to group 1 if $\hat{\rho}(\boldsymbol{w}) \geq 0.5$ and allocates $\boldsymbol{w}$ to group 0 if $\hat{\rho}(\boldsymbol{w}) < 0.5$. The training data for logistic regression are cases $(\boldsymbol{x}_i, Y_i)$ where $Y_i = j$ if the ith case is in group j for $j = 0, 1$ and $i = 1, ..., n$. Logistic regression produces an *estimated sufficient predictor* $ESP = \hat{\alpha} + \hat{\boldsymbol{\beta}}^T \boldsymbol{x}$. Then

$$\hat{\rho}(\boldsymbol{x}) = \frac{e^{ESP}}{1 + e^{ESP}} = \frac{\exp(\hat{\alpha} + \hat{\boldsymbol{\beta}}^T \boldsymbol{x})}{1 + \exp(\hat{\alpha} + \hat{\boldsymbol{\beta}}^T \boldsymbol{x})}.$$

See Section 4.3 for more on logistic regression. The response plot is an important tool for visualizing the logistic regression.

An extension of the above binary logistic regression model uses

$$\hat{\rho}(\boldsymbol{w}) = \frac{e^{\hat{h}(\boldsymbol{w})}}{1 + e^{\hat{h}(\boldsymbol{w})}},$$

and will be discussed below after some notation. Note that $\hat{h}(\boldsymbol{w}) > 0$ corresponds to $\hat{\rho}(\boldsymbol{w}) > 0.5$ while $\hat{h}(\boldsymbol{w}) < 0$ corresponds to $\hat{\rho}(\boldsymbol{w}) < 0.5$. LR uses $\hat{h}(\boldsymbol{w}) = ESP$ and the binary logistic GAM defined in Definition 5.7 uses $\hat{h}(\boldsymbol{w}) = ESP = EAP$. These two methods are robust to nonnormality and are special cases of 1D regression. See Definition 1.2.

Definition 5.7. Let $\rho(w) = \exp(w)/[1 + \exp(w)]$.

a) For the *binary logistic GLM*, $Y_1, ..., Y_n$ are independent with $Y|SP \sim binomial(1, \rho(SP))$ where $\rho(SP) = P(Y = 1|SP)$. This model has $E(Y|SP) = \rho(SP)$ and $V(Y|SP) = \rho(SP)(1 - \rho(SP))$.

b) For the *binary logistic GAM*, $Y_1, ..., Y_n$ are independent with $Y|AP \sim$ binomial$(1, \rho(\text{AP}))$ where $\rho(AP) = P(Y = 1|AP)$. This model has $E(Y|AP) = \rho(AP)$ and $V(Y|AP) = \rho(AP)(1-\rho(AP))$. The response plot and discriminant rule are similar to those of Definition 5.6, and the EAP–response plot adds the estimated mean function $\rho(EAP)$ and a step function to the plot. The *logistic GAM discriminant rule* allocates $\boldsymbol{w}$ to group 1 if $\hat{\rho}(\boldsymbol{w}) \geq 0.5$ and allocates $\boldsymbol{w}$ to group 0 if $\hat{\rho}(\boldsymbol{w}) < 0.5$ where

$$\hat{\rho}(\boldsymbol{w}) = \frac{e^{EAP}}{1 + e^{EAP}}$$

and $EAP = \hat{\alpha} + \sum_{j=1}^{p} \hat{S}_j(\boldsymbol{w}_j)$.

Lasso for binomial logistic regression can be used as in Section 4.6.2. Changing the 10-fold CV criterion to classification error might be useful. For this data from Section 4.6.2, the default deviance criterion had moderate overfit and gave a better response plot than the classification error criterion, which has severe underfit. Compare the following R code to the code in Section 4.6.2.

```
set.seed(1976)  #Binary regression
library(glmnet)
n<-100
m<-1 #binary regression
q <- 100 #100 nontrivial predictors, 95 inactive
k <- 5 #k_S = 5 population active predictors
y <- 1:n
mv <- m + 0 * y
```

```
vars <- 1:q
beta <- 0 * 1:q
beta[1:k] <- beta[1:k] + 1
beta
alpha <- 0
x <- matrix(rnorm(n * q), nrow = n, ncol = q)
SP <- alpha + x[,1:k] %*% beta[1:k]
pv <- exp(SP)/(1 + exp(SP))
y <- rbinom(n,size=m,prob=pv)
y
out<-cv.glmnet(x,y,family="binomial",type.measure="class")
lam <- out$lambda.min
bhat <- as.vector(predict(out,type="coefficients",s=lam))
ahat <- bhat[1] #alphahat
bhat<-bhat[-1]
vin <- vars[bhat!=0]
vin #underfit compared to the default in Section 4.6.2
[1] 2 4
ind <- as.data.frame(cbind(y,x[,vin])) #relaxed lasso GLM
tem <- glm(y~.,family="binomial",data=ind)
tem$coef
lrplot3(tem=tem,x=x[,vin]) #binary response plot
```

5.4 KNN

The K-nearest neighbors (KNN) method identifies the K cases in the training data that are closest to $\boldsymbol{w}$. Suppose m_j of the K cases are from group j. Then the KNN estimate of $p_j(\boldsymbol{w}) = P(Y = j|\boldsymbol{W} = \boldsymbol{w}) = P(\boldsymbol{w}$ is from the jth group) is $\hat{p}_j(\boldsymbol{w}) = m_j/K$. (Actually $m_j/K \approx cp_j(\boldsymbol{w})$ so $m_j/m_k \approx p_j(\boldsymbol{w})/p_k(\boldsymbol{w})$. See the end of this section.) Applying the Bayesian discriminant rule to the $\hat{p}_j(\boldsymbol{w})$ gives the KNN discriminant rule.

Definition 5.8. The K-nearest neighbors (KNN) discriminant rule allocates $\boldsymbol{w}$ to group a if m_a maximizes m_j for $j = 1, ..., G$.

A couple of examples will be useful. When $K = 1$, find the case in the training data closest to $\boldsymbol{w}$. If that training data case is from group j then allocate $\boldsymbol{w}$ to group j. Suppose n_j is the largest n_k for $k = 1, ..., G$. Hence group j is the group with the most training data cases. Then if $K = n$, $\boldsymbol{w}$ is always allocated to group j. The $K = n$ rule is bad. The $K = 1$ rule is surprisingly good, but tends to have low bias and high variability. Generally values of $K > 1$ will have smaller test error rates.

For KNN and other discriminant analysis rules, it is often useful to standardize the data so that all variables have a sample mean of 0 and sample

standard deviation of 1. The `scale` function in R can be used to standardize data. The test data is standardized using means and SDs from the training data. The jth variable from $\boldsymbol{x}_i$ uses $(x_{ij} - \overline{x}_j)/S_j$. Hence the jth variable from a text case $\boldsymbol{w}$ would use $(w_j - \overline{x}_j)/S_j$. Here $\overline{x}_j$ and S_j are the sample mean and standard deviation of the jth variable using all of the training data (so group is ignored).

To see why KNN might be reasonable, let D_ϵ be a hypersphere of radius ϵ centered at $\boldsymbol{w}$. Since the pdf $f_j(\boldsymbol{x})$ is continuous, there exists $\epsilon > 0$ small enough such that $f_j(\boldsymbol{x}) \approx f_j(\boldsymbol{w})$ for all $\boldsymbol{x} \in D_\epsilon$ and for each $j = 1, ..., G$. If $\boldsymbol{z}$ is a random vector from a distribution with pdf $f_j(\boldsymbol{x})$, then $P_j(\boldsymbol{z} \in D_\epsilon) =$

$$\int_{D_\epsilon} f_j(\boldsymbol{x})d\boldsymbol{x} \approx f_j(\boldsymbol{w}) \int_{D_\epsilon} 1d\boldsymbol{x} = f_j(\boldsymbol{w})Vol(D_\epsilon) = f_j(\boldsymbol{w})\frac{2\pi^{p/2}}{p\Gamma(p/2)}\epsilon^p.$$

Here P_j denotes the probability when the distribution has pdf $f_j(\boldsymbol{x})$.

If for $i = 1, ..., n$, the $\boldsymbol{z}_i$ are iid from a distribution with pdf $f_j(\boldsymbol{x})$, ϵ is fixed, and if $f_j(\boldsymbol{w}) > 0$, then the number of $\boldsymbol{z}_i$ in D_ϵ is proportional to n. Hence if the number of $\boldsymbol{z}_i$ in D_ϵ is proportional to n^δ with $0 < \delta < 1$, then $\epsilon \to 0$. So if $K/n \to 0$ in KNN, then the hypersphere containing the K cases has radius $\epsilon \to 0$ as $n \to \infty$. Hence the above approximations will be valid for large n. Note that if $p = 1$, then D_ϵ is the line segment $(w - \epsilon, w + \epsilon)$ and $Vol(D_\epsilon) = 2\epsilon =$ length of the line segment. If $p = 2$, then D_ϵ is the circle of radius ϵ centered at $\boldsymbol{w}$ and $Vol(D_\epsilon) = \pi\epsilon^2 =$ the area of the circle. If $p = 3$, then D_ϵ is the sphere of radius ϵ centered at $\boldsymbol{w}$ and $Vol(D_\epsilon) = 4\pi\epsilon^3/3 =$ the volume of the sphere.

Now suppose that the training data $\boldsymbol{x}_{1,1}, ..., \boldsymbol{x}_{n_G,G}$ is a random sample from the G populations so that $n_j/n \xrightarrow{P} \pi_j$ as $n \to \infty$ for $j = 1, ..., G$. Then for ϵ small and K large, $m_j/K \approx$

$$P(\boldsymbol{W} \in D_\epsilon, Y = j) = P(\boldsymbol{W} \in D_\epsilon | Y = j)P(Y = j) \approx \pi_j f_j(\boldsymbol{w})Vol(D_\epsilon).$$

Now $P(\boldsymbol{W} \in D_\epsilon) = \sum_{j=1}^G P(\boldsymbol{W} \in D_\epsilon, Y = j) =$
$\sum_{j=1}^G P(\boldsymbol{W} \in D_\epsilon | Y = j)P(Y = j)$ since the sets $\{Y = j\}$ form a disjoint partition. Hence

$$P(Y = k|\boldsymbol{W} \in D_\epsilon) = \frac{P(Y = k, \boldsymbol{W} \in D_\epsilon)}{P(\boldsymbol{W} \in D_\epsilon)} = \frac{P(\boldsymbol{W} \in D_\epsilon)|Y = k)P(Y = k)}{P(\boldsymbol{W} \in D_\epsilon)}$$

$$\approx \frac{\pi_k f_k(\boldsymbol{w})Vol(D_\epsilon)}{\sum_{j=1}^G \pi_j f_j(\boldsymbol{w})Vol(D_\epsilon)},$$

which is the quantity used by the Bayes classifier since the constant $Vol(D_\epsilon)$ cancels. This argument can also be used to justify Equation (5.1). Since the denominator is a constant, allocating $\boldsymbol{w}$ to group a with the largest m_a/K,

or equivalently with the largest m_a, approximates the Bayes classifier if n is very large, K is large, and ϵ is very small.

This approximation likely needs unrealistically large n, especially if p is large and $\boldsymbol{w}$ is in a region where there is a lot of group overlap. However, KNN often works well in practice. Silverman (1986, pp. 96-100) also discusses using KNN to find an estimator $\hat{f}(\boldsymbol{w})$ of $f(\boldsymbol{w})$.

As claimed above Definition 5.8, note, for large K and small ϵ, that

$$m_j/K \approx P(\boldsymbol{W} \in D_\epsilon, Y = j) = P(Y = j|\boldsymbol{W} \in D_\epsilon)P(\boldsymbol{W} \in D_\epsilon) \approx$$

$$cP(Y = j|\boldsymbol{W} = \boldsymbol{w}) = cp_k(\boldsymbol{w})$$

where $c = P(\boldsymbol{W} \in D_\epsilon)$.

5.5 Some Matrix Optimization Results

The following results will be useful for multivariate analysis including Fisher's discriminant analysis. Let $\boldsymbol{B} > 0$ denote that $\boldsymbol{B}$ is a positive definite matrix. The *generalized eigenvalue problem* finds eigenvalue eigenvector pairs $(\lambda, \boldsymbol{g})$ such that $\boldsymbol{C}^{-1}\boldsymbol{A}\boldsymbol{g} = \lambda\boldsymbol{g}$ which are also solutions to the equation $\boldsymbol{A}\boldsymbol{g} = \lambda\boldsymbol{C}\boldsymbol{g}$. Then the pairs are used to maximize or minimize the *Rayleigh quotient* $\frac{\boldsymbol{a}^T\boldsymbol{A}\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{C}\boldsymbol{a}}$. Results from linear algebra show that if $\boldsymbol{C} > 0$ and $\boldsymbol{A}$ are both symmetric, then the p eigenvalues of $\boldsymbol{C}^{-1}\boldsymbol{A}$ are real, and the number of nonzero eigenvalues of $\boldsymbol{C}^{-1}\boldsymbol{A}$ is equal to $\text{rank}(\boldsymbol{C}^{-1}\boldsymbol{A}) = \text{rank}(\boldsymbol{A})$. Note that if $\boldsymbol{a}_1 = c_1\boldsymbol{g}_1$ is the maximizer and $\boldsymbol{a}_p = c_p\boldsymbol{g}_p$ is the minimizer of the Rayleigh quotient for any nonzero constants c_1 and c_p, then there is a vector $\boldsymbol{\beta}$ that is the maximizer or minimizer such that $\|\boldsymbol{\beta}\| = 1$.

Theorem 5.1. Let $\boldsymbol{B} > 0$ be a $p \times p$ symmetric matrix with eigenvalue eigenvector pairs $(\lambda_1, \boldsymbol{e}_1), ..., (\lambda_p, \boldsymbol{e}_p)$ where $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_p > 0$ and the orthonormal eigenvectors satisfy $\boldsymbol{e}_i^T\boldsymbol{e}_i = 1$ while $\boldsymbol{e}_i^T\boldsymbol{e}_j = 0$ for $i \neq j$. Let $\boldsymbol{d}$ be a given $p \times 1$ vector and let $\boldsymbol{a}$ be an arbitrary nonzero $p \times 1$ vector. See Johnson and Wichern (1988, pp. 64-65, 184).

a) $\max_{\boldsymbol{a} \neq \boldsymbol{0}} \frac{\boldsymbol{a}^T\boldsymbol{d}\boldsymbol{d}^T\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{B}\boldsymbol{a}} = \boldsymbol{d}^T\boldsymbol{B}^{-1}\boldsymbol{d}$ where the max is attained for $\boldsymbol{a} = c\boldsymbol{B}^{-1}\boldsymbol{d}$ for any constant $c \neq 0$. Note that the numerator $= (\boldsymbol{a}^T\boldsymbol{d})^2$.

b) $\max_{\boldsymbol{a} \neq \boldsymbol{0}} \frac{\boldsymbol{a}^T\boldsymbol{B}\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{a}} = \max_{\|\boldsymbol{a}\|=1} \boldsymbol{a}^T\boldsymbol{B}\boldsymbol{a} = \lambda_1$ where the max is attained for $\boldsymbol{a} = \boldsymbol{e}_1$.

c) $\min_{\boldsymbol{a} \neq \boldsymbol{0}} \frac{\boldsymbol{a}^T\boldsymbol{B}\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{a}} = \min_{\|\boldsymbol{a}\|=1} \boldsymbol{a}^T\boldsymbol{B}\boldsymbol{a} = \lambda_p$ where the min is attained for $\boldsymbol{a} = \boldsymbol{e}_p$.

d) $\max_{\boldsymbol{a} \perp \boldsymbol{e}_1,\ldots,\boldsymbol{e}_k} \frac{\boldsymbol{a}^T\boldsymbol{B}\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{a}} = \max_{\|\boldsymbol{a}\|=1, \boldsymbol{a} \perp \boldsymbol{e}_1,\ldots,\boldsymbol{e}_k} \boldsymbol{a}^T\boldsymbol{B}\boldsymbol{a} = \lambda_{k+1}$ where the max is attained for $\boldsymbol{a} = \boldsymbol{e}_{k+1}$ for $k = 1, 2, ..., p-1$.

e) Let $(\overline{\boldsymbol{x}}, \boldsymbol{S})$ be the observed sample mean and sample covariance matrix where $\boldsymbol{S} > 0$. Then $\max_{\boldsymbol{a} \neq \boldsymbol{0}} \frac{n\boldsymbol{a}^T(\overline{\boldsymbol{x}} - \boldsymbol{\mu})(\overline{\boldsymbol{x}} - \boldsymbol{\mu})^T\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{S}\boldsymbol{a}} = n(\overline{\boldsymbol{x}} - \boldsymbol{\mu})^T\boldsymbol{S}^{-1}(\overline{\boldsymbol{x}} - \boldsymbol{\mu}) = T^2$ where the max is attained for $\boldsymbol{a} = c\boldsymbol{S}^{-1}(\overline{\boldsymbol{x}} - \boldsymbol{\mu})$ for any constant $c \neq 0$.

f) Let $\boldsymbol{A}$ be a $p \times p$ symmetric matrix. Let $\boldsymbol{C} > 0$ be a $p \times p$ symmetric matrix. Then $\max_{\boldsymbol{a} \neq \boldsymbol{0}} \frac{\boldsymbol{a}^T\boldsymbol{A}\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{C}\boldsymbol{a}} = \lambda_1(\boldsymbol{C}^{-1}\boldsymbol{A})$, the largest eigenvalue of $\boldsymbol{C}^{-1}\boldsymbol{A}$. The value of $\boldsymbol{a}$ that achieves the max is the eigenvector $\boldsymbol{g}_1$ of $\boldsymbol{C}^{-1}\boldsymbol{A}$ corresponding to $\lambda_1(\boldsymbol{C}^{-1}\boldsymbol{A})$. Similarly $\min_{\boldsymbol{a} \neq \boldsymbol{0}} \frac{\boldsymbol{a}^T\boldsymbol{A}\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{C}\boldsymbol{a}} = \lambda_p(\boldsymbol{C}^{-1}\boldsymbol{A})$, the smallest eigenvalue of $\boldsymbol{C}^{-1}\boldsymbol{A}$. The value of $\boldsymbol{a}$ that achieves the min is the eigenvector $\boldsymbol{g}_p$ of $\boldsymbol{C}^{-1}\boldsymbol{A}$ corresponding to $\lambda_p(\boldsymbol{C}^{-1}\boldsymbol{A})$.

Proof Sketch. For a), note that $\text{rank}(\boldsymbol{C}^{-1}\boldsymbol{A}) = 1$, where $\boldsymbol{C} = \boldsymbol{B}$ and $\boldsymbol{A} = \boldsymbol{d}\boldsymbol{d}^T$, since $\text{rank}(\boldsymbol{C}^{-1}\boldsymbol{A}) = \text{rank}(\boldsymbol{A}) = \text{rank}(\boldsymbol{d}) = 1$. Hence $\boldsymbol{C}^{-1}\boldsymbol{A}$ has one nonzero eigenvalue eigenvector pair $(\lambda_1, \boldsymbol{g}_1)$. Since

$$(\lambda_1 = \boldsymbol{d}^T\boldsymbol{B}^{-1}\boldsymbol{d}, \boldsymbol{g}_1 = \boldsymbol{B}^{-1}\boldsymbol{d})$$

is a nonzero eigenvalue eigenvector pair for $\boldsymbol{C}^{-1}\boldsymbol{A}$, and $\lambda_1 > 0$, the result follows by f).

Note that b) and c) are special cases of f) with $\boldsymbol{A} = \boldsymbol{B}$ and $\boldsymbol{C} = \boldsymbol{I}$.

Note that e) is a special case of a) with $\boldsymbol{d} = (\overline{\boldsymbol{x}} - \boldsymbol{\mu})$ and $\boldsymbol{B} = \boldsymbol{S}$.

(Also note that $(\lambda_1 = (\overline{\boldsymbol{x}} - \boldsymbol{\mu})^T\boldsymbol{S}^{-1}(\overline{\boldsymbol{x}} - \boldsymbol{\mu}), \boldsymbol{g}_1 = \boldsymbol{S}^{-1}(\overline{\boldsymbol{x}} - \boldsymbol{\mu}))$ is a nonzero eigenvalue eigenvector pair for the rank 1 matrix $\boldsymbol{C}^{-1}\boldsymbol{A}$ where $\boldsymbol{C} = \boldsymbol{S}$ and $\boldsymbol{A} = (\overline{\boldsymbol{x}} - \boldsymbol{\mu})(\overline{\boldsymbol{x}} - \boldsymbol{\mu})^T$.)

For f), see Mardia et al. (1979, p. 480). □

Suppose $\boldsymbol{A} > 0$ and $\boldsymbol{C} > 0$ are $p \times p$ symmetric matrices, and let $\boldsymbol{C}^{-1}\boldsymbol{A}\boldsymbol{a} = \lambda\boldsymbol{a}$. Then $\boldsymbol{A}\boldsymbol{a} = \lambda\boldsymbol{C}\boldsymbol{a}$, or $\boldsymbol{A}^{-1}\boldsymbol{C}\boldsymbol{a} = \frac{1}{\lambda}\boldsymbol{a}$. Hence if $(\lambda_i(\boldsymbol{C}^{-1}\boldsymbol{A}), \boldsymbol{a})$ are eigenvalue eigenvector pairs of $\boldsymbol{C}^{-1}\boldsymbol{A}$, then $\left(\lambda_i(\boldsymbol{A}^{-1}\boldsymbol{C}) = \frac{1}{\lambda_i(\boldsymbol{C}^{-1}\boldsymbol{A})}, \boldsymbol{a}\right)$ are eigenvalue eigenvector pairs of $\boldsymbol{A}^{-1}\boldsymbol{C}$. Thus we can maximize $\frac{\boldsymbol{a}^T\boldsymbol{A}\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{C}\boldsymbol{a}}$ with the eigenvector $\boldsymbol{a}$ corresponding to the smallest eigenvalue of $\boldsymbol{A}^{-1}\boldsymbol{C}$, and minimize $\frac{\boldsymbol{a}^T\boldsymbol{A}\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{C}\boldsymbol{a}}$ with the eigenvector $\boldsymbol{a}$ corresponding to the largest eigenvalue of $\boldsymbol{A}^{-1}\boldsymbol{C}$.

Remark 5.1. Suppose $\boldsymbol{A}$ and $\boldsymbol{C}$ are symmetric $p \times p$ matrices, $\boldsymbol{A} > 0$, $\boldsymbol{C}$ is singular, and it is desired to make $\frac{\boldsymbol{a}^T\boldsymbol{A}\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{C}\boldsymbol{a}}$ large but finite. Hence

$\frac{\boldsymbol{a}^T\boldsymbol{C}\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{A}\boldsymbol{a}}$ should be made small but nonzero. The above result suggests that the eigenvector $\boldsymbol{a}$ corresponding to the smallest nonzero eigenvalue of $\boldsymbol{A}^{-1}\boldsymbol{C}$ may be useful. Similarly, suppose it is desired to make $\frac{\boldsymbol{a}^T\boldsymbol{A}\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{C}\boldsymbol{a}}$ small but nonzero. Hence $\frac{\boldsymbol{a}^T\boldsymbol{C}\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{A}\boldsymbol{a}}$ should be made large but finite. Then the eigenvector $\boldsymbol{a}$ corresponding to the largest eigenvalue of $\boldsymbol{A}^{-1}\boldsymbol{C}$ may be useful.

5.6 FDA

The FDA method of discriminant analysis, a special case of the generalized eigenvalue problem, finds eigenvalue eigenvector pairs so that the $\hat{\boldsymbol{e}}_1^T\boldsymbol{x}_{ij}$ have low variability in each group, but the variability of the $\hat{\boldsymbol{e}}_1^T\boldsymbol{x}_{ij}$ between groups is large. More precisely, let $\hat{\boldsymbol{W}}$ be a $p \times p$ dispersion matrix used to measure variability within groups and let $\hat{\boldsymbol{B}}$ be a $p \times p$ symmetric matrix used to measure variability between classes. Let the eigenvalue eigenvector pairs of a matrix $\hat{\boldsymbol{W}}^{-1}\hat{\boldsymbol{B}}$ be $(\hat{\lambda}_1, \hat{\boldsymbol{e}}_1), ..., (\hat{\lambda}_p, \hat{\boldsymbol{e}}_p)$ where $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \cdots \geq \hat{\lambda}_p$. Then from Theorem 5.1 f), $\max_{\boldsymbol{a} \neq \boldsymbol{0}} \frac{\boldsymbol{a}^T\hat{\boldsymbol{B}}\boldsymbol{a}}{\boldsymbol{a}^T\hat{\boldsymbol{W}}\boldsymbol{a}} = \hat{\lambda}_1$, the largest eigenvalue of $\hat{\boldsymbol{W}}^{-1}\hat{\boldsymbol{B}}$. The value of $\boldsymbol{a}$ that achieves the max is the eigenvector $\hat{\boldsymbol{e}}_1$. Then $\hat{\boldsymbol{e}}_2$ will achieve the max among all unit vectors orthogonal to $\hat{\boldsymbol{e}}_1$. Similarly, $\hat{\boldsymbol{e}}_3$ will achieve the max among all unit vectors orthogonal to $\hat{\boldsymbol{e}}_1$ and $\hat{\boldsymbol{e}}_2$, et cetera.

Many choices of $\hat{\boldsymbol{W}}$ have been suggested. Typically assume $\text{rank}(\hat{\boldsymbol{W}}) = p$ and $\text{rank}(\hat{\boldsymbol{B}}) = \min(p, G-1)$. Let $q \leq \min(p, G-1)$ be the number of nonzero eigenvalues $\hat{\lambda}_i$ of $\hat{\boldsymbol{W}}^{-1}\hat{\boldsymbol{B}}$. Let $(T_i, \boldsymbol{C}_i)$ be an estimator of multivariate location and dispersion for the ith group. Let $\overline{T} = \frac{1}{G}\sum_{i=1}^{G} T_i$. Let $\hat{\boldsymbol{B}}_T = \sum_{i=1}^{G}(T_i - \overline{T})(T_i - \overline{T})^T$. Note that $\hat{\boldsymbol{B}}_T/(G-1)$ is the sample covariance matrix of the $T_1, ..., T_G$. Let $\hat{\boldsymbol{W}}_T = \sum_{i=1}^{G} \boldsymbol{C}_i$. Typically $(T_i, \boldsymbol{C}_i) = (\overline{\boldsymbol{x}}_i, \boldsymbol{S}_i)$ is used where the notation $\overline{T} = \overline{\overline{\boldsymbol{x}}}$ is used. Let $\hat{\boldsymbol{B}}_B = \sum_{i=1}^{G} \hat{\pi}_i(T_i - \overline{T})(T_i - \overline{T})^T$, and $\hat{\boldsymbol{W}}_B = \sum_{i=1}^{G} \hat{\pi}_i \boldsymbol{C}_i$. Let $\hat{\boldsymbol{W}}_L = G\hat{\boldsymbol{\Sigma}}_{pool}$. See Equation (5.3). Let $\boldsymbol{A} = (a_{ij})$ be a $p \times p$ matrix, and let $diag(\boldsymbol{A}) = diag(a_{11}, ..., a_{pp})$ be the diagonal matrix with the a_{ii} along the diagonal. Let $\hat{\boldsymbol{W}}_D = diag(\hat{\boldsymbol{W}}_A)$ for any previously defined $\hat{\boldsymbol{W}}_A$, e.g. $A = T$. Then $\hat{\boldsymbol{W}}_D$ is nonsingular if all $w_{ii} > 0$ even if $\hat{\boldsymbol{W}}_A = (w_{ij})$ is singular. Sometimes $\overline{T}_B = \sum_{i=1} \hat{\pi}_i T_i$ is used instead of $\overline{T}$. The rule may also use $\hat{\boldsymbol{B}} = c_1\hat{\boldsymbol{B}}_A$ and $\hat{\boldsymbol{W}} = c_2\hat{\boldsymbol{W}}_A$ for positive constants c_1 and c_2, e.g. $c_1 = 1/(G-1)$ and $c_2 = 1/(n-G)$.

The FDA rule finds $\hat{\boldsymbol{e}}_1$ and summarizes the group by the linear combination $\hat{\boldsymbol{e}}_1^T T_i$. Then FDA allocates $\boldsymbol{w}$ to the group a for which $\hat{\boldsymbol{e}}_1^T\boldsymbol{w}$ is closest to $\hat{\boldsymbol{e}}_1^T T_a$. (We can view $\hat{\boldsymbol{e}}_1^T T_i$ as a summary of the n_i linear combinations of

the predictors $\hat{\boldsymbol{e}}_1^T \boldsymbol{x}_{ij}$ in the ith group where $j = 1, ..., n_i$.) The FDA method should work well if the within group variability is small and the between group variability is large.

Definition 5.9. For *Fisher's discriminant analysis* (FDA), the *FDA discriminant rule* allocates $\boldsymbol{w}$ to group a that minimizes $|\hat{\boldsymbol{e}}_1^T \boldsymbol{w} - \hat{\boldsymbol{e}}_1^T T_i|$ for $i = 1, ..., G$.

Remark 5.2. a) Often it is suggested to use PCA for DA: find D such that the first D principal components explain at least 95% of the variance. Then use the $D \leq \min(n, p)$ principal components as the variables. The problem with this idea is that principal components are used to explain the structure of the dispersion matrix of the data, not to be linear combinations of the data that are good for DA. Using the J linear combinations from FDA such that

$$\sum_{i=1}^{J} \hat{\lambda}_i / \sum_{i=1}^{p} \hat{\lambda}_i \geq 0.95$$

might be a better choice for DA, especially if the number of nonzero eigenvalues q is not too small.

b) Often DA rules from the other FDA eigenvectors simply replace $\hat{\boldsymbol{e}}_1$ with $\hat{\boldsymbol{e}}_j$. It might be better to consider J rules such that $(\hat{\boldsymbol{e}}_1^T \boldsymbol{w}, ..., \hat{\boldsymbol{e}}_k^T \boldsymbol{w})^T$ is closest to $(\hat{\boldsymbol{e}}_1^T T_a, ..., \hat{\boldsymbol{e}}_k^T T_a)^T$ for $k = 1, ..., J$ where $a \in \{1, ..., G\}$ and J is as in Remark 5.2 a). Or let $\hat{\boldsymbol{V}} = [\hat{\boldsymbol{e}}_1 \ \hat{\boldsymbol{e}}_2 \ \cdots \ \hat{\boldsymbol{e}}_q]$. Then allocate $\boldsymbol{w}$ to group a that minimizes $D_j^2(\boldsymbol{w})$ where $D_j^2(\boldsymbol{w}) = (\boldsymbol{w} - T_j)^T \hat{\boldsymbol{V}} \hat{\boldsymbol{V}}^T (\boldsymbol{w} - T_j)^T - 2\log(\hat{\pi}_j)$ where $\hat{\boldsymbol{W}}_B$ and $\hat{\boldsymbol{B}}_B$ are used. See Filzmoser et al. (2006).

c) If $\hat{\boldsymbol{W}}$ is singular and $\hat{\boldsymbol{B}}$ is nonsingular, then the eigenvalue eigenvector pair(s) corresponding to the smallest nonzero eigenvalue(s) of $\hat{\boldsymbol{B}}^{-1} \hat{\boldsymbol{W}}$ may be of interest, as argued below Theorem 5.1.

Following Koch (2014, pp. 120-124) closely, consider the population version of FDA where the ith group has mean and covariance matrix $(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_{\boldsymbol{x}_i})$ for $i = 1, ..., G$ where $\boldsymbol{x}_i$ is a random vector from the population corresponding to the ith group. Let $\overline{\boldsymbol{\mu}} = \frac{1}{G} \sum_{i=1}^{G} \boldsymbol{\mu}_i$, $\boldsymbol{B} = \sum_{i=1}^{G} (\boldsymbol{\mu}_i - \overline{\boldsymbol{\mu}})(\boldsymbol{\mu}_i - \overline{\boldsymbol{\mu}})^T$, and $\boldsymbol{W} = \sum_{i=1}^{G} \boldsymbol{\Sigma}_{\boldsymbol{x}_i}$. Then the *between group variability*

$$b(\boldsymbol{a}) = \boldsymbol{a}^T \boldsymbol{B} \boldsymbol{a} = \sum_{i=1}^{G} |\boldsymbol{a}^T (\boldsymbol{\mu}_i - \overline{\boldsymbol{\mu}})|, \tag{5.4}$$

and the *within group variability* =

$$w(\boldsymbol{a}) = \boldsymbol{a}^T \boldsymbol{W} \boldsymbol{a} = \sum_{i=1}^{G} \boldsymbol{a}^T \boldsymbol{\Sigma}_{\boldsymbol{x}_i} \boldsymbol{a} = \sum_{i=1}^{G} \mathrm{Var}(\boldsymbol{a}^{\mathrm{T}} \boldsymbol{x}_{\mathrm{i}}) \tag{5.5}$$

since $\text{Var}(\boldsymbol{a}^T\boldsymbol{x}_i) = E[(\boldsymbol{a}^T\boldsymbol{x}_i - E(\boldsymbol{a}^T\boldsymbol{x}_i))^2] = E[\boldsymbol{a}^T(\boldsymbol{x}_i - E(\boldsymbol{x}_i))(\boldsymbol{x}_i - E(\boldsymbol{x}_i))^T\boldsymbol{a}]$ $= \boldsymbol{a}^T\boldsymbol{\Sigma}_{\boldsymbol{x}_i}\boldsymbol{a}$. Then

$$\max_{\boldsymbol{a}\neq\boldsymbol{0}} \frac{b(\boldsymbol{a})}{w(\boldsymbol{a})} = \max_{\boldsymbol{a}\neq\boldsymbol{0}} \frac{\boldsymbol{a}^T\boldsymbol{B}\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{W}\boldsymbol{a}}$$

is achieved by $\boldsymbol{a} = \boldsymbol{e}_1$, the eigenvector corresponding to the largest eigenvalue $\lambda_1(\boldsymbol{W}^{-1}\boldsymbol{B})$ of $\boldsymbol{W}^{-1}\boldsymbol{B}$. Hence $b(\boldsymbol{e}_1)$ is large while $w(\boldsymbol{e}_1)$ is small in that the ratio is a max.

FDA approximates Equations (5.4) and (5.5) by using $\hat{\boldsymbol{B}}_T$ and $\hat{\boldsymbol{W}}_T$ with $(T_i, \boldsymbol{C}_i) = (\overline{\boldsymbol{x}}_i, \boldsymbol{S}_i)$. Note that $\boldsymbol{W}/G$ tends not to be a good estimator of dispersion unless the G groups have the same covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{x}_i} = \boldsymbol{\Sigma}_{\boldsymbol{x}}$ for $i = 1, ..., G$, but $w(\boldsymbol{a})$ is a good measure of within group variability even if the $\boldsymbol{\Sigma}_{\boldsymbol{x}_i}$ are not equal. Also, if $\hat{\boldsymbol{W}}_A$ is such that $\boldsymbol{a}^T\hat{\boldsymbol{W}}_A\boldsymbol{a}$ can be made small, then FDA will likely work well with $\hat{\boldsymbol{B}}_T$ and $\hat{\boldsymbol{W}}_A$ if there are no outliers.

Remark 5.3. If $G = 2$, $(T_i, \boldsymbol{C}_i) = (\overline{\boldsymbol{x}}_i, \boldsymbol{S}_i)$, $\hat{\boldsymbol{B}} = \hat{\boldsymbol{B}}_T$, and $\hat{\boldsymbol{W}} = 2\boldsymbol{S}_{pool}$, then LDA and FDA are equivalent. See Koch (2014, p. 129). This result helps explain why LDA works well on so many data sets.

Two special cases are illustrative. First, let $\hat{\boldsymbol{W}} = \boldsymbol{I}_p$ and use $\hat{\boldsymbol{B}}_T$. Then FDA attempts to find a vector $\hat{\boldsymbol{e}}_1$ such that the $\hat{\boldsymbol{e}}_1^T T_i$ are far from $\hat{\boldsymbol{e}}_1^T\overline{T}$. Then find group a such that $\hat{\boldsymbol{e}}_1^T\boldsymbol{w}$ is closer to $\hat{\boldsymbol{e}}_1^T T_a$ than to $\hat{\boldsymbol{e}}_1^T T_i$ for $i \neq a$. Second, consider $G = 2$. Then $\hat{\boldsymbol{B}}_T = (T_1 - T_2)(T_1 - T_2)^T/2$. Using Theorem 5.1a) with $\boldsymbol{d} = (T_1 - T_2)/\sqrt{2}$ shows that $\hat{\boldsymbol{e}}_1 = \dfrac{\hat{\boldsymbol{W}}^{-1}(T_1 - T_2)}{\|\hat{\boldsymbol{W}}^{-1}(T_1 - T_2)\|}$. If the $\hat{\boldsymbol{W}}^{-1}\boldsymbol{x}_{ij}$ are "standardized data," and the $\hat{\boldsymbol{W}}^{-1}T_i$ are standardized centers for $i = 1, 2$, then FDA projects $\boldsymbol{w}$ on the line between the standardized centers and allocates $\boldsymbol{w}$ to the group with the standardized center closest to $\hat{\boldsymbol{e}}_1^T\boldsymbol{w}$.

```
library(MASS) ##Use ?lda. Output for Ex. 5.1.
out <- lda(as.matrix(iris[, 1:4]), iris$Species)
names(out); out; plot(out) #plots LD1 versus LD2
Prior probabilities of groups:
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333
Group means:
           Sep.Len Sep.Wid Pet.Len Pet.Wid
setosa       5.006   3.428   1.462   0.246
versicolor   5.936   2.770   4.260   1.326
virginica    6.588   2.974   5.552   2.026
Coefficients of linear discriminants:
                    LD1         LD2
Sepal.Length  0.8293776  0.02410215
Sepal.Width   1.5344731  2.16452123
```

```
Petal.Length -2.2012117 -0.93192121
Petal.Width  -2.8104603  2.83918785
Proportion of trace:
   LD1    LD2
0.9912 0.0088

gp <- as.integer(iris$Species)
x <- as.matrix(iris[,1:4])            #AER 0.02
out<- lda(x,gp); 1-mean(predict(out,x)$class==gp)
plot(out) #Get numbers in Figure 5.1.
```

Example 5.1. The library *MASS* has a function `lda` that does FDA. The famous iris data set has variables x_1 = sepal length, x_2 = sepal width, x_3 = petal length, and x_4 = petal width. There are three groups corresponding to types of iris: setosa, versicolor, and virginica. The above *R* code performs FDA. Figure 5.1 shows the plot of LD1 = $\hat{\boldsymbol{e}}_1$ versus LD2 = $\hat{\boldsymbol{e}}_2$. Since the proportion of trace for LD2 is small, LD2 is not needed. Note that LD1 separates setosa from the other two types of iris, and versicolor and virginica are nearly separated.

Let $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{e}}_1$ = LD1 be the first eigenvector from FDA. The function `FDAboot` bootstraps $\hat{\boldsymbol{\beta}}$ and gives the nominal 95% shorth CIs. Also shown below is the sample mean vector of the bootstrapped $\hat{\boldsymbol{\beta}}_i^*$ where $i = 1, ..., B = 1000$. The bootstrap is performed by taking samples of size n_i with replacement from each group for $i = 1, ..., G$. Perform FDA on the combined sample to get $\hat{\boldsymbol{\beta}}_j^*$. Since $\hat{\boldsymbol{\beta}}$ is an eigenvector, the bootstrapped eigenvector could estimate $\hat{\boldsymbol{\beta}}$ or $-\hat{\boldsymbol{\beta}}$. Pick a $\hat{\beta}_j$ that is large in magnitude, and see how many times the $\hat{\beta}_j^*$ have the same sign as $\hat{\beta}_j$. Multiply the bootstrap vector by -1 if it has opposite sign. In the output below, all $B = 1000$ bootstrap vectors had $\hat{\beta}_4^* < 0$.

```
#Sample sizes may not be large enough for the
#shorth CI coverage to be close to the nominal 95%.
out<-FDAboot(x,gp)
apply(out$betas,2,mean)
[1]  0.8468  1.5807 -2.2558 -2.9180
sum(out$betas[,4]<0) #all betahat^*
[1] 1000  #estimate betahat, not -betahat
ddplot4(out$betas) #right click Stop
#covers the identity line
out$shorci[[1]]$shorth
[1] 0.3148 1.4634
out$shorci[[2]]$shorth
[1] 0.7745 2.3096
out$shorci[[3]]$shorth
[1] -2.9276 -1.6260
```

```
out$shorci[[4]]$shorth
[1] -3.8609 -1.8875
```

Next, R code is given for robust FDA. The function `getUbig` gets the RMVN set U_i for each group for $i = 1, ..., G$ and combines the sets into one large data set. RMVN is useful when n/p is large. Then RFDA is the classical FDA applied to this cleaned data set. See the output below. Figure 5.2 only uses the cleaned cases since outliers could obscure the plot, and this technique can distort the amount of group overlap.

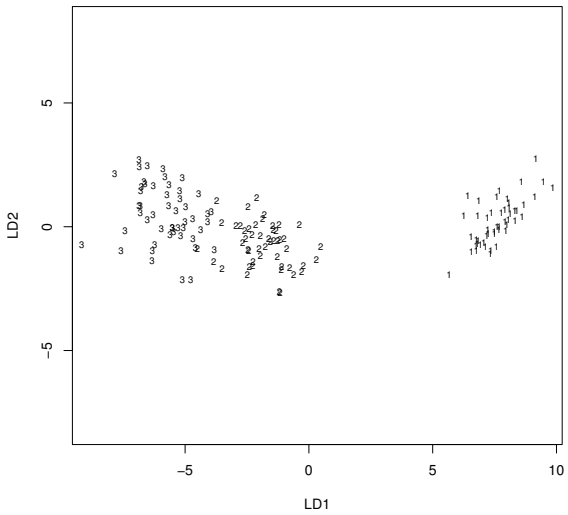

Fig. 5.1 Plot of LD1 versus LD2 for the iris data.

```
tem<-getubig(x,gp) ##Robust FDA
outr<-lda(tem$Ubig,tem$grp)
1-mean(predict(outr,x)$class==gp) #AER 0.03
plot(outr)
outr
Prior probabilities of groups:
        1         2         3
0.3206107 0.3282443 0.3511450
Group means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.026190    3.438095     1.464286   0.2309524
2     5.923256    2.813953     4.234884   1.3093023
3     6.486957    2.950000     5.454348   2.0173913
```

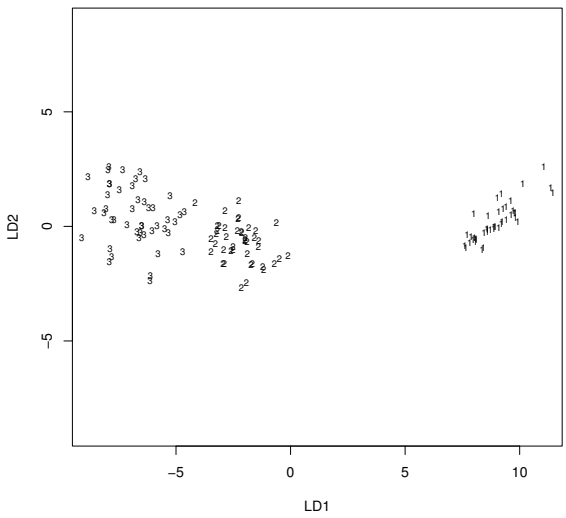

Fig. 5.2 RFDA Plot of LD1 versus LD2 for the iris data.

```
Coefficients of linear discriminants:
                    LD1         LD2
Sepal.Length  0.4281837 -0.06899442
Sepal.Width   2.5221645  2.01270912
Petal.Length -2.3230167 -1.11944258
Petal.Width  -3.2947263  3.25076179
Proportion of trace:
   LD1    LD2
0.9942 0.0058
```

The `covmb2` subset B can be found when $p < n$ or $p \geq n$. See Section 1.3. The function `getBbig` gets the set B_i for each group for $i = 1, ..., G$ and combines the sets into one large data set. Then a robust FDA is the classical FDA applied to this cleaned data set. For the iris data, using `covmb2` did not discard any cases, so the robust FDA and classical FDA had identical output. See the R code below.

```
#Robust FDA with covmb2 set B from each group.
#This subset of cases can be found when p > n.
tem<-getBbig(x,gp)
outr<-lda(tem$Bbig,tem$grp)      #AER 0.02
plot(outr); 1-mean(predict(outr,x)$class==gp)
outr #Output is same as that for classical FDA.
```

5.7 Estimating the Test Error

Definition 5.10. The test error rate L_n is the population proportion of misclassification errors made by the DA method on test data.

The Bayes classifier has the smallest expected test error, but the Bayes classifier generally can't be computed used since the π_k and f_k are unknown. If it was known that $\pi_1 = 0.9$, a simple DA rule would be to always allocate $\boldsymbol{w}$ to group 1. Then the test error of this rule would be $L_n = 0.1$.

Generally the test error L_n needs to be estimated by $\hat{L}_n$. A simple method for estimating the test error is to apply the DA method to the training data and find the proportion of classification errors made. To help see why this method is poor, consider KNN with $K = 1$. Then the training data is perfectly classified with a training error rate of 0, although the test error rate may be quite high.

Definition 5.11. The *training error rate* or *apparent error rate* (AER) is

$$AER = \hat{L}_n = \frac{1}{n} \sum_{i=1}^{n_j} \sum_{j=1}^{G} I[\hat{Y}_{ij} \neq Y_{ij}]$$

where $\hat{Y}_{ij}$ is the DA estimate of Y_{ij} using all n training cases $\boldsymbol{x}_{1,1}, ..., \boldsymbol{x}_{G,n_G}$. Note that $Y_{ij} = j$ since $\boldsymbol{x}_{ij}$ comes from the jth group. If m_j of the n_j group j cases are correctly classified, then the *apparent error rate for group* j is $1 - m_j/n_j$. If $m_A = \sum_{j=1}^{G} m_j$ of the $n = \sum_{j=1}^{G} n_j$ training cases are correctly classified, then $AER = 1 - m_A/n$.

DA methods fit the training data better than test data, so the AER tends to underestimate the error rate for test data. We want to use a DA method with a low test error rate. Cross validation (CV) divides the training data into a big part and a small part, perhaps J times. For each of the J divisions, the DA rule is computed for the big part and applied to the small part. Hence the small part is used as a validation set. The proportion of errors made for the small part is recorded.

For leave one out or delete one cross validation, $J = n$, the big part uses $n - 1$ cases from the training data while the small part uses the 1 case left out of the big part. This case will either be correctly or incorrectly classified. The leave one out CV rule can sometimes be rapidly computed, but usually requires the DA method to be fit n times.

Definition 5.12. An estimator of the test error rate is the *leave one out cross validation* error rate

$$\hat{L}_n = \frac{1}{n}\sum_{i=1}^{n_j}\sum_{j=1}^{G} I(\hat{Y}_{ij} \neq Y_{ij})$$

where $\hat{Y}_{ij}$ is the estimate of Y_{ij} when $\boldsymbol{x}_{ij}$ is deleted from the n training cases $\boldsymbol{x}_{1,1}, ..., \boldsymbol{x}_{G,n_G}$. Note that $\hat{L}_n$ is the proportion of training cases that are misclassified by the n leave one out rules. If m_C is the number of cases correctly classified by leave one out classification, then $\hat{L}_n = 1 - m_C/n$.

For KNN, find the K cases in the training data closest to $\boldsymbol{x}_{i,j}$ not including $\boldsymbol{x}_{i,j}$. Then compute the leave one out cross validation error rate as in Definition 5.12.

Assume that the training data $\boldsymbol{x}_{1,1}, ..., \boldsymbol{x}_{n_G,G}$ is a random sample from the G populations so that $n_j/n \xrightarrow{P} \pi_j$ as $n \to \infty$ for $j = 1, ..., G$. Hence n_j/n is a consistent estimator of π_j. Following Devroye and Wagner (1982), when $K = 1$ the test error rate L_n of KNN method converges in probability to L where $L_B \leq L \leq 2L_B$ and L_B is the test error rate of the Bayes classifier. If $K_n \to \infty$ and $K_n/n \to 0$ as $n \to \infty$, then the KNN method converges to the Bayes classifier in that the KNN test error rate $L_n \xrightarrow{P} L_B$. Then the leave one out cross validation error rate $\hat{L}_n$ is a good estimator of L_n in that $2e^{-2n\epsilon^2}$ was usually an upper bound on $P[|\hat{L}_n - L_n| \geq \epsilon]$ for small $\epsilon > 0$.

For the method below, $J = 1$ and the validation set or hold-out set is the small part of the data. Typically 10% or 20% of the data is randomly selected to be in the validation set. Note that the DA method is only computed once to compute the error rate.

Definition 5.13. The *validation set* approach has $J = 1$. Let the validation set contain n_v cases $(\boldsymbol{x}_1, Y_1), ..., (\boldsymbol{x}_{n_v}, Y_{n_v})$, say. Then the *validation set* error rate is

$$\hat{L}_n = \frac{1}{n_v}\sum_{i=1}^{n_v} I(\hat{Y}_i \neq Y_i)$$

where $\hat{Y}_i$ is the estimate of Y_i computed from the DA method applied to the $n - n_v$ cases not in the validation set. If m_L is the number of the n_v cases from the validation set correctly classified, then $\hat{L}_n = 1 - m_L/n_v$.

The k-fold CV has $J = k$ partitions of the data into big and small sets, and the DA method is computed k times. The values $k = 5$ and 10 are common because they have been shown empirically to work well.

Definition 5.14. For *k-fold cross validation* (k-fold CV), randomly divide the training data into k groups or folds of approximately equal size $n_j \approx n/k$ for $j = 1, ..., k$. Leave out the first fold, fit the DA method to the $k - 1$ remaining folds, and then find the proportion of errors for the first fold. Repeat for folds 2, ..., k. The k-fold CV error rate is

$$\hat{L}_n = \frac{1}{n}\sum_{i=1}^{n_j}\sum_{j=1}^{G} I(\hat{Y}_{ij} \neq Y_{ij})$$

where $\hat{Y}_{ij}$ is the estimate of Y_{ij} when $\boldsymbol{x}_{ij}$ is in the deleted fold. If m_k is the number of the n training cases correctly classified, then $\hat{L}_n = 1 - m_k/n$.

Definition 5.15. A **truth table** or **confusion matrix** for a G category classifier is a $G \times G$ table with G labels on the top for the "truth" (true classes) and G labels on the left side for the predicted classes. The cells give classification counts. The diagonal cells are counts for correctly classified cases, while the off diagonals are counts for incorrectly classified cases. The error rate = (sum of off diagonal cells)/(sum of all cells) =
1 - (sum of diagonal cells)/(sum of all cells).

For a binary classifier, consider the following truth table where the counts TN = true negative, FN = false negative, FP = false positive, and TP = true positive.

	truth		total
	-1	1	
predict -1	TN	FN	N^*
1	FP	TP	P^*
total	N	P	

The true positive rate = TP/P = *sensitivity* = power = recall = 1− type II error. The false positive rate = FP/N = 1− *specifity* ≈ type I error. The positive predicted value = TP/P^* ≈ *precison* = 1− false discovery proportion. The negative predicted value = TN/N. The error rate = $(FP + FN)/(FP + FN + TN + TP)$.

For a binary classifier, sometimes one error is much more important than the other. For example consider a loan with categories "default" and "does not default." Misclassifying "default" should be small compared to misclassifying "does not default."

A ROC curve is used to evaluate a binary classifier. The horizontal axis is the false positive rate while the vertical axis is the true positive rate. Both axes go from 0 to 1, so the total area of the square plot is 1. The overall performance of the binary classifier is summarized by the area under the curve (AUC). An ideal ROC curve is close to the top left corner of the plot, so the larger the AUC, the better the classifier. Note that $0 \leq AUC \leq 1$. A classifier with AUC = 0.5 does no better than chance. A ROC from test data or validation data is better than a ROC from training data.

5.8 Some Examples

Example 5.2. The following output illustrates crude variable selection using the *LDA* function. See Problems 5.6 and 5.7. The code deletes predictors as long as the AER does not increase if the predictor is deleted. Using all of the data, the AER = 0.0357. Eventually the AER = 0.

```
library(MASS) #Output for Example 5.2.
group <- pottery[pottery[,1]!=5,1]
group <- (as.integer(group!=1)) + 1
x <- pottery[pottery[,1]!=5,-1]

out<-lda(x,group)
1-mean(predict(out,x)$class==group)
[1] 0.03571429 #AER using all of the predictors.
out<-lda(x[,-c(1)],group)
1-mean(predict(out,x[,-c(1)])$class==group)
out<-lda(x[,-c(1,2)],group)
1-mean(predict(out,x[,-c(1,2)])$class==group)
out<-lda(x[,-c(1,2,3)],group)
1-mean(predict(out,x[,-c(1,2,3)])$class==group)
out<-lda(x[,-c(1,2,3,4)],group)
1-mean(predict(out,x[,-c(1,2,3,4)])$class==group)
out<-lda(x[,-c(1,2,3,4,5)],group)
1-mean(predict(out,x[,-c(1,2,3,4,5)])$class==group)
[1] 0.03571429 #Can delete predictors 1-5.
out<-lda(x[,-c(1,2,3,4,5,6)],group)
1-mean(predict(out,x[,-c(1,2,3,4,5,6)])$class==group)
[1] 0.07142857 #Predictor x6 is important.
out<-lda(x[,-c(1,2,3,4,5,7)],group)
1-mean(predict(out,x[,-c(1,2,3,4,5,7)])$class==group)
out<-lda(x[,-c(1,2,3,4,5,7,8)],group)
1-mean(predict(out,x[,-c(1,2,3,4,5,7,8)])
$class==group)
out<-lda(x[,-c(1,2,3,4,5,7,8,9)],group)
1-mean(predict(out,x[,-c(1,2,3,4,5,7,8,9)])
$class==group)
out<-lda(x[,-c(1,2,3,4,5,7,8,9,10)],group)
1-mean(predict(out,x[,-c(1,2,3,4,5,7,8,9,10)])
$class==group)
out<-lda(x[,-c(1,2,3,4,5,7,8,9,10,11)],group)
1-mean(predict(out,x[,-c(1,2,3,4,5,7,8,9,10,11)])
$class==group)
[1] 0.07142857 #Predictor x11 is important.
out<-lda(x[,-c(1,2,3,4,5,7,8,9,10,12)],group)
```

```
1-mean(predict(out,x[,-c(1,2,3,4,5,7,8,9,10,12)])
$class==group)
out<-lda(x[,-c(1,2,3,4,5,7,8,9,10,12,13)],group)
1-mean(predict(out,x[,-c(1,2,3,4,5,7,8,9,10,12,13)])
$class==group)
out<-lda(x[,-c(1,2,3,4,5,7,8,9,10,12,13,14)],group)
1-mean(predict(out,x[,-c(1,2,3,4,5,7,8,9,10,12,13,
14)])$class==group)
[1] 0.07142857 #Predictor x14 is important.
out<-lda(x[,-c(1,2,3,4,5,7,8,9,10,12,13,15)],group)
1-mean(predict(out,x[,-c(1,2,3,4,5,7,8,9,10,12,13,
15)])$class==group)
out<-lda(x[,-c(1,2,3,4,5,7,8,9,10,12,13,15,16)],group)
1-mean(predict(out,x[,-c(1,2,3,4,5,7,8,9,10,12,13,
15,16)])$class==group)
out<-lda(x[,-c(1,2,3,4,5,7,8,9,10,12,13,15,16,17)],
group)
1-mean(predict(out,x[,-c(1,2,3,4,5,7,8,9,10,12,13,
15,16,17)])$class==group)
[1] 0.03571429
out<-lda(x[,-c(1,2,3,4,5,7,8,9,10,12,13,15,16,17,
18)],group)
1-mean(predict(out,x[,-c(1,2,3,4,5,7,8,9,10,12,13,
15,16,17,18)])$class==group)
[1] 0.07142857  #Predictor x18 is important.
out<-lda(x[,-c(1,2,3,4,5,7,8,9,10,12,13,15,16,17,
19)],group)
1-mean(predict(out,x[,-c(1,2,3,4,5,7,8,9,10,12,13,
15,16,17,19)])$class==group)
[1] 0.03571429
out<-lda(x[,-c(1,2,3,4,5,7,8,9,10,12,13,15,16,17,
19,20)],group)
1-mean(predict(out,x[,-c(1,2,3,4,5,7,8,9,10,12,13,
15,16,17,19,20)])$class==group)
[1] 0
#Predictors x6, x11, x14, x18 seem good for LDA.
```

Example 5.3. This example illustrates that the AER tends to underestimate the test error rate compared to the validation set approach. The validation test error estimates can change greatly when the random number generator seed is changed. See Definitions 5.11 and 5.13. The men's basketball data set `mbb1415` is described in Problem 7.4, which tells how to get the data set into *R*. The KNN method AER is especially poor when K is small ($K < 10$, say). The KNN method also depends on a random number seed, perhaps to handle ties. (If there are three groups and $K = 3$, it is possible that the 3 nearest neighbors to $\boldsymbol{w}$ come from groups 1, 2, and 3. How does

KNN decide which group to allocate $\boldsymbol{w}$?) The R commands below standardize the variables to have mean 0 and variance 1, puts guards into group 1, small forwards into group 2, centers and power forwards into group 3, and individuals with unknown position into group 0. Then individuals who do not play much (are in the bottom quartile in playing time) are deleted. Next, players in group 0 are deleted, leaving a data set z with 86 cases, 3 groups, and 35 predictor variables. The data set z is also divided into a validation test set ztest of 20 cases and a training set ztrain of 66 cases.

```
set.seed(1)
z <- mbb1415[,-1]
z <- scale(z) #standardize the variables
grp <- mbb1415[,1]
grp[grp==2]<-1
grp[grp==3]<-2
grp[grp==4]<-3
grp[grp==5]<-3
#Put guards in group 1, small forwards in group 2,
#centers and power forwards in group 3,
#unknowns in group 0.
#Get rid of players who did not play much.
z <- z[mbb1415[,3]>182,]
grp <- grp[mbb1415[,3]>182]
#Get rid of group 0, 86 cases left.
z <- z[grp>0,]
grp<-grp[grp>0]
indx<-sample(1:86,replace=F)
train <- indx[21:86]
test <- indx[1:20]
ztest <- z[test,]   #20 test cases
grptest <- grp[test]
ztrain <- z[train,]
grptrain <- grp[train]
```

Since x_1 is used as group, $z_i = x_{i+1}$. Below we use z_7 = turnovers, z_{10} = stl.pos (stolen possessions, a ball handling rating), z_{12} = rebounds, z_{13} = offensive rebounds, z_{28} = three point field goal percentage, and z_{32} = free throw percentage. With 2 nearest neighbors, the AER is 0.151, but (the validation error rate) VER = 0.45. With 1 nearest neighbor, the AER = 0 since each training case is its own nearest neighbor. Hence the training cases are perfectly classified.

```
#see what the variables are
z[1,c(7,10,12,13,28,32)]

library(class)
```

```
out <- knn(z[,c(7,10,12,13,28,32)],
z[,c(7,10,12,13,28,32)],grp,k=2)
mean(grp!=out)  #0.151 AER

out<-knn(ztrain[,c(7,10,12,13,28,32)],
ztest[,c(7,10,12,13,28,32)],grptrain,k=2)
mean(grptest!=out) #0.45 validation ER

out <- knn(z[,c(7,10,12,13,28,32)],
z[,c(7,10,12,13,28,32)],grp,k=1)
mean(grp!=out)  #0.0 AER

out<-knn(ztrain[,c(7,10,12,13,28,32)],
ztest[,c(7,10,12,13,28,32)],grptrain,k=1)
mean(grptest!=out) #0.45 validation ER
```

The output below shows that VER = 0.5 and AER = 0.22 with FDA (LDA), and VER = 0.45 and AER = 0.13 with QDA.

```
library(MASS) #three ways to get VER = 0.5
out <- lda(z[,c(7,10,12,13,28,32)],grp, subset=train)
1-mean(predict(out,z[-train,c(7,10,12,13,28,32)])
$class==grp[-train])
1-mean(predict(out,z[test,c(7,10,12,13,28,32)])
$class==grptest)
1-mean(predict(out,ztest[,c(7,10,12,13,28,32)])
$class==grptest)
out<-lda(z[,c(7,10,12,13,28,32)],grp)
1-mean(predict(out,z[,c(7,10,12,13,28,32)])
$class==grp) #AER =0.22

out <- qda(z[,c(7,10,12,13,28,32)],grp, subset=train)
#VER = 0.45
1-mean(predict(out,ztest[,c(7,10,12,13,28,32)])
$class==grptest)
out<-qda(z[,c(7,10,12,13,28,32)],grp)
1-mean(predict(out,z[,c(7,10,12,13,28,32)])
$class==grp) #AER =0.13
```

5.9 Classification Trees, Bagging, and Random Forests

A classification tree is a flexible method for classification that is very similar to the regression tree of Section 4.10. The method produces a graph called a tree. Each branch has a label like $x_i > 7.56$ if x_i is quantitative, or $x_j \in \{a, c\}$

(written $x_j = ac$) where x_j is a factor taking on values a, b, c, d, e, f, say. **Unless told otherwise**, go to the left branch if the condition is true, go to the right branch if the condition is false. (Some software switches this. Check the story problem.) The bottom of the tree has leaves that give a label for a group such as $\hat{Y} = j$ for some $j = 1, ..., G$. The root is the top node, a leaf is a terminal node, and a split is a rule for creating new branches. Each node has a left and right branch.

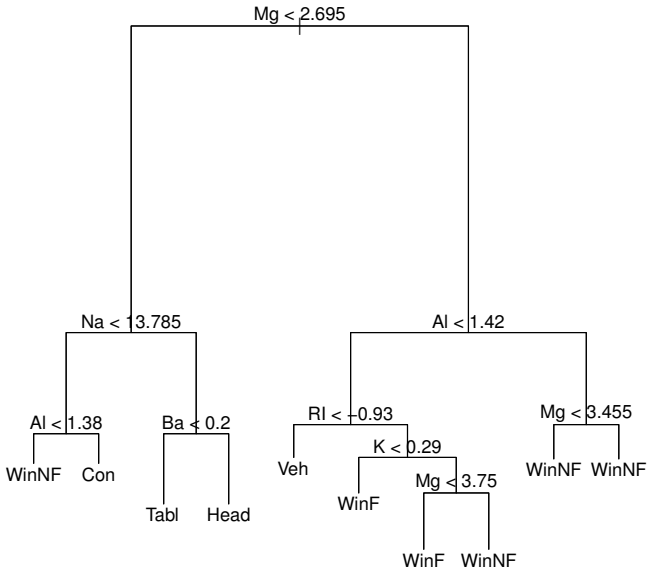

Fig. 5.3 Classification Tree for Example 5.4.

Example 5.4.

The Venables and Ripley (2010) *fgl* data set has fragments of glass classified by five chemicals $x_1 = Al, x_2 = Ba, x_3 = K, x_4 = Mg, x_5 = Na$, and $x_6 = RI$ = refractive index. The categories which occur are window float glass (WinF), window non-float glass (WinNF), vehicle window glass (Veh), containers (Con), tableware (Tabl), and vehicle headlamps (Head). In the second node to the left, the split is $NA < 13.785$, but the 13.785 is hard to read.

a) Predict the class Y if $Mg = 2, Na = 14$ and $Ba = 0.35$.

Solution: Go left, right, right to predict class Head.

b) Predict the class Y if $Mg = 3.1$ and $Al = 1.6$.

Solution: Go right right left to predict class WinNF.

Note that the tree in Figure 5.3 can be simplified: predict WinNF if $Mg \geq 2.65$ and i) $Al \geq 1.42$ or ii) $Al < 1.42$ and $RI \geq -0.93$.

Classification trees have some advantages. Trees can be easier to interpret than competing methods when some predictors are numerical and some are categorical. Trees are invariant to monotone (increasing or decreasing) transformations of the predictor variable x_i. Trees can handle complex unknown interactions. Classification and regression trees i) give prediction rules that can be rapidly and repeatedly evaluated, ii) are useful for screening predictors (interactions, variable selection), iii) can be used to assess the adequacy of linear models, and iv) can summarize large multivariate data sets.

Trees that use recursive partitioning for classification and regression trees use the CART algorithm. In growing a tree, the binary partitioning algorithm recursively splits the data in each node until either the node is homogeneous (roughly 0 training data misclassifications for a classification tree) or the node contains too few observations (default ≤ 5). The *deviance* is a measure of node homogeneity, and deviance = 0 for a perfectly homogeneous node. For a classification tree, $\hat{Y}$ is often the mode of the node labels ($\hat{Y}$ is the class that occurs the most).

Trees divide the predictor space (set of possible values of the training data $\boldsymbol{x}_i$) into J distinct and nonoverlapping regions $R_1, ..., R_J$ that are high dimensional boxes. Then for every observation that falls in R_j, make the same prediction. Hence $\hat{Y}_{R_j}$ = modal class $mode_j$ of training data Y_i in R_j. Choose R_j so $RSS = \sum_{j=1}^{J} \sum_{i \in R_j} I(Y_i \neq \hat{Y}_{R_j})$ is small. Let $\{\boldsymbol{x} | x_j < s\}$ be the region in the predictor space such that $x_j < s$ where $\boldsymbol{x} = (x_1, ..., x_p)^T$. Define 2 regions $R_1(j, s) = \{\boldsymbol{x} | x_j < s\}$ and $R_2(j, s) = \{\boldsymbol{x} | x_j \geq s\}$. Then seek cutpoint s and variable x_j to minimize

$$\sum_{i:\boldsymbol{x}_i \in R_1(j,s)} I(Y_i \neq \hat{Y}_{R_1}) + \sum_{i:\boldsymbol{x}_i \in R_2(j,s)} I(Y_i \neq \hat{Y}_{R_2}).$$

This can be done "quickly" if p is small (could use order statistics). Then repeat the process looking for the best predictor and the best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions. Only split one of the regions, R_1, R_2, and R_3. Continue this process until a stopping criterion is reached such as no region contains more than 5 observations (and stop if the region is homogeneous). If J is too large, the tree overfits.

The null classifier hat $\hat{Y} = d$ where d is the modal (dominant) class. So if k% of the test observations belong to the dominant class, then the test error =

$$\frac{100 - k}{100} \leq 1 - \frac{1}{G}$$

where there are G groups since $k \geq 100/G$. Classifiers that do not beat the null classifier are very bad.

Classification trees are often beat by one of the earlier techniques from this chapter. Bagging, pruning, and random forests makes trees more competitive. The following subsections follow James et al. (2013) closely.

5.9.1 Pruning

Trees use regions $R_1, ..., R_J$, and if J is too large, the tree overfits. One strategy is to grow a large tree T_0 with J_0 regions, then prune it to get a subtree T_α with J_α regions.

Next, we describe cost complexity pruning = weakest link pruning. Let $T \subseteq T_0$, $\alpha \geq 0$, and $|T|$ = number of terminal nodes of tree T. Each terminal node corresponds to a hyperbox region R_i. Let R_m be the region corresponding to the mth terminal node and $\hat{Y}_{R_m}$ be the predicted response for R_m. For each value of $\alpha > 0$, there corresponds a subtree $T \subseteq T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i:\boldsymbol{x}_i \in R_m} I(Y_i \neq \hat{Y}_{R_m}) + \alpha|T| \tag{5.6}$$

is as small as possible. (Replace $I(Y_i \neq \hat{Y}_{R_m})$ by $(y_i - \hat{y}_{R_m})^2$ for a regression tree.) Note that $\alpha = 0$ has $T = T_0$ and (5.16) $= RSS(T_0)$ = training data RSS for T_0. Much like lasso, there is a sequence of nested subtrees

$$T_{\alpha_m} \subseteq \cdots \subseteq T_{\alpha_2} \subseteq T_{\alpha_1} \subseteq T_0. \tag{5.7}$$

Branches get "pruned" from T_0 in a nested and predictable fashion.

The pruning algorithm is a) build tree T_0, stopping when each (region corresponding to a terminal node has ≤ 5 observations. b) Use (5.6) to obtain (5.7). c) Use k-fold CV to choose $\alpha = \alpha_d$: for each $i \in 1, ..., k$, i) repeat steps a) and b) on all but the ith fold. ii) Evaluate the mean squared prediction error

$$MSE_i = \frac{1}{n_i} \sum_{j=1}^{n_i} I(Y_{ji} \neq \hat{Y}_j(i))$$

on the data Y_{ji} in the left out fold i as a function of α. Note that $MSE_i =$ proportion misclassified in the ith fold. Average the results for each value of α am pick α_d to minimize the average error

$$CV(k) = \frac{1}{k} \sum_{i=1}^{k} MSE_i.$$

d) Use tree T_{α_d} from (5.7). Note that if $n_i = n/k$, then

$$CV(k) = \frac{1}{n}\sum_{j=1}^{n} I(Y_{ji} \neq \hat{Y}_j(i)) =$$

proportion of misclassified observations. (For a regression tree, use

$$MSE_i = \frac{1}{n_i}\sum_{j=1}^{n_i}(Y_{ji} - \hat{Y}_j(i))^2.)$$

5.9.2 Bagging

Bagging was used before: compute $T_1^*, ..., T_B^*$ with the bootstrap, and the sample mean

$$\overline{T}^* = \frac{1}{B}\sum_{i=1}^{B} T_i$$

is the baggin estimator. For a regression tree, draw a sample of size n with replacement from the training data $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$. Fit the tree and find $\hat{f}_1(\boldsymbol{x})$. Repeat B times to get $T_i^* = \hat{f}_i(\boldsymbol{x})$. The trees are not pruned, so terminate when each terminal node has 5 or fewer observations.

Bagging a classification tree draws a sample of size n_j from each group with replacement. For the ith bootstrap estimator ($i = 1, ..., B$), fit the classification tree, and let $\hat{f}_i^*(\boldsymbol{x}) = j_i(\boldsymbol{x}) \in \{1, ..., G\}$ where Y takes on levels $1, ..., G$. That is, determine how the classification tree classifies $\boldsymbol{x}$. Compute $\hat{f}_1^*(\boldsymbol{x}), ..., \hat{f}_B^*(\boldsymbol{x})$, and let $m_k =$ the number of $j_i(\boldsymbol{x}) = k$ for $k = 1, ..., G$. Take $\hat{f}_{bag}(\boldsymbol{x}) = d$ where $m_d = max\{m_1, ..., m_G\}$.

For each bootstrap sample b, let $\boldsymbol{x}_{i_1}, ..., \boldsymbol{x}_{i_{k_b}}$ be the k_b observations not in the bootstrap sample. These a the "out of bag" (OOB) observations. Predict $\hat{Y}$ for each OOB observation. Doing this for all B bootstraps produces about $e^{-1}b \approx B/3$ predictors for each $\boldsymbol{x}_i$. Let $\hat{Y}_{io}$ = mode level for a classification tree. Then the OOB MSE =

$$\frac{1}{n}\sum_{i=1}^{n} I(Y_i \neq \hat{Y}_{i0})$$

is "virtually equivalent" to the leave one out CV estimator for large enough B. (For a regression tree, let $\hat{Y}_{io}$ = the average of the $\hat{Y}_i$, and replace $I(Y_i \neq \hat{Y}_{i0})$ by $(Y_i - \hat{Y}_{i0})^2$ to get the OOB MSE.)

For classification trees, let $\hat{\rho}_{mk}$ = proportion of training observations in R_m from the kth class. Then Gini's index =

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

is small if all $\hat{\rho}_{mk}$ are close to 0 or 1.

For bagging with B trees, a measure of variable importance can be computed for each variable using the number of splits for each variable. This measure can be summarized with a variable importance plot.

For a binary classifier whith $Y = 0$ or 1, for a fixed test value $\boldsymbol{x}$, the bootstrap produces B estimators of $P(Y = 1|\boldsymbol{x})$. Two common ways to get $\hat{Y}|\boldsymbol{x}$ are a) $\hat{Y}|\boldsymbol{x}$ = mode class of 0 or 1, and b) average the B estimates of $P(Y = 1|\boldsymbol{x})$ and set $\hat{Y}|\boldsymbol{x} = 0$ if ave. $\hat{P}(Y = 1|\boldsymbol{x}) \leq 0.5$, with $\hat{Y}|\boldsymbol{x} = 1$, otherwise.

5.9.3 Random Forests

For random forests, the bootstrap is used, but each time a split is considered, a random sample of $m = \lceil \sqrt{p} \rceil$ predictors is chosen as split candidates. Random forest tend to produce bootstrap trees that are less correlated than bagged trees (that use $m = p$), and the random forests estimator tends to have better test error and OOB error than the bagging estimator. Also, B around a few hundred seems to work.

If there is a single strong predictor, bagged trees tend to use that predictor in the first split. For random forests, the strong predictor is not considered for $(p - m)/p$ splits, on average.

5.10 Support Vector Machines

This section follow James et al. (2013, ch. 9) closely. Logistic regression is used a lot in biostatistics and epidemiology where the focus is statistical inference. Support vector machines (SVMs) are used in machine learning where the goal is classification accuracy.

5.10.1 Two Groups

When $p >> n$, there is often a hyperplane that perfectly separates two groups (even if the two groups are iid from the same population: severe overfitting). The launching point for SVMs was finding the optimal separating hyperplane. *Wide data* has $p >> n$. If $n \leq p + 1$, then there is a separating hyperplane unless there are "exact predictor ties across the class barrier."

For 2 groups, let $SP = \beta_0 + \boldsymbol{\beta}^T\boldsymbol{x}$. Classify $\boldsymbol{x}$ in group 1 if $ESP > 0$ and in group -1 if $ESP < 0$. So the classifier $\hat{C}(\boldsymbol{x}) = sign(ESP)$. Note that the second group now has label -1 instead of 0.

Suppose two groups of training data can be separated by a hyperplane. Then there are two parallel separating hyperplanes where the first separating hyperplane passes through some cases in group 1 and the second hyperplane passes through some cases in group 2. The distance between the two separating hyperplanes is called the margin between classes. The cases that just touch the two separating hyperplanes are called the support set. Then the "optimal separating hyperplane" ESP has the largest margin on the training data, and the optimal separating hyperplane is parallel and equidistant from the two separating hyperplanes that determine the support set.

As a visual aid, use "0" for cases from group -1 and "+" for cases from group 1. Draw a plot on a piece of paper where the two groups can be separated by a line. A separating line that touches one case from each group has margin 0. Draw two parallel lines such that one line touches at least one 0 and one line touches at least one +. Make the distance between the two parallel lines as far as possible (biggest margin). Then the parallel line in the middle of these two parallel lines is the optimal sepparating hyperplane (line).

Think of the hyperplane $\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$ as separating $\mathbb{R}^p$ into two halves.

Definition 5.16. A separating hyperplane has $SP > 0$ if $\boldsymbol{x} \in$ group 1 and $SP < 0$ if $\boldsymbol{x} \in$ group -1. So $Y_i \ SP_i = Y_i(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i) > 0$ for $i = 1, ..., n$.

Now let $Z = 1$ iff $Y = 1$ and $Z = 0$ iff $Y = -1$. Then think of the binary classifier that uses ESP as a binary regression $Z|\boldsymbol{x} \sim bin(m = 1, \rho(\boldsymbol{x}))$ where $\rho(\boldsymbol{x}) = \rho(SP) = P(Z = 1|\boldsymbol{x}) = P(Y = 1|\boldsymbol{x})$ is unknown. Make a response plot of ESP versus Z with lowess and possibly a step function added as visual aids. The bootstrap is likely useful if $n_i \geq 10p$ for both groups. a) Use the bootstrap with with n_i cases selected with replacements from each group. b) Use the bootstrap with $Z_i^* = 1$ with probability $\hat{\rho}(\boldsymbol{x}_i)$ and $Z_i^* = 0$ with probability $1 - \hat{\rho}(\boldsymbol{x}_i)$. Fit the SVM using $\boldsymbol{Y}_j^*$ and $\boldsymbol{X}$ for $j = 1, ..., B$.

Classification and regression trees (CART) splits $\mathbb{R}_p$ with regions $R_m \in \mathbb{R}_p$ while a SVM splits $\mathbb{R}_p$ into two regions using $ESP \in \mathbb{R}$ so there is dimension reduction. The SVM split tries to make the 2 "halves" or partitions as homogeneous as possible.

The hyperplanes parallel to the ESP hyperplane that form the boundaries of the margin are called fences. The fence pass through at least two training data cases. These cases form the support set S of support vectors. It turns out that if a separating hyperplane exists, then the optimal margin classifier $\hat{\boldsymbol{\beta}}_M = \sum_{i \in S} \hat{\alpha}_i \boldsymbol{x}_i$.

Let M be the margin. The *optimal margin classifier* $(\hat{\beta}_{0M}, \hat{\boldsymbol{\beta}}_M)$ maximizes M subject to

$$Y_i \ SP_i = Y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq M \tag{5.8}$$

for all $i = 1, ..., n$. This is called a *hard margin classifier* since no cases from either group can pass the fences of the classifier. The maximization is over $\beta_0 \in \mathbb{R}$ and $\boldsymbol{\beta} \in \mathbb{R}_p$. The maximization is equivalent to minimizing $\|\boldsymbol{\beta}\|_2$ subject to (5.8).

A *soft margin classifier* allows cases from either group to pass the fences or to be misclassified. This classifier minimizes $\|\boldsymbol{\beta}\|_2$ subject to $Y_i(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i) \geq 1 - \epsilon_i$ for $i = 1, ..., n$ where the slack variables $\epsilon_i \geq 0$ and $\sum_{i=1}^n \epsilon_i \leq D$. Hastie et al. (2001, p. 380) showed that this minimization is equivalent to minimizing

$$\sum_{i=1}^{n} [1 - Y_i(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i)]_+ + \lambda \|\boldsymbol{\beta}\|_2^2 \tag{5.9}$$

where $[w]_+ = w$ if $w \geq 0$ and $[w]_+ = 0$ if $w < 0$. The *hinge loss* $[1 - Y_i(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i)]_+ = 0$ if $\boldsymbol{x}_i$ is on the correct side of the margin. Otherwise, the hinge loss is the cost of $\boldsymbol{x}_i$ being on the wrong side of the margin. The minimization is over $\beta_0 \in \mathbb{R}$ and $\boldsymbol{\beta} \in \mathbb{R}_p$, and the criterion (5.9) is similar to the ridge regression criterion.

A *support vector machine* (SVM) that uses $\boldsymbol{x}_i$ minimizes the above criterion. For separable data, $(\hat{\beta}_{0,SVM}, \hat{\boldsymbol{\beta}}_{SVM}) \to (\hat{\beta}_{0,M}, \hat{\boldsymbol{\beta}}_M)$ as $\lambda \to 0$. A lasso-SVM minimizes

$$\sum_{i=1}^{n} [1 - Y_i(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i)]_+ + \lambda \|\boldsymbol{\beta}\|_1, \tag{5.10}$$

and does variable selection. A "ridged logistic regression" with $Y_i \in \{-1, 1\}$ minimizes

$$\sum_{i=1}^{n} \log[1 + \exp(-Y_i(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i))] + \lambda \|\boldsymbol{\beta}\|_2^2. \tag{5.11}$$

The criterion (5.9) and (5.11) are similar. It can be shown that the SVM maximizes M = width of margin subject to $\sum_{j=1}^p \beta_j^2 = 1$ such that $\epsilon_i \geq 0$, $\sum_{i=1}^p \epsilon_i \leq D$, and $Y_i(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i) \geq M(1 - \epsilon_i)$. Compare (5.8). The maximization is over $\beta_0 \in \mathbb{R}$, $\boldsymbol{\beta} \in \mathbb{R}^p$, and $\epsilon_1, ..., \epsilon_n$.

A slack variable $\epsilon_i = 0$ if $\boldsymbol{x}_i$ is on the correct side of the margin. If $\epsilon_i > 0$, then $\boldsymbol{x}_i$ is on the wrong side of the hyperplane. $Y_i(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i) \geq M$ has $\epsilon_i = 0$ and is necessary for $\boldsymbol{x}_i$ to be on the correct side of the margin. If $Y_i(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i) \geq M(1 - \epsilon_i)$ with $\epsilon_i >$ (but not if $\epsilon_i = 0$), then $\boldsymbol{x}_i$ is on the wrong side of the hyperplane. See Definition 5.15.

It can be shown that $\hat{\boldsymbol{\beta}}_{SVM} = \sum_{i \in S} \hat{\gamma}_i \boldsymbol{x}_i$, and $ESP = \hat{\beta}_{0,SVM} + \boldsymbol{x}^T \hat{\boldsymbol{\beta}}_{SVM} = \hat{\beta}_{0,SVM} + \sum_{i \in S} \hat{\gamma}_i \boldsymbol{x}^T \boldsymbol{x}_i$. This quantity can ge computed using the $n \times n$ Gram matrix $\boldsymbol{X}\boldsymbol{X}^T$ with $O(n^2 p)$ complexity, or using $\boldsymbol{X}^T \boldsymbol{X}$ with $O(np^2)$ complexity. Ridge regression could also be computed this way.

Sometimes one or a few cases shift the maximal margin hyperplane. The SVM classifier is a soft margin classifier and can do better.

The SVM that uses $\boldsymbol{x}_i$ is like LDA and logistic regression for two groups. An SVM that uses a kernel function is similar to QDA. Let the kernel function be $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. A linear kernel is $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j$. A polynomial kernel of degree d is $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i^T \boldsymbol{x}_j)^d$. A radial kernel is $k(\boldsymbol{x}_i, \boldsymbol{x}_j) =$

$$\exp\left[-\gamma \sum_{k=1}^{p} (x_{ik} - x_{jk})^2\right] = \exp[-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2].$$

If $\boldsymbol{x}$ is far from $\boldsymbol{x}_i$, then $\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2$ is large so $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp[-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2]$ is tiny, and $\boldsymbol{x}_i$ has almost no contribution to $SP = SP(\boldsymbol{x}) =$
$\beta_0 + \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i)$. Compare KNN.

A *support vector machine* (SVM) uses

$$SP = SP(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) = \beta_0 + \sum_{i \in S} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i)$$

where S is the index of support vectors. The support vectors determine the hyperplane and the margin: if the support vectors are moved, then the hyperplane moves.

Using $k(\boldsymbol{x}, \boldsymbol{x}_i)$ leads o nonlinear decision boundaries if the kernel k is nonlinear. The kernel is a bivariate transformation. There are $\binom{n}{2} = n(n-1)/2$ istinct pairs $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that are needed to estimate β_0 and the α_i. The SVM with $ESP = ESP(\boldsymbol{x}) = \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\alpha}_i k(\boldsymbol{x}, \boldsymbol{x}_i)$ is a competitor for QDA while the SVM with $ESP = ESP(\boldsymbol{x}) = \hat{\beta}_0 + \hat{\boldsymbol{\beta}}^T \boldsymbol{x}$ is a competitor for LDA.

5.10.2 SVM With More Than Two Groups

There are two common ways to extend binary classifies, such as SVMs and binary logistic regression, to $G > 2$ classes. First, the *one versus one* or *all pairs* classifier constructs $\binom{G}{2}$ binary classifiers, one for each pair of groups. Classify $\boldsymbol{x}$ with $f_{ij}(\boldsymbol{x}) = ESP_{ij}(\boldsymbol{x})$, and let m_i = number of times $\boldsymbol{x}$ is predicted to be in class i. Then $\hat{Y}(\boldsymbol{x}) = d$ where $m_d = \max(m_1, ..., m_G)$.

Second, the *one versus all* classifier fits G binary classifiers (such as SVMs): group $i = 1$ versus the $G-1$ other classes coded as -1 with $ESP_i(\boldsymbol{x}) = f_i(\boldsymbol{x})$. Then $\hat{Y}(\boldsymbol{x}) = d$ where $f_d(\boldsymbol{x}) = \max(f_1(\boldsymbol{x}), ..., f_G(\boldsymbol{x}))$.

5.11 Summary

1) In *supervised classification*, there are G known groups or populations and m test cases. Each case is assigned to exactly one group based on its mea-

surements $\boldsymbol{w}_i$. Assume that for each population there is a probability density function (pdf) $f_j(\boldsymbol{z})$ where $\boldsymbol{z}$ is a $p \times 1$ vector and $j = 1, ..., G$. Hence if the random vector $\boldsymbol{x}$ comes from population j, then $\boldsymbol{x}$ has pdf $f_j(\boldsymbol{z})$. Assume that there is a random sample of n_j cases $\boldsymbol{x}_{1,j}, ..., \boldsymbol{x}_{n_j,j}$ for each group. The $n = \sum_{j=1}^{G} n_j$ cases make up the training data. Let $(\overline{\boldsymbol{x}}_j, \boldsymbol{S}_j)$ denote the sample mean and covariance matrix for each group. Let the ith test case $\boldsymbol{w}_i$ be a new $p \times 1$ random vector from one of the G groups, but the group is unknown. *Discriminant analysis* attempts to allocate the $\boldsymbol{w}_i$ to the correct groups for $i = 1, ..., m$.

2) The *maximum likelihood discriminant rule* allocates case $\boldsymbol{w}$ to group a if $\hat{f}_a(\boldsymbol{w})$ maximizes $\hat{f}_j(\boldsymbol{w})$ for $j = 1, ..., G$. This rule is robust to nonnormality and the assumption of equal population dispersion matrices, but f_j is hard to estimate for $p > 2$.

3) Given the $\hat{f}_j(\boldsymbol{w})$ or a plot of the $\hat{f}_j(\boldsymbol{w})$, determine the maximum likelihood discriminant rule.

For the following rules, assume that costs of correct and incorrect allocation are unknown or equal, and assume that the probabilities $\pi_j = \rho_j(\boldsymbol{w}_i)$ that $\boldsymbol{w}_i$ is in group j are unknown or equal: $\pi_j = 1/G$ for $j = 1, ..., G$. Often it is assumed that the G groups have the same covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{x}}$. Then the pooled covariance matrix estimator is

$$\boldsymbol{S}_{pool} = \frac{1}{n - G} \sum_{j=1}^{G} (n_j - 1) \boldsymbol{S}_j$$

where $n = \sum_{j=1}^{G} n_j$. Let $(\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j)$ be the estimator of multivariate location and dispersion for the jth group, e.g. the sample mean and sample covariance matrix $(\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j) = (\overline{\boldsymbol{x}}_j, \boldsymbol{S}_j)$.

4) Assume the population dispersion matrices are equal: $\boldsymbol{\Sigma}_j \equiv \boldsymbol{\Sigma}$ for $j = 1, ..., G$. Let $\hat{\boldsymbol{\Sigma}}_{pool}$ be an estimator of $\boldsymbol{\Sigma}$. Then the *linear discriminant rule* is allocate $\boldsymbol{w}$ to the group with the largest value of

$$d_j(\boldsymbol{w}) = \hat{\boldsymbol{\mu}}_j^T \hat{\boldsymbol{\Sigma}}_{pool}^{-1} \boldsymbol{w} - \frac{1}{2} \hat{\boldsymbol{\mu}}_j^T \hat{\boldsymbol{\Sigma}}_{pool}^{-1} \hat{\boldsymbol{\mu}}_j = \hat{\alpha}_j + \hat{\boldsymbol{\beta}}_j^T \boldsymbol{w}$$

where $j = 1, ..., G$. *Linear discriminant analysis* (LDA) uses $(\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_{pool}) = (\overline{\boldsymbol{x}}_j, \boldsymbol{S}_{pool})$. LDA is robust to nonnormality and somewhat robust to the assumption of equal population covariance matrices.

5) The *quadratic discriminant rule* is allocate $\boldsymbol{w}$ to the group with the largest value of

$$Q_j(\boldsymbol{w}) = \frac{-1}{2} \log(|\hat{\boldsymbol{\Sigma}}_j|) - \frac{1}{2} (\boldsymbol{w} - \hat{\boldsymbol{\mu}}_j)^T \hat{\boldsymbol{\Sigma}}_j^{-1} (\boldsymbol{w} - \hat{\boldsymbol{\mu}}_j)$$

where $j = 1, ..., G$. *Quadratic discriminant analysis* (QDA) uses $(\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j) = (\overline{\boldsymbol{x}}_j, \boldsymbol{S}_j)$. QDA has some robustness to nonnormality.

6) The *distance discriminant rule* allocates $\boldsymbol{w}$ to the group with the smallest squared distance $D^2_{\boldsymbol{w}}(\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j) = (\boldsymbol{w}-\hat{\boldsymbol{\mu}}_j)^T \hat{\boldsymbol{\Sigma}}_j^{-1}(\boldsymbol{w}-\hat{\boldsymbol{\mu}}_j)$ where $j = 1, ..., k$. This rule is robust to nonnormality and the assumption of equal $\boldsymbol{\Sigma}_j$, but needs $n_j \geq 10p$ for $j = 1, ..., G$.

7) Assume that $G = 2$ and that there is a group 0 and a group 1. Let $\rho(\boldsymbol{w}) = P(\boldsymbol{w} \in \text{group 1})$. Let $\hat{\rho}(\boldsymbol{w})$ be the logistic regression (LR) estimate of $\rho(\boldsymbol{w})$. Logistic regression produces an estimated sufficient predictor $ESP = \hat{\alpha} + \hat{\boldsymbol{\beta}}^T \boldsymbol{w}$. Then

$$\hat{\rho}(\boldsymbol{w}) = \frac{e^{ESP}}{1 + e^{ESP}} = \frac{\exp(\hat{\alpha} + \hat{\boldsymbol{\beta}}^T \boldsymbol{w})}{1 + \exp(\hat{\alpha} + \hat{\boldsymbol{\beta}}^T \boldsymbol{w})}.$$

The *logistic regression discriminant rule* allocates $\boldsymbol{w}$ to group 1 if $\hat{\rho}(\boldsymbol{w}) \geq 0.5$ and allocates $\boldsymbol{w}$ to group 0 if $\hat{\rho}(\boldsymbol{w}) < 0.5$. Equivalently, the LR rule allocates $\boldsymbol{w}$ to group 1 if $ESP \geq 0$ and allocates $\boldsymbol{w}$ to group 0 if $ESP < 0$.

8) Let $Y_i = j$ if case i is in group j for $j = 0, 1$. Then a *response plot* is a plot of ESP versus Y_i (on the vertical axis) with $\hat{\rho}(\boldsymbol{x}) \equiv \hat{\rho}(ESP)$ added as a visual aid where $\boldsymbol{x}_i$ is the vector of predictors for case i. Also divide the ESP into J slices with approximately the same number of cases in each slice. Then compute the sample mean = sample proportion in slice s: $\hat{\rho}_s = \overline{Y}_s = \sum_s Y_i/m_s$ where m_s is the number of cases in slice s. Then plot the resulting step function as a visual aid. If n_0 and n_1 are the sample sizes of both groups and $n_i \geq 5p$, then the logistic regression model was useful if the step function of observed slice proportions scatter fairly closely about the logistic curve $\hat{\rho}(ESP)$. If the LR response plot is good, $n_0 \geq 5p$ and $n_1 \geq 5p$, then the LR rule is robust to nonnormality and the assumption of equal population dispersion matrices. Know how to tell a good LR response plot from a bad one.

9) Given LR output, as shown below in symbols and for a real data set, and given $\boldsymbol{x}$ to classify, be able to a) compute ESP, b) classify $\boldsymbol{x}$ in group 0 or group 1, c) compute $\hat{\rho}(\boldsymbol{x})$.

Label	Estimate	Std. Error	Est/SE	p-value
Constant	$\hat{\alpha}$	$se(\hat{\alpha})$	$z_{o,0}$	for Ho: $\alpha = 0$
x_1	$\hat{\beta}_1$	$se(\hat{\beta}_1)$	$z_{o,1} = \hat{\beta}_1/se(\hat{\beta}_1)$	for Ho: $\beta_1 = 0$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_p	$\hat{\beta}_p$	$se(\hat{\beta}_p)$	$z_{o,p} = \hat{\beta}_p/se(\hat{\beta}_p)$	for Ho: $\beta_p = 0$

```
Binomial Regression Kernel mean function = Logistic
Response = Status,Terms = (Bottom Left),Trials = Ones
Coefficient Estimates
Label       Estimate   Std. Error  Est/SE  p-value
Constant  -389.806     104.224     -3.740  0.0002
Bottom     2.26423     0.333233     6.795  0.0000
```

```
Left        2.83356     0.795601      3.562   0.0004
```

10) Suppose there is training data $\boldsymbol{x}_{ij}$ for $i = 1, ..., n_j$ for group j. Hence it is known that $\boldsymbol{x}_{ij}$ came from group j where there are $G \geq 2$ groups. Use the discriminant analysis method to classify the training data. If m_j of the n_j group j cases are correctly classified, then the *apparent error rate for group* j is $1 - m_j/n_j$. If $m_A = \sum_{j=1}^{G} m_j$ of the $n = \sum_{j=1}^{G} n_j$ cases were correctly classified, then the *apparent error rate* $\text{AER} = 1 - m_A/n$.

11) Get apparent error rates for `LDA,` and `QDA` with the following commands.

```
out2  <- lda(x,group)
1-mean(predict(out2,x)$class==group)

out3  <- qda(x,group)
1-mean(predict(out3,x)$class==group)
```

Get the AERs for the methods that use variables x_1, x_3, and x_7 with the following commands.

```
out <- lda(x[,c(1,3,7)],group)
1-mean(predict(out,x[,c(1,3,7)])$class==group)

out <- qda(x[,c(1,3,7)],group)
1-mean(predict(out,x[,c(1,3,7)])$class==group)
```

Get the AERs for the methods that leave out variables x_1, x_4, and x_5 with the following commands.

```
out <- lda(x[,-c(1,4,5)],group)
1-mean(predict(out,x[,-c(1,4,5)])$class==group)

out <- qda(x[,-c(1,4,5)],group)
1-mean(predict(out,x[,-c(1,4,5)])$class==group)
```

12) Expect the apparent error rate to be too low: the method works better on the training data than on the new test data to be classified.

13) Cross validation (CV): for $i = 1, ..., n$ where the training data has n cases, compute the discriminant rule with case i left out and see if the rule correctly classifies case i. Let m_C be the number of cases correctly classified. Then the CV error rate is $1 - m_C/n$.

14) Suppose the training data has n cases. Randomly select a subset L of n_v cases to be left out when computing the discriminant rule. Hence $n - n_v$ cases are used to compute the discriminant rule. Let m_L be the number of cases from subset L that are correctly classified. Then the "leave a subset out" error rate is $1 - m_L/n_v$. Here n_v should be large enough to get a good rate. Often use n_v between $0.1n$ and $0.5n$.

15) Variable selection is the search for a subset of variables that does a good job of classification.

16) Crude forward selection: suppose $X_1, ..., X_p$ are variables.

Step 1) Choose variable $W_1 = X_1$ that minimizes the AER.

Step 2) Keep W_1 in the model, and add variable W_2 that minimizes the AER. So W_1 and W_2 are in the model at the end of Step 2).

Step k) Have $W_1, ..., W_{k-1}$ in the model. Add variable W_k that minimizes the AER. So $W_1, ..., W_k$ are in the model at the end of Step k).

Step p) $W_1, ..., W_p = X_1, ..., X_p$, so all p variables are in the model.

17) Crude backward elimination: suppose $X_1, ..., X_p$ are variables.

Step 1) $W_1, ..., W_p = X_1, ..., X_p$, so all p variables are in the model.

Step 2) Delete variable $W_p = X_j$ such that the model with $p-1$ variables $W_1, ..., W_{p-1}$ minimizes the AER.

Step 3) Delete variable $W_{p-1} = X_j$ such that the model with $p-2$ variables $W_1, ..., W_{p-2}$ minimizes the AER.

Step k) $W_1, ..., W_{p-k+2}$ are in the model. Delete variable $W_{p-k+2} = X_j$ such that the model with $p-k+1$ variables $W_1, ..., W_{p-k+1}$ minimizes the AER.

Step p) Have W_1 and W_2 in the model. Delete variable W_2 such that the model with 1 variable W_1 minimizes the AER.

18) Other criterion can be used and `proc stepdisc` in *SAS* does variable selection.

19) In *R*, using LDA, leave one variable out at a time as long as the AER does not increase much, to find a good subset quickly.

5.12 Complements

This chapter followed Olive (2017c: ch. 8) closely. Discriminant analysis has a massive literature. James et al. (2013) and Hastie et al. (2009) discuss many other important methods such as trees, random forests, boosting, and support vector machines. Koch (2014, pp. 120-124) shows that Fisher's discriminant analysis is a generalized eigenvalue problem. James et al. (2013) has useful *R* code for fitting KNN. Cook and Zhang (2015) show that envelope methods have the potential to significantly improve standard methods of linear discriminant analysis.

Huberty and Olejnik (2006) and McLachlan (2004) are useful references for discriminant analysis. Silverman (1986, § 6.1) is a good reference for nonparametric discriminant analysis. Discrimination when $p > n$ is interesting. See Cai and Liu (2011) and Mai et al. (2012). See Friedman (1989) for regularized discriminant analysis.

A DA method for two groups can be extended to G groups by performing the DA method G times where $Y_{ij} = 1$ if $\boldsymbol{x}_{ij}$ is in the jth group and $Y_{ij} = 0$

if $\boldsymbol{x}_{ij}$ is not in the jth group for $j = 1, ..., G$. Then compute $\hat{\rho}_j = \hat{P}(\boldsymbol{w}$ is in the jth) group, and assign $\boldsymbol{w}$ to group a where $\hat{\rho}_a$ is a max.

There are variable selection methods for DA, and some implementations are needed in R, especially forward selection for when $p > n$. Witten and Tibshirani (2011) give a LASSO type FDA method useful for $p > n$. See the R package *penalizedLDA*. An outlier resistant version can be made using *getBbig* to find B_{big}. See Section 1.3 and Example 5.1.

Olive and Hawkins (2005) suggest that fast variable selection methods originally meant for multiple linear regression are also often effective for logistic regression when the C_p criterion is used. See Olive (2010: ch. 10, 2013b, 2017a: ch. 13) for more information about variable selection and response plots for logistic regression.

Hand (2006) notes that supervised classification is a research area in statistics, machine learning, pattern recognition, computational learning theory, and data mining. Hand (2006) argues that simple classification methods, such as linear discriminant analysis, are almost as good as more sophisticated methods such as neural networks and support vector machines.

5.13 Problems

PROBLEMS WITH AN ASTERISK * ARE ESPECIALLY USEFUL.

5.1*. Assume the cases in each of the G groups are iid from a population with covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{x}}(j)$ Find $E(\boldsymbol{S}_{pool})$ assuming that the k groups have the same covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{x}}(j) \equiv \boldsymbol{\Sigma}_{\boldsymbol{x}}$ for $j = 1, ..., G$.

```
Logistic Regression Output for Problem 5.2
Response = nodal involvement, Terms = (acid size xray)
Label      Estimate  Std. Error     Est/SE    p-value
Constant  -3.57564   1.18002        -3.030     0.0024
acid       2.06294   1.26441         1.632     0.1028
size       1.75556   0.738348        2.378     0.0174
xray       2.06178   0.777103        2.653     0.0080

Number of cases: 53, Degrees of freedom: 49,
Deviance: 50.660
```

5.2. Following Collett (1999, p. 11), treatment for prostate cancer depends on whether the cancer has spread to the surrounding lymph nodes. Let the response variable = group y = *nodal involvement* (0 for absence, 1 for presence). Let $x_1 = acid$ (serum acid phosphatase level), $x_2 = size$ (= tumor size: 0 for small, 1 for large) and $x_3 = xray$ (xray result: 0 for negative,

1 for positive). Assume the case to be classified has $\boldsymbol{x}$ with $x_1 = acid = 0.65$, $x_2 = 0$, and $x3 = 0$. Refer to the above output.

a) Find ESP for $\boldsymbol{x}$.
b) Is $\boldsymbol{x}$ classified in group 0 or group 1?
c) Find $\hat{\rho}(\boldsymbol{x})$.

5.3. Recall that X comes from a uniform(a,b) distribution, written $x \sim U(a, b)$, if the pdf of x is $f(x) = \dfrac{1}{b-a}$ for $a < x < b$ and $f(x) = 0$, otherwise. Suppose group 1 has $X \sim U(-3, 3)$, group 2 has $X \sim U(-5, 5)$, and group 3 has $X \sim U(-1, 1)$. Find the maximum likelihood discriminant rule for classifying a new observation x.

```
#Problem 5.4
out <- lda(state[,1:4],state[,5])
1-mean(predict(out,state[,1:4])$class==state[,5])
[1] 0.3
```

5.4. The above LDA output is for the Minor (2012) state data where gdp = GDP per capita, povrt = poverty rate, unins = 3 year average uninsured rate 2007-9, and lifexp = life expectancy for the 50 states. The fifth variable was a 1 if the state was not worker friendly and a 2 if the state was worker friendly. With these two groups, what was the apparent error rate (AER) for LDA?

```
> out <- lda(x,group) #Problem 5.5
> 1-mean(predict(out,x)$class==group)
[1] 0.02
>
> out<-lda(x[,-c(1)],group)
> 1-mean(predict(out,x[,-c(1)])$class==group)
[1] 0.02
> out<-lda(x[,-c(1,2)],group)
> 1-mean(predict(out,x[,-c(1,2)])$class==group)
[1] 0.04
> out<-lda(x[,-c(1,3)],group)
> 1-mean(predict(out,x[,-c(1,3)])$class==group)
[1] 0.03333333
> out<-lda(x[,-c(1,4)],group)
> 1-mean(predict(out,x[,-c(1,4)])$class==group)
[1] 0.04666667
>
> out<-lda(x[,c(2,3,4)],group)
> 1-mean(predict(out,x[,c(2,3,4)])$class==group)
[1] 0.02
```

5.5. The above output is for LDA on the famous iris data set. The variables are x_1 = sepal length, x_2 = sepal width, x_3 = petal length, and x_4 = petal

width. These four predictors are in the x data matrix. There are three groups corresponding to types of iris: setosa, versicolor, and virginica.

a) What is the AER using all 4 predictors?

b) Which variables, if any, can be deleted without increasing the AER in a)?

5.6.

```
Logistic Regression Output
Response = survival, Terms = (Age Vel)
Coefficient Estimates
Label      Estimate   Std. Error   Est/SE   p-value
Constant  -16.9845    5.14715      -3.300   0.0010
Age        0.162501   0.0414345     3.922   0.0001
Vel        0.233906   0.0862480     2.712   0.0067
```

The survival outcomes of 58 side-impact collisions using crash dummies was examined. $x_1 = age$ is the "age" of the crash dummy while $x_2 = vel$ was the velocity of the automobile at impact. The group = response variable *survival* was coded as a 1 if the accident would have been fatal, 0 otherwise. Assume the case to be classified has $\boldsymbol{x}$ with age $= x_1 = 60.0$ and velocity $= x_2 = 50.0$.

a) Find ESP for $\boldsymbol{x}$.

b) Is $\boldsymbol{x}$ classified in group 0 or group 1?

c) Find $\hat{\rho}(\boldsymbol{x})$.

5.7.

```
out <- lda(state[,1:4],state[,5])
1-mean(predict(out,state[,1:4])$class==state[,5])
[1] 0.3
```

The LDA output above is for the Minor (2012) state data where gdp = GDP per capita, povrt = poverty rate, unins = 3 year average uninsured rate 2007-9, and lifexp = life expectancy for the 50 states. The fifth variable Y was a 1 if the state was not worker friendly and a 2 if the state was worker friendly. With these two groups, what was the apparent error rate (AER) for LDA?

5.8.

```
> out <- lda(x,group)
> 1-mean(predict(out,x)$class==group)
[1] 0.02
>
> out<-lda(x[,-c(1)],group)
> 1-mean(predict(out,x[,-c(1)])$class==group)
[1] 0.02
> out<-lda(x[,-c(1,2)],group)
> 1-mean(predict(out,x[,-c(1,2)])$class==group)
```

```
[1] 0.04
> out<-lda(x[,-c(1,3)],group)
> 1-mean(predict(out,x[,-c(1,3)])$class==group)
[1] 0.03333333
> out<-lda(x[,-c(1,4)],group)
> 1-mean(predict(out,x[,-c(1,4)])$class==group)
[1] 0.04666667
>
> out<-lda(x[,c(2,3,4)],group)
> 1-mean(predict(out,x[,c(2,3,4)])$class==group)
[1] 0.02
```

The above output is for LDA on the famous iris data set. the variables are x_1 = sepal length, x_2 = sepal width, x_3 = petal length and x_4 = petal width. These four predictors are in the x data matrix. There are three groups corresponding to types of iris: setosa versicolor virginica.

a) What is the AER using all 4 predictors?

b) Which variables, if any, can be deleted without increasing the AER in a)?

5.9. The James et al. (2013) ISLR Default data set is simulated data for predicting which customers will default on their credit card debt. Let $Y = 1$ if the customer defaulted and $Y = -1$ otherwise. The predictors were $x_1 = Yes$ if the customer is a student and $X_1 = No$, otherwise, $x_2 = balance$ = the average monthly balance after the monthly payment, and $x_3 = income$ of the customer.

```
i) For SVM
        truth
predict   -1       1           AER =
     -1   9667     333
     1     0       0

ii) For bagging
        truth
predict   -1       1           AER =
     -1   9566     227
     1     101     106

iii) For random forests
         truth
predict   -1       1           AER =
     -1    9625    245
     1       42    88
```

a) Compute the error rate AER for each table.

b) Which method was worst for predicting a default?

5.10. This problem uses the Gladstone (1905) brain weight data and classifies gender (F for $y = -1$ or $z = 0$, M for $y = 1 = z$) using various predictors including head measurements, brain weight, and height. Some outliers were removed and the data set was divided into a training set with $n = 200$ cases and a test set with $m = 61$ cases. Compute the VER for each table.

```
        truth
predict -1    1
     -1  16   12      bagging VER =
     1    3   30

        truth
predict -1    1
     -1  15   13      random forest VER =
     1    4   29

       truth
predict  -1   1     (10-fold CV) SVM VER =
     -1  12   13
     1    7   29

       truth
predict  -1   1          LDA VER =
     -1  12   18
     1    7   24

        truth
predict  -1   1
     -1  17   21          QDA VER =
     1    2   21

        truth
predict  -1   1
     -1  14   14  (K = 7) KNN VER =
     1    5   28
```

R Problems

Warning: Use the command *source("G:/slpack.txt")* **to download the programs. See Preface or Section 8.1.** Typing the name of the `slpack` function, e.g. *ddplot*, will display the code for the function. Use the `args` command, e.g. *args(ddplot)*, to display the needed arguments for the function. For some of the following problems, the *R* commands can be copied and pasted from (http://parker.ad.siu.edu/Olive/slrhw.txt) into *R*.

5.11. The Wisseman et al. (1987) pottery data has 36 pottery shards of Roman earthware produced between second century B.C. and fourth century A.D. Often the pottery was stamped by the manufacturer. A chemical

analysis was done for 20 chemicals (variables), and 28 cases were classified as Arrentine (group 1) or nonArrentine (group 2), while 8 cases were of questionable origin. So the training data has $n = 28$ and $p = 20$.

a) Copy and paste the R commands for this part into R to make the data set.

b) Because of the small sample size, LDA should be used instead of QDA. Nonetheless, variable selection using QDA will be done. Copy and paste the R commands for this part into R. The first 9 variables result in no misclassification errors.

c) Now use commands like those shown in Example 5.2 to delete variables whose deletion does not result in a classification error. You should get four variables are needed for perfect classification. What are they (e.g. X1, X2, X3, and X4)?

5.12. Variable selection for LDA used the pottery data described in Problem 5.11, and suggested that variables X6, X11, X14, and X18 are good. Use the R commands for this problem to get the apparent error rate AER.

5.13. This problem uses KNN on the same data set as in Problem 5.11.

a) Copy and paste the commands for this part into R to show AER = 0 for KNN if $K = 1$.

b) Copy and paste the commands for this part into R to get the validation error rate for KNN if $K = 1$. Give the rate. The validation set has 12 cases and KNN is computed from the remaining 16 cases.

c) Use these commands to give the AER if $K = 2$.

d) Use these commands to give the validation ER if $K = 2$.

e) Use these commands to give the AER for 2NN using variables $X6, X11, X14$, and $X18$ that were good for LDA in Problem 5.7.

f) Use these commands to give the validation ER for 2NN using variables $X6, X11, X14$, and $X18$ that were good for LDA.

5.14. For the Gladstone (1905) data, the response variable Y = *gender*, gives the group (0-F, 1-M). The predictors are x_1 = *age*, x_2 = *log(age)*, x_3 = *breadth* of head, x_4 and x_5 are indicators for *cause* of death coded as a factor, x_6 = *cephalic index* (a head measurement), x_7 = *circumference* of head, x_8 = *height* of the head, x_9 = *height* of the person, x_{10} = *length* of head, x_{11} = *size* of the head, and x_{12} = *log(size)* of head. The sample size is $n = 267$.

a) The R code for this part does backward elimination for logistic regression. Backward elimination should only be used if $n \geq Jp$ with $J \geq 5$ and preferably $J \geq 10$.

Include the coefficients for the selected model (given by the `summary(back)` command) in *Word*. (You may need to do some editing to make the table readable.)

b) The R code for this part gives the response plot for the backward elimination submodel I_B. Does the response plot look ok?

c) Use the R code for this part to give the AER for I_B.

d) Use the R code for this part to give a validation ER for I_B.

(Another validation ER would apply backward elimination on the cases not in the validation set. We just used the variables from the backward elimination model selected using the full data set. The first method is likely superior, but the second method is easier to code.)

e) These *R* commands will use lasso with a classification criterion. We got rid of the factor (two indicator variables) since `cv.glmnet` uses a matrix of predictors. Lasso can handle indicators like gender as a response variable, but will not keep or delete groups two or more indicators that are needed for a quantitative variable with 3 or more levels. These commands give the k-fold CV error rate for the lasso logistic regression. What is it?

f) Use the commands for this part to get the relaxed lasso response plot where relaxed lasso uses the lasso from part e). Include the plot in *Word*.

g) Use the commands from this plot to make the EE plot of the ESP from relaxed lasso (ESPRL) versus the ESP from lasso (ESPlasso).

5.15. This problem creates a classification tree. The vignette Therneau and Atkinson (2017) and book MathSoft (1999b) were useful. The dataset has $n = 81$ children who have had corrective spinal surgery. The variables are $Y = Kyphosis$: postoperative deformity is present/absent, and predictors $x_1 = Age$ of child in months, $x_n = Number$ vertebrae involved in the operation, and $Start$ = beginning of the range of vertebrae involved.

a) Use the *R* code for this part to print the classification tree. Then predict whether $Y = absent$ or $Y = present$ if $Start = 13$ and $Age = 25$.

b) Then predict whether $Y = absent$ or $Y = present$ if $Start = 10$ and $Age = 120$. Note that you go to the left of the tree branch if the label condition is true, and to the right of the tree branch if the label condition is not true.

5.16. This is the pottery data of Problem 5.11, but the 28 cases were classified as Arrentine for $y = -1$ and nonArrentine for $y = 1$.

a) Copy and paste the commands for this part into *R*. These commands make the data and do bagging. Copy and paste the truth table into *Word*. What is the AER?

b) Copy and paste the commands for this part into *R*. These commands do random forests. Copy and paste the truth table into *Word*. What is the AER?

c) Copy and paste the commands for this part into *R*. These commands do SVM with a fixed cost. Copy and paste the truth table into *Word*. What is the AER?

d) Copy and paste the commands for this part into *R*. These commands do SVM with a cost chosen by 10-fold CV. Copy and paste the truth table into *Word*. What is the AER?

5.17. This problem uses the Gladstone (1905) brain weight data and classifies gender (F for $y = -1$, M for $y = 1$) using various predictors including head measurements, brain weight, and height. Some outliers were removed

and the data set was divided into a training set with $n = 200$ cases and a test set with $m = 61$ cases.

a) Copy and paste the commands for this part into *R*. These commands make the data and do bagging. Copy and paste the truth table into *Word*. What is the AER?

b) Copy and paste the commands for this part into *R*. These use bagging on the training data and validation set. Copy and paste the truth table into *Word*. What is the bagging validation error rate?

c) Copy and paste the commands for this part into *R*. These commands do random forests. Copy and paste the truth table into *Word*. What is the AER?

d) Copy and paste the commands for this part into *R*. These use random forests on the training data and validation set. Copy and paste the truth table into *Word*. What is the random forests validation error rate?

e) Copy and paste the commands for this part into *R*. These commands do SVM with a cost chosen by 10-fold CV. Copy and paste the truth table into *Word*. What is the AER?

f) Copy and paste the commands for this part into *R*. These commands do SVM with a cost chosen by 10-fold CV on the training data and validation set. Copy and paste the truth table into *Word*. What is the SVM validation error rate?