Outline

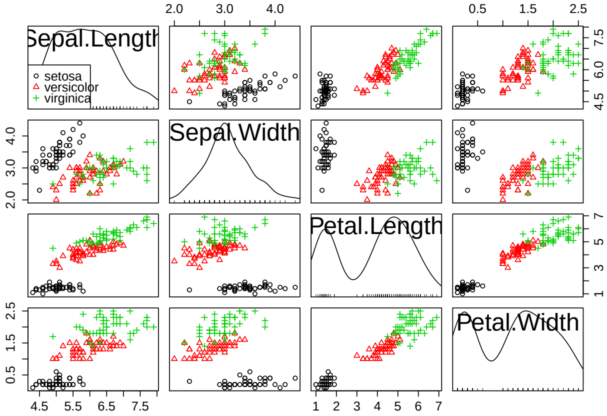
Classification and Discriminant Analysis

- Data: $\{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, n\}$ with multivariate observations $\boldsymbol{x}_i \in \mathbb{R}^p$ (continuous) and population labels or class information $y_i = 1, \ldots, K$ (categorical).
- Assume $\boldsymbol{X} \mid (Y = k) \;\sim\; F_k$, for different distributions F_k.
- Binary classification: If there are only two cases, may write $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{n_1} \sim$ i.i.d. F_1 and $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_{n_2} \sim$ i.i.d. F_2.
- Classification aims to classify a new observation, or several new observations into one of those classes (groups, populations).
- A *classifier* (classification rule) is a function $\phi(\boldsymbol{x}) : \mathcal{X} \to \{1, \ldots, K\}$.

The next section would be . . . . . .

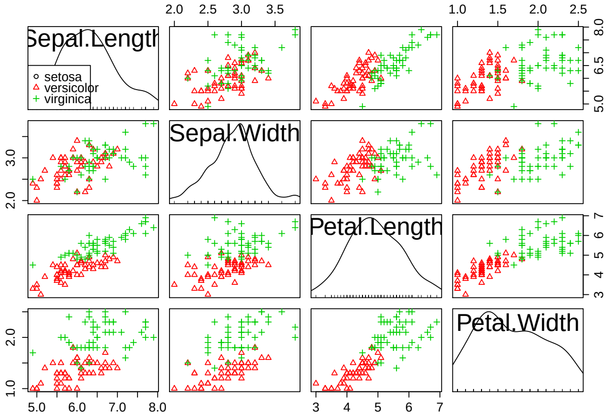
Example: Fisher's iris data

- Fisher's iris dataset. Classification of flowers into different species (setosa, versicolor, virginica) based on lengths and widths of sepal and petal (4 variables). $n = 150$ observations



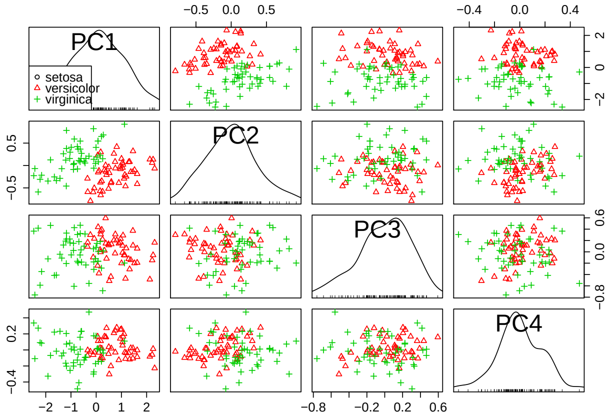
Example: Fisher's iris data

Focus on the latter two labels (red and green).

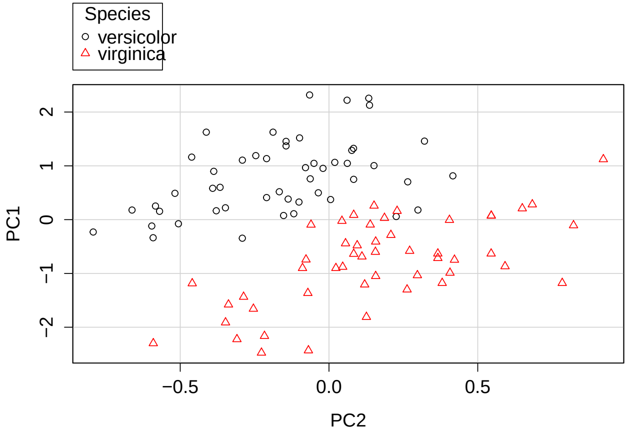


Example: Fisher's iris data

Try dimension reduction. Focus on the first two principal component scores.



Example: Fisher's iris data

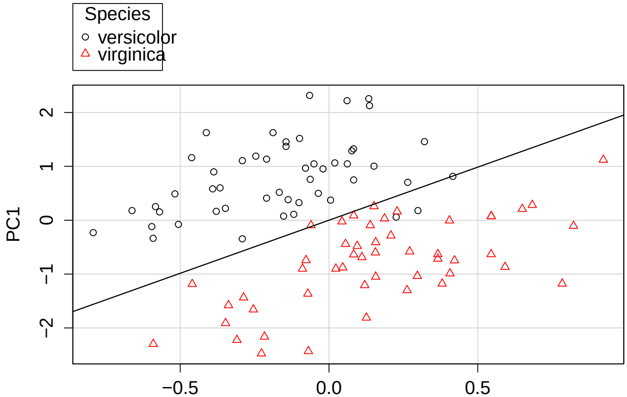


Example: Fisher's iris data

An example of classifier given by a linear hyperplane.
$\phi(\boldsymbol{z}) : \mathbb{R}^2 \to \{\text{versicolor}, \text{virginica}\}$, where

$$\phi(\boldsymbol{z}) = \begin{cases} \text{versicolor}, & \boldsymbol{b}'\boldsymbol{z} < 0; \\ \text{virginica}, & \boldsymbol{b}'\boldsymbol{z} \geq 0, \end{cases} \qquad \boldsymbol{b} = (-1.55, 3.06)'$$

assuming no intercept



Example: Classifiers

The previous example on classifying Fisher's iris data is an example of linear classifier. A linear classifier $\phi(\boldsymbol{x})$ is a function of linear combinations of input vector $\boldsymbol{x}$ and is of the form

$$\phi(\boldsymbol{x}) = h(b_0 + \boldsymbol{b}'\boldsymbol{x}).$$

In binary classification ($K = 2$), the linear classifier may be written as

$$\phi(\boldsymbol{x}) = \text{sign}(b_0 + \boldsymbol{b}'\boldsymbol{x}) :$$

$$b_0 + \boldsymbol{b}'\boldsymbol{x} > 0 \Rightarrow +1 \text{ Class}; \;\; b_0 + \boldsymbol{b}'\boldsymbol{x} < 0 \Rightarrow -1 \text{ Class}$$

$\mathbb{R}^p$ is divided by a hyperplane $\left\{\boldsymbol{x} : b_0 + \boldsymbol{b}'\boldsymbol{x} = 0\right\}$
Moreover, a classifier may be quadratic,

$$\phi(\boldsymbol{x}) = h(b_0 + \boldsymbol{b}'\boldsymbol{x}, \boldsymbol{x}'\boldsymbol{C}\boldsymbol{x}),$$

and beyond.

The next section would be ......

Setup

Assume $K = 2$ for simplicity

$$\boldsymbol{X} \mid (Y = 1) \sim f_1(\boldsymbol{x}), \quad \boldsymbol{X}|(Y = 2) \sim f_2(\boldsymbol{x}).$$

Denote a random observation from this population (a mixture of two populations) by $(\boldsymbol{X}, Y)$. We assume

$$P(Y = 1) = \pi_1, \quad P(Y = 2) = \pi_2, \quad \pi_1 + \pi_2 = 1$$

If the observed value of $\boldsymbol{X}$ is $\boldsymbol{x}$, then Bayes theorem yields the posterior probability that the observed $\boldsymbol{x}$ was from the population 1:

$$\eta(\boldsymbol{x}) := P(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x}) = \frac{f_1(\boldsymbol{x})\pi_1}{f_1(\boldsymbol{x})\pi_1 + f_2(\boldsymbol{x})\pi_2}.$$

0-1 Loss and Bayes (decision) rule

- Use a decision theory framework, if we care about the 0-1 loss: $\mathbb{1}_{\{Y \neq \delta(\boldsymbol{X})\}}$, we would hope to choose a decision function to minimize the risk associated with the 0-1 loss.

$$\mathrm{E}[\mathbb{1}_{\{Y \neq \delta(\boldsymbol{X})\}}] = \Pr[Y \neq \delta(\boldsymbol{X})]$$

- We only need to choose a best decision $\delta(\boldsymbol{x})$ for each given $\boldsymbol{X} = \boldsymbol{x}$, that minimizes $\Pr[Y \neq \delta(\boldsymbol{x}) \mid \boldsymbol{X} = \boldsymbol{x}]$
- Rewrite

$$\begin{aligned} &\Pr[Y \neq \delta(\boldsymbol{x}) \mid \boldsymbol{X} = \boldsymbol{x}] \\ =&\Pr[1 \neq \delta(\boldsymbol{x}) \mid \boldsymbol{X} = \boldsymbol{x}, Y = 1] P(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x}) \\ &+ \Pr[2 \neq \delta(\boldsymbol{x}) \mid \boldsymbol{X} = \boldsymbol{x}, Y = 1] P(Y = 2 \mid \boldsymbol{X} = \boldsymbol{x}) \end{aligned}$$

- The blue parts are either 0 or 1. Hence we only need to choose $\delta(\boldsymbol{x})$ to be the class j with greater $P(Y = j \mid \boldsymbol{X} = \boldsymbol{x})$

Bayes Rule Classifier for Gaussian

The derivation from the last page is how we find the Bayes (decision rule): Bayes Rule classifier assigns the class label (1 or 2) which gives the higher posterior probability:

$$\phi_{\text{Bayes}}(\boldsymbol{x}) = \underset{k=1,2}{\operatorname{argmax}} P(Y = k \mid \boldsymbol{X} = \boldsymbol{x}).$$

Now assume Gaussian data:

$$\boldsymbol{X} \mid (Y = 1) \sim N_p(\mu_1, \Sigma), \quad \boldsymbol{X} | (Y = 2) \sim N_p(\mu_2, \Sigma), \quad \mu_1 \neq \mu_2.$$

Recall that

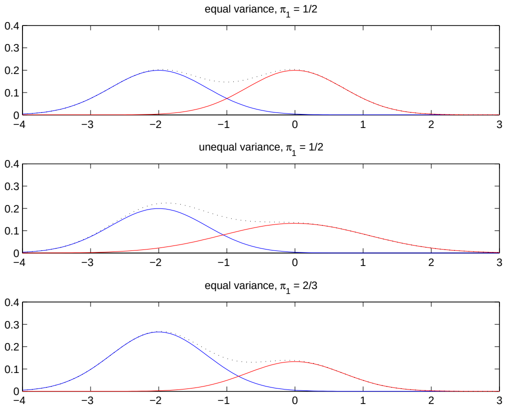
$$\eta(\boldsymbol{x}) := P(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x}) = \frac{f_1(\boldsymbol{x})\pi_1}{f_1(\boldsymbol{x})\pi_1 + f_2(\boldsymbol{x})\pi_2}.$$

Bayes Rule Classifier: Gaussian 1-D

As a special case, assume $p = 1$ and

$$X|(Y = 1) \sim N_1(\mu_1, \sigma_1^2), \quad X|(Y = 2) \sim N_1(\mu_2, \sigma_2^2), \quad \mu_1 \neq \mu_2.$$

Then $P(Y = i \mid X = x) \propto \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp(-\frac{(x-\mu_i)^2}{2\sigma_i^2})\pi_i$



Bayes Rule Classifier: Gaussian 1-D

The Bayes classifier classifier classify a point x into class 1 (blue)

1. case I ($\sigma_1 = \sigma_2, \pi_1 = \pi_2$): if

$$|x - \mu_1| < |x - \mu_2|.$$

2. case II ($\sigma_1 < \sigma_2, \pi_1 = \pi_2$): if

$$(\frac{x - \mu_1}{\sigma_1^2})^2 < (\frac{x - \mu_2}{\sigma_2^2})^2 + \log(\sigma_2^2/\sigma_1).$$

3. case III ($\sigma_1 = \sigma_2, \pi_1 > \pi_2$): if

$$(\frac{x - \mu_1}{\sigma_1^2})^2 < (\frac{x - \mu_2}{\sigma_2^2})^2 + 2\log(\pi_1/\pi_2).$$

More General Bayes Rule Classifier

In general, for $k = 1, \ldots, K > 2$. If the observed value of $\boldsymbol{X}$ is $\boldsymbol{x}$, then Bayes theorem yields the posterior probability that the observed $\boldsymbol{x}$ was from the population k:

$$P(Y = k \mid \boldsymbol{X} = \boldsymbol{x}) = \frac{f_k(\boldsymbol{x})\pi_k}{f(\boldsymbol{x})},$$

where f_k is the conditional density function of $\boldsymbol{X} \mid (Y = k)$. Bayes Rule classifier assigns the class label (among $1, \ldots, K$) which gives highest posterior probability:

$$\phi_{\text{Bayes}}(\boldsymbol{x}) = \operatorname*{argmax}_{k=1,\ldots,K} P(Y = k \mid \boldsymbol{X} = \boldsymbol{x}).$$

Gaussian Example:

$$\boldsymbol{X} \mid (Y = k) \sim N_p(\mu_k, \Sigma_k), \quad k = 1, \ldots, K,$$

$$P(Y = k) = \pi_k, \quad \sum_{k=1}^{K} \pi_k = 1,$$

$\boldsymbol{X} \sim f(\boldsymbol{x})$, a mixture density for multivariate normals

More General Bayes Rule Classifier for Gaussian Data

See that

$$\phi_{\text{Bayes}}(\boldsymbol{x}) = k \text{ iff } P(Y = k \mid \boldsymbol{X} = \boldsymbol{x}) \geq P(Y = i \mid \boldsymbol{X} = \boldsymbol{x}) \text{ for all } i,$$

which is equivalent to

$$\frac{1}{2}(\boldsymbol{x} - \mu_k)'\Sigma_k^{-1}(\boldsymbol{x} - \mu_k) + \frac{1}{2}\log|\Sigma_k| - \log(\pi_k) \leq \frac{1}{2}(\boldsymbol{x} - \mu_i)'\Sigma_i^{-1}(\boldsymbol{x} - \mu_i) + \frac{1}{2}\log|\Sigma_i| - \log(\pi_i) \text{ for all } i.$$

Moreover, $\phi_{\text{Bayes}}(\boldsymbol{x}) = \operatorname{argmin}_{k=1,\ldots,K} \delta_k(\boldsymbol{x})$, where

$$\delta_k(\boldsymbol{x}) = b_{0k} + \boldsymbol{b}_k'\boldsymbol{x} + \boldsymbol{x}'\boldsymbol{C}_k\boldsymbol{x},$$

$$\text{where } b_{0k} = \frac{1}{2}\mu_k'\Sigma_k^{-1}\mu_k + \frac{1}{2}\log|\Sigma_k| - \log(\pi_k),$$

$$\boldsymbol{b}_k = -\Sigma_k^{-1}\mu_k, \ \boldsymbol{C}_k = \frac{1}{2}\Sigma_k^{-1}.$$

Sample Bayes Rule Classifier for Gaussian Data

In practice, we do not know the parameters (μ_k, Σ_k, π_k). Given n observations $(\boldsymbol{x}_{ij}, i)$, $(i = 1, \ldots, K), (j = 1, \ldots, n_k)$, $n = \sum_{k=1}^{K} n_k$, sample versions of the classifiers are obtained by substituting μ_k with $\hat{\mu}_k = \bar{\boldsymbol{x}}_k$, Σ_k with $\widehat{\Sigma}_k = \boldsymbol{S}_k$. Also $\hat{\pi}_k = n_k/n$.
The moment estimate of the Bayes rule classifier for Gaussian data is then $\phi(\boldsymbol{x}) = \mathrm{argmin}_{k=1,\ldots,K}\, \hat{\delta}_k(\boldsymbol{x})$, where

$$\hat{\delta}_k(\boldsymbol{x}) = b_{0k} - \boldsymbol{b}_k' \boldsymbol{x} + \boldsymbol{x}' \boldsymbol{C}_k \boldsymbol{x},$$

$$\text{where } b_{0k} = \frac{1}{2} \bar{\boldsymbol{x}}_k' \boldsymbol{S}_k^{-1} \bar{\boldsymbol{x}} + \frac{1}{2} \log |\boldsymbol{S}_k| - \log(n_k/n),$$

$$\boldsymbol{b}_k = \boldsymbol{S}_k^{-1} \bar{\boldsymbol{x}}_k, \;\; \boldsymbol{C}_k = \frac{1}{2} \boldsymbol{S}_k^{-1}.$$

Mahalanobis distance

In a special case where

$\Sigma_k \equiv \Sigma, \pi_k = 1/K$, for all k:

The Bayes rule classifier boils down to comparing the quantity $d_M^2(\boldsymbol{x}, \mu_k) = (\boldsymbol{x} - \mu_k)'\Sigma^{-1}(\boldsymbol{x} - \mu_k)$, which is called (squared) Mahalanobis distance.

- Mahalanobis distance $d_M(\boldsymbol{x}, \mu)$ measures how much $\boldsymbol{x}$ is away from the center of distribution $N_p(\mu, \Sigma)$.
- The set of points with the same Mahalanobis distance is an ellipsoid.
- Replacing Σ with its estimator $\boldsymbol{S}$, and $\boldsymbol{x}$ with the sample mean $\bar{\boldsymbol{x}}$, the squared Mahalanobis distance is proportional to Hotelling's T^2 statistic.

Special cases of Bayes Rule Classifier for Gaussian Data

Equal covariance $\Sigma_k \equiv \Sigma$:
$\phi_{\text{Bayes}}(\boldsymbol{x}) = \operatorname{argmin}_{k=1,2} \delta_k(\boldsymbol{x})$, where

$$\delta_k(\boldsymbol{x}) = b_{0k} - \boldsymbol{b}_k' \boldsymbol{x},$$
$$\text{where } b_{0k} = \frac{1}{2}\mu_k' \Sigma^{-1} \mu_k - \log(\pi_k),$$
$$\boldsymbol{b}_k = \Sigma^{-1}\mu_k$$

For Equal covariance $\Sigma_k \equiv \Sigma$ and binary classification $(K = 2)$:
$\phi_{\text{Bayes}}(\boldsymbol{x}) = 1$ when

$$(\mu_2 - \mu_1)' \Sigma^{-1} (\boldsymbol{x} - \frac{\mu_1 + \mu_2}{2}) < \log(\pi_1/\pi_2),$$

or

$$\boldsymbol{v}'\boldsymbol{x} - \boldsymbol{v}'\bar{\mu} < \log(\pi_1/\pi_2), \quad \boldsymbol{v} = \Sigma^{-1}(\mu_2 - \mu_1).$$

Sample Bayes Rule Classifier for Gaussian Data (with equal Cov. mat.)

In practice, we do not know the parameters (μ_k, Σ, π_k). Given n observations $(\boldsymbol{x}_{ij}, i)$, $(i = 1, \ldots, K), (j = 1, \ldots, n_k)$, $n = \sum_{k=1}^{K} n_k$, a sample version of the classifiers are obtained by substituting μ_k with $\hat{\mu}_k = \bar{\boldsymbol{x}}_k$, Σ with $\widehat{\Sigma} = \boldsymbol{S}_P$ (pooled sample covariance matrix). Also $\hat{\pi}_k = n_k/n$.
The moment estimate of the Bayes rule classifier with the equal covariance assumption is then $\phi(\boldsymbol{x}) = \mathrm{argmin}_{k=1,\ldots,K}\, \hat{\delta}_k(\boldsymbol{x})$, where

$$\begin{aligned} \hat{\delta}_k(\boldsymbol{x}) &= b_{0k} - \boldsymbol{b}_k'\boldsymbol{x}, \\ \text{where } b_{0k} &= \frac{1}{2}\bar{\boldsymbol{x}}_k'\boldsymbol{S}_P^{-1}\bar{\boldsymbol{x}}_k - \log(n_k/n), \\ \boldsymbol{b}_k &= \boldsymbol{S}_P^{-1}\bar{\boldsymbol{x}}_k. \end{aligned}$$

So far, we have developed TWO classifiers for Gaussian Data

Linear Discriminant Analysis

(Sample) Bayes rule classifier with equal covariance. For binary classification,

$$\phi(\boldsymbol{x}) = 1 \text{ if } \boldsymbol{b}'(\boldsymbol{x} - \frac{\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2}{2}) < \log(\frac{n_1}{n_2}), \quad \boldsymbol{b} = \boldsymbol{S}_P^{-1}(\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1).$$

In general,

$$\phi(\boldsymbol{x}) = \operatorname*{argmax}_k (b_{0k} - \boldsymbol{b}_k'\boldsymbol{x}), \quad \boldsymbol{b}_k = \boldsymbol{S}_P^{-1}\bar{\boldsymbol{x}}_k.$$

Quadratic Discriminant Analysis

(Sample) Bayes rule classifier with unequal covariance (see pp 19).

$$\phi(\boldsymbol{x}) = \operatorname*{argmax}_k (b_{0k} - \boldsymbol{b}_k'\boldsymbol{x} + \boldsymbol{x}'\boldsymbol{C}_k\boldsymbol{x}).$$

Fisher's LDA

- LDA (the sample estimate of Bayes rule classifier for Gaussian data with equal covariance) is often referred to as `R.A. Fisher's` Linear Discriminant Analysis.
- His original work did not involve any distributional assumption, and develops LDA through a geometric understanding of PCA.
- The LDA direction $\boldsymbol{u}_0 \in \mathbb{R}^p$ is a direction vector orthogonal to the separating hyperplane, and is found by maximizing the between-group variance while minimizing the within-group variance of the projected scores.

$$\boldsymbol{u}_0 = \underset{\boldsymbol{u} \in \mathbb{R}^p}{\operatorname{argmax}} \frac{(\boldsymbol{u}'\bar{\boldsymbol{x}}_1 - \boldsymbol{u}'\bar{\boldsymbol{x}}_2)^2}{\boldsymbol{u}'\boldsymbol{S}_P\boldsymbol{u}}$$

Since $\frac{(\boldsymbol{u}'\bar{\boldsymbol{x}}_1 - \boldsymbol{u}'\bar{\boldsymbol{x}}_2)^2}{\boldsymbol{u}'\boldsymbol{S}_P\boldsymbol{u}} = \frac{\boldsymbol{u}'(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)'\boldsymbol{u}}{\boldsymbol{u}'\boldsymbol{S}_P\boldsymbol{u}}$, from Theorem 2.5 in HS, we have

$$\boldsymbol{u}_0 = e.v.1\{S_P^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)'\}$$

Note that

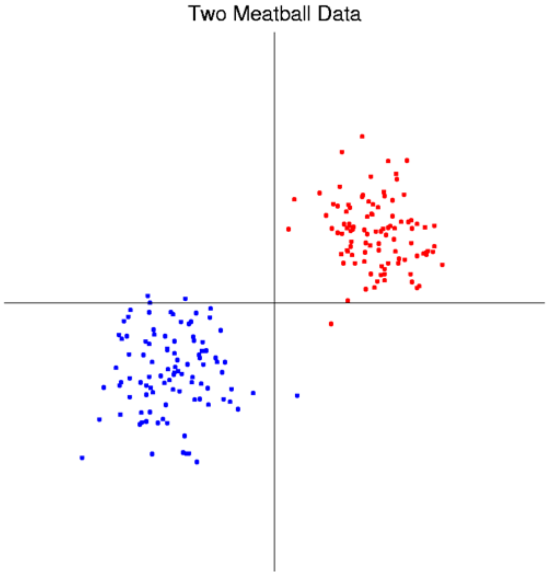
$$\left[S_P^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)'\right] S_P^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2) = \lambda S_P^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)$$

where $\lambda = (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)'S_P^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)$ which happens to be the greatest eigenvalue of $\left[S_P^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)'\right]$. Hence the solution of $\boldsymbol{u}_0$ is actually

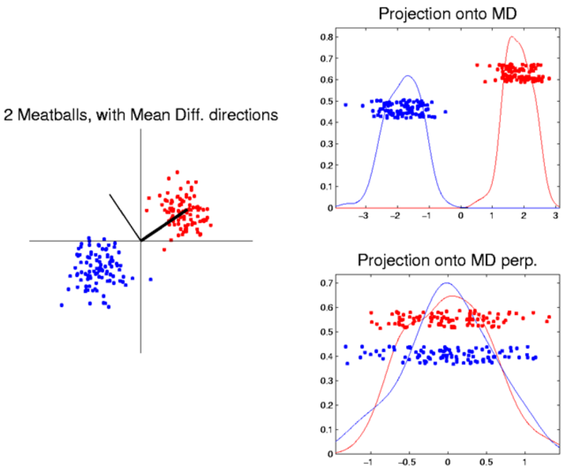
$$\boldsymbol{u}_0 = S_P^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)$$

Fisher's LDA–Geometric understanding

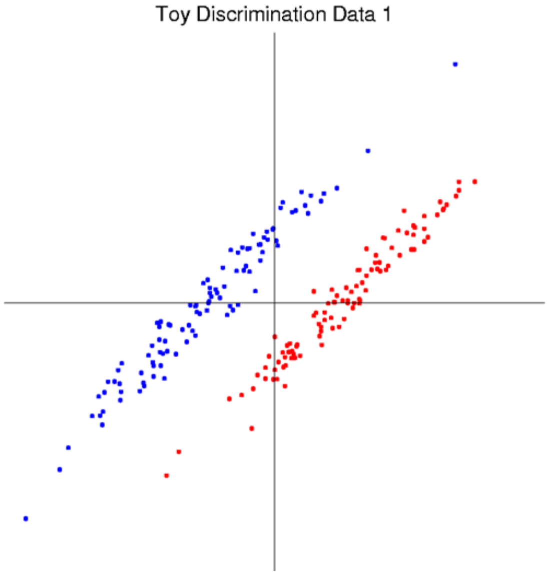
Two point clouds, each with $\boldsymbol{S}_i = \mathbb{I}_2$. $\boldsymbol{u}_0 \propto \boldsymbol{b} = \boldsymbol{S}_P^{-1}(\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1)$.



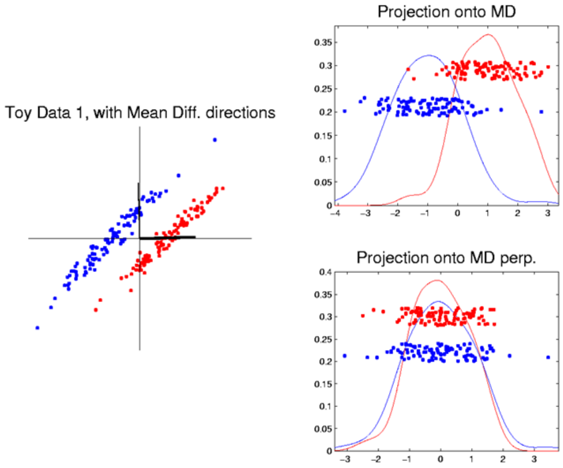
$$\boldsymbol{u}_0 \propto \boldsymbol{S}_P^{-1}(\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1) = (\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1).$$ (direction of mean difference)



Slanted clouds. Assumed to have equal covariance.

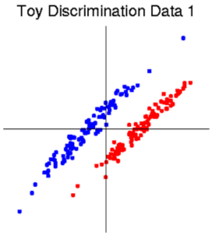


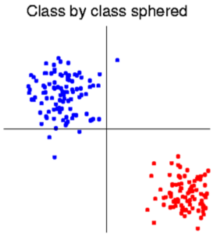
Mean difference direction not efficient, as $\boldsymbol{S}_P \neq c\mathbb{I}_2$.



Individually transform subpopulations so that both are '*spherical*' about their means.

$$\boldsymbol{y}_{ij} = \boldsymbol{S}_P^{-1/2} \boldsymbol{x}_{ij}.$$





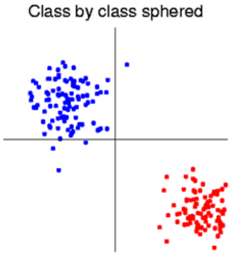
In transformed space, best separating hyperplane is the perpendicular bisector of line between means.
Transformed normal vector $\boldsymbol{b}_Y$:

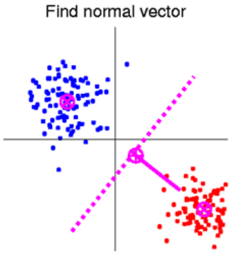
$$\bar{\boldsymbol{y}}_2 - \bar{\boldsymbol{y}}_1 = \boldsymbol{S}_P^{-1/2}(\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1).$$

Transformed intercept $\boldsymbol{b}_{0(Y)}$:

$$(\bar{\boldsymbol{y}}_1 + \bar{\boldsymbol{y}}_2)/2 = \boldsymbol{S}_P^{-1/2}(\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2)/2.$$

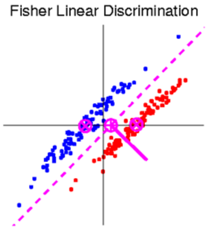
Transformed input $\boldsymbol{y} = \boldsymbol{S}_P^{-1/2}\boldsymbol{x}$ is classified to 1 if $\boldsymbol{b}_Y'(\boldsymbol{y} - \boldsymbol{b}_{0(Y)}) < 0$

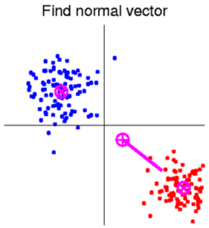




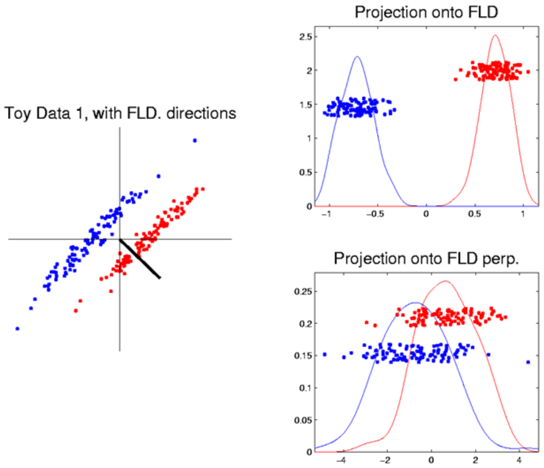
Original input $\boldsymbol{x} = \boldsymbol{S}_P^{1/2}\boldsymbol{y}$ is classified to 1 if

$$\begin{aligned}
&\boldsymbol{b}_Y'(\boldsymbol{y} - \boldsymbol{b}_{0(Y)}) < 0 \\
\Leftrightarrow\ &\boldsymbol{S}_P^{-1/2}(\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1)'(\boldsymbol{S}_P^{-1/2}\boldsymbol{x} - \boldsymbol{S}_P^{-1/2}(\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2)/2) < 0 \\
\Leftrightarrow\ &\boldsymbol{S}_P^{-1}(\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1)'(\boldsymbol{x} - (\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2)/2) < 0
\end{aligned}$$





Leads to Fisher's LDA ($\boldsymbol{u}_0 \propto \boldsymbol{S}_P^{-1}(\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1)$) by actively using covariance structure.



LDA vs QDA – Examples

In the next four sets of examples,

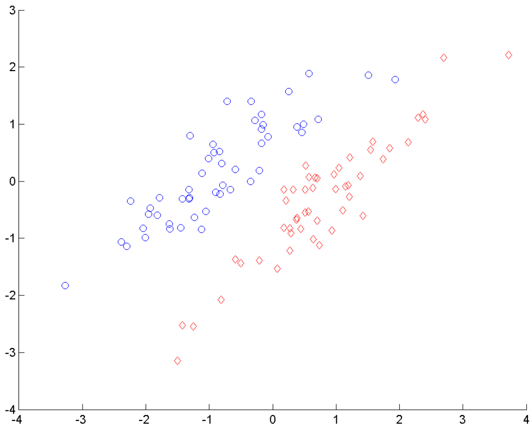
- Blue and red points represent observations from two different populations.
- Blue line is the *separating hyperplane* given by computing the sample LDA, and is a line perpendicular to *LDA direction* $\boldsymbol{b} = \boldsymbol{S}_P^{-1}(\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1)$, and is the set

$$\{\boldsymbol{x} \in \mathbb{R}^2 : \boldsymbol{b}'(\boldsymbol{x} - \frac{\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2}{2}) = \log(\frac{n_1}{n_2})\}.$$

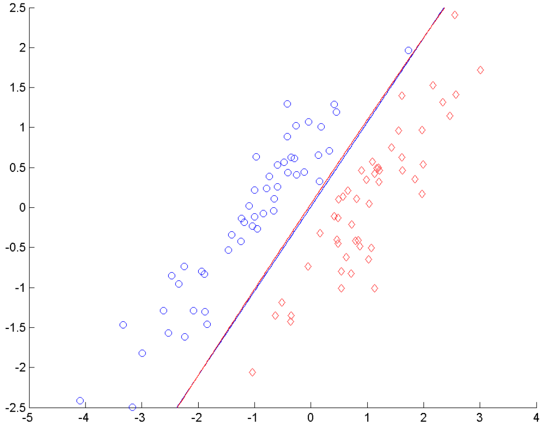
- Red curve represent the boundary of classification regions given by the sample QDA, and is

$$\{\boldsymbol{x} \in \mathbb{R}^2 : b_{01} - \boldsymbol{b}_1'\boldsymbol{x} + \boldsymbol{x}'\boldsymbol{C}_1\boldsymbol{x} = b_{02} - \boldsymbol{b}_2'\boldsymbol{x} + \boldsymbol{x}'\boldsymbol{C}_2\boldsymbol{x}\}.$$

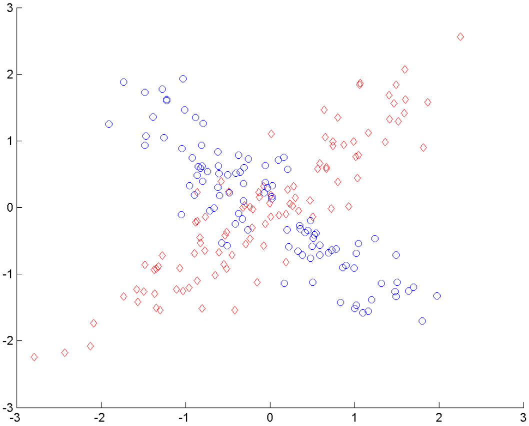
LDA vs QDA – Ex.1



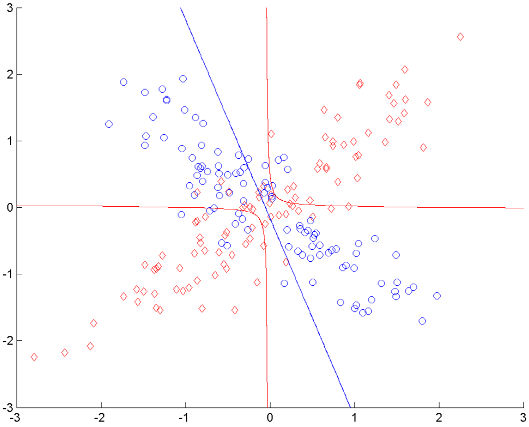
LDA vs QDA – Ex.1



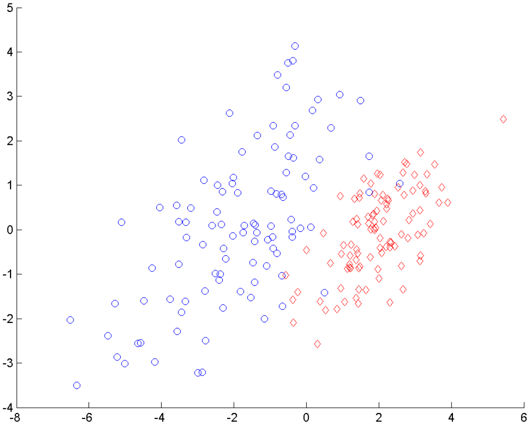
LDA vs QDA – Ex.2



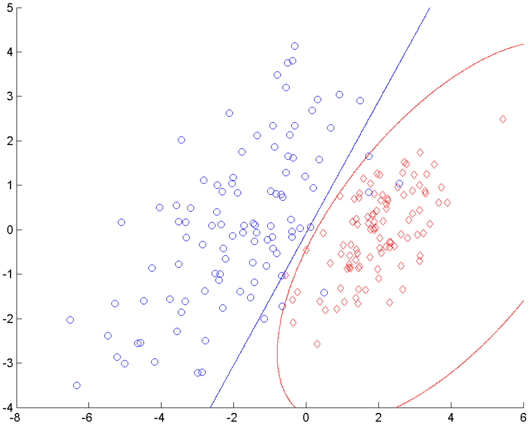
LDA vs QDA – Ex.2



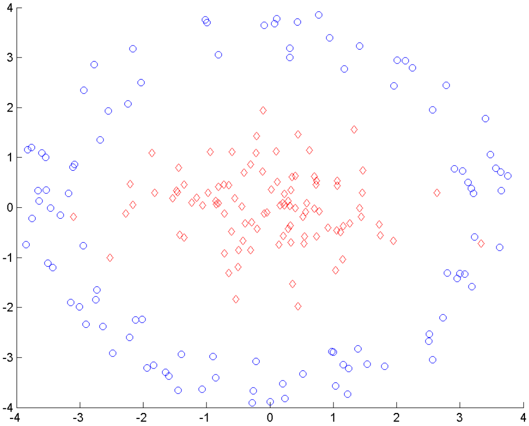
LDA vs QDA – Ex.3



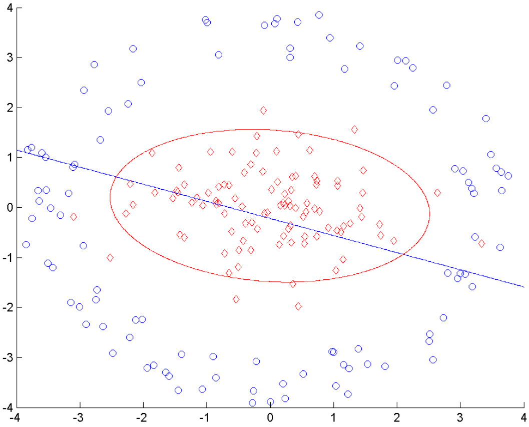
LDA vs QDA – Ex.3



LDA vs QDA – Ex.4



LDA vs QDA – Ex.4



The next section would be . . . . . .

Logistic Regression – model

- Logistic Regression is another model based method (we need to impose a model to the underlying distribution.)
- Binary classification case. Assume that $Y = 0$ or 1, esp. the occurrence of an event.
- **Model:**

$$\operatorname{logit}(\eta(\boldsymbol{x})) := \log\left(\frac{\eta(\boldsymbol{x})}{1-\eta(\boldsymbol{x})}\right) = b + \boldsymbol{\beta}'\boldsymbol{x} = f(\boldsymbol{x}),$$

  where $\eta(\boldsymbol{x}) = \Pr(Y = 1 \mid X = \boldsymbol{x})$ and $\eta(\boldsymbol{x}) = \Pr(Y = 0 \mid X = \boldsymbol{x})$ and $\boldsymbol{\beta} \in \mathbb{R}^p$ is the coefficient vector (similar to $\boldsymbol{b}$ in the previous section.)
- Can show that $\eta(\boldsymbol{x}) = \frac{\exp(f(\boldsymbol{x}))}{1+\exp(f(\boldsymbol{x}))}$ and $1 - \eta(\boldsymbol{x}) = \frac{1}{1+\exp(f(\boldsymbol{x}))}$.

Conditional likelihood

- Data: $\{(\boldsymbol{x}_i, y_i), i = 1, \ldots, n\}$
- Given $\boldsymbol{X}_i = \boldsymbol{x}_i$, the Y_i is a Bernoulli random variable (conditionally) with parameter $\eta(\boldsymbol{x})$. [recall that $\eta(\boldsymbol{x})$ depends on b and $\boldsymbol{\beta}$.]
- Conditional likelihood of $(b, \boldsymbol{\beta})$ is

$$\prod_{i=1}^{n} \eta(\boldsymbol{x}_i; b, \boldsymbol{\beta})^{y_i} [1 - \eta(\boldsymbol{x}_i; b, \boldsymbol{\beta})]^{1-y_i}$$

- Conditional log-likelihood of $(b, \boldsymbol{\beta})$ is

$$\begin{aligned} \ell(b, \boldsymbol{\beta}) &:= \sum_{i=1}^{n} \{y_i \log(\eta(\boldsymbol{x}_i; b, \boldsymbol{\beta})) + (1 - y_i) \log[1 - \eta(\boldsymbol{x}_i; b, \boldsymbol{\beta})]\} \\ &= \sum_{i=1}^{n} \{y_i \log\left(\frac{\exp(f(\boldsymbol{x}_i))}{1 + \exp(f(\boldsymbol{x}_i))}\right) + (1 - y_i) \log\left[\frac{1}{1 + \exp(f(\boldsymbol{x}_i))}\right]\} \end{aligned}$$

$$\begin{aligned}
\ell(b, \boldsymbol{\beta}) &:= \sum_{i=1}^{n} \{y_i \log(\eta(\boldsymbol{x}_i; b, \boldsymbol{\beta})) + (1 - y_i) \log[1 - \eta(\boldsymbol{x}_i; b, \boldsymbol{\beta})]\} \\
&= \sum_{i=1}^{n} \{y_i \log\left(\frac{\exp(f(\boldsymbol{x}_i))}{1 + \exp(f(\boldsymbol{x}_i))}\right) + (1 - y_i) \log\left[\frac{1}{1 + \exp(f(\boldsymbol{x}_i))}\right]\} \\
&= \sum_{i=1}^{n} \{y_i f(\boldsymbol{x}_i) - \log[1 + \exp(f(\boldsymbol{x}_i))]\} \\
&= \sum_{i=1}^{n} \{y_i (b + \boldsymbol{\beta}' \boldsymbol{x}_i) - \log[1 + \exp(b + \boldsymbol{\beta}' \boldsymbol{x}_i)]\}
\end{aligned}$$

The maximizer of $\ell(b, \boldsymbol{\beta})$, say $(b^*, \boldsymbol{\beta}^*)$ can be plugged into $f(\boldsymbol{x}; b, \boldsymbol{\beta})$

$$f(\boldsymbol{x}) = b^* + \boldsymbol{\beta}^* \boldsymbol{x}$$

1. $f(\boldsymbol{x}) > 0 \Rightarrow \eta(\boldsymbol{x}) > 1/2 \Rightarrow Y$ is more likely to be 1
2. $f(\boldsymbol{x}) < 0 \Rightarrow \eta(\boldsymbol{x}) < 1/2 \Rightarrow Y$ is more likely to be 0

Optimization

For simplicity, view $(b, \boldsymbol{\beta}')'$ as new $\boldsymbol{\beta}$
Search solution $\boldsymbol{\beta}$ to score equation

$$\dot{\ell}(\boldsymbol{\beta}) = \mathbf{0}$$

- Recall univariate Newton-Raphson method: find root of $f(x) = 0$. Iteratively do:

$$x_{n+1} \leftarrow x_n - f(x_n)/f'(x_n)$$

  Motivated by Taylor expansion.
- Here:

$$g(\boldsymbol{\beta}, b) := \boldsymbol{\beta}^{(k+1)} \leftarrow \boldsymbol{\beta}^{(k)} - [\ddot{\ell}(\boldsymbol{\beta}^{(k)})]^{-1}\dot{\ell}(\boldsymbol{\beta}^{(k)}),$$

  where $\ddot{\ell}(\boldsymbol{\beta}) = \frac{\partial^2}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T}\ell(\boldsymbol{\beta})$ is the Hessian matrix

Calculations lead to

$$
\begin{aligned}
\dot{\ell}(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \boldsymbol{x}_i\{y_i - \eta(\boldsymbol{x}_i;\boldsymbol{\beta})\} = \mathbf{X}(\boldsymbol{y} - \boldsymbol{\eta}) && (1)\\
&\text{where } \boldsymbol{\eta} := (\eta(\boldsymbol{x}_1;\boldsymbol{\beta}),\ldots,\eta(\boldsymbol{x}_n;\boldsymbol{\beta}))^T && (2)\\
\ddot{\ell}(\boldsymbol{\beta}) &= \frac{\partial}{\partial \boldsymbol{\beta}^T}\mathbf{X}(\boldsymbol{y} - \boldsymbol{\eta}) && (3)\\
&= -\mathbf{X}\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\beta}^T} && (4)\\
&= -\mathbf{X}\mathbf{W}\mathbf{X}^T && (5)
\end{aligned}
$$

Note that $\frac{\partial \eta(\boldsymbol{x}_i;\boldsymbol{\beta})}{\partial \boldsymbol{\beta}^T} = \eta(\boldsymbol{x}_i;\boldsymbol{\beta})[1 - \eta(\boldsymbol{x}_i;\boldsymbol{\beta})]\boldsymbol{x}_i^T$.

Hence $\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\beta}^T} = \mathbf{W}\mathbf{X}^T$, where $\mathbf{W} = \mathrm{Diag}\{\eta(\boldsymbol{x}_i;\boldsymbol{\beta})[1 - \eta(\boldsymbol{x}_i;\boldsymbol{\beta})]\}$

Write the N-R method as

$$\begin{aligned}
\boldsymbol{\beta}^{(k+1)} &\leftarrow \boldsymbol{\beta}^{(k)} - [\ddot{\ell}(\boldsymbol{\beta}^{(k)})]^{-1}\dot{\ell}(\boldsymbol{\beta}^{(k)}) && (6)\\
&= \boldsymbol{\beta}^{(k)} + [\mathbf{X}\mathbf{W}\mathbf{X}^T]^{-1}\mathbf{X}(\boldsymbol{y} - \boldsymbol{\eta}) && (7)\\
&= [\mathbf{X}\mathbf{W}\mathbf{X}^T]^{-1}\mathbf{X}\mathbf{W}[\mathbf{X}^T\boldsymbol{\beta}^{(k)} + \mathbf{W}^{-1}(\boldsymbol{y} - \boldsymbol{\eta})] && (8)\\
&= [\mathbf{X}\mathbf{W}\mathbf{X}^T]^{-1}\mathbf{X}\mathbf{W}\boldsymbol{z} && (9)
\end{aligned}$$

- This is exactly the solution to weighted least square with design matrix $\mathbf{X}$, response variable $\boldsymbol{z}$ and weights $\eta(\boldsymbol{x}_i; \boldsymbol{\beta})[1 - \eta(\boldsymbol{x}_i; \boldsymbol{\beta})]$.
- One must update the response variable and the weight matrix for each iteration.
- Convergence is NOT guaranteed. $\mathbf{W}$ and $\mathbf{X}\mathbf{W}\mathbf{X}^T$ must be invertible.
- Data separation issue: if two classes are well separated, all $\eta(\boldsymbol{x}_i)$ are too close to 0 or 1 $\Rightarrow$ $\mathbf{W}$ is almost $\mathbf{0}$ (trouble!)

LDA vs Logistic Regression

- Logistic Regression is less sensitive to nongaussian data.
- Logistic Regression beats LDA when nongaussian or the covariance is not equal
- LDA is subject to outliers.
- Logistic less efficient than LDA. The latter exploits the full likelihood while logistic regression uses conditional likelihood.
- Logistic regression needs large sample size to work well. LDA can be quite flexible.
- Both have big problem when $p \gg n$.

Alternative coding for logistic regression

Recall for $y_i = 0,\ 1$

$$y_i \log\left(\frac{\exp(f(\boldsymbol{x}_i))}{1+\exp(f(\boldsymbol{x}_i))}\right) + (1-y_i)\log\left[\frac{1}{1+\exp(f(\boldsymbol{x}_i))}\right]$$

This is equivalent to the following function for coding $y_i = \pm 1$

$$\log\left(\frac{1}{\exp(-y_i f(\boldsymbol{x}_i))+1}\right) = -\log\left(\exp(-y_i f(\boldsymbol{x}_i))+1\right)$$

Logistic regression can be viewed as minimizing over $(\boldsymbol{\beta}, b)$

$$\sum_{i=1}^{n} \log\left(\exp(-y_i f(\boldsymbol{x}_i))+1\right) = \sum_{i=1}^{n} L(y_i(\boldsymbol{\beta}^T \boldsymbol{x}_i + b))$$

Gradient descent optimization

The gradient descent algorithm takes the following update iteratively to minimize $f(\boldsymbol{\omega})$:

$$\boldsymbol{\omega}_{(k+1)} \leftarrow \boldsymbol{\omega}_{(k)} - \gamma f'(\boldsymbol{\omega}_{(k)})$$

where $0 < \gamma \leq 1$ is step size.

- Compared to the Newton-Raphson method

  $$\boldsymbol{\omega}_{(k+1)} \leftarrow \boldsymbol{\omega}_{(k)} - (f''(\boldsymbol{\omega}))^{-1} f'(\boldsymbol{\omega}_{(k)}),$$

  the gradient descent method directly update the minimizing point toward the direction of smaller (smallest) value of f, while Newton-Raphson method essentially indirectly optimizes by finding the root of $f'(\boldsymbol{\omega}) = 0$
- The direction $\gamma f'(\boldsymbol{\omega}_{(k)})$ is different from $(f''(\boldsymbol{\omega}))^{-1} f'(\boldsymbol{\omega}_{(k)})$.
- N-R should converge sooner than gradient descent. The latter may call for many iterations.

The goal is to minimize over $\boldsymbol{\omega} = (\boldsymbol{\beta}, b)$

$$f(\boldsymbol{\omega}) := \sum_{i=1}^{n} \log[1 + \exp(-y_i \boldsymbol{\omega}^T \mathbf{x}_i)]$$

whose gradient is

$$\begin{aligned} f'(\boldsymbol{\omega}) &:= \sum_{i=1}^{n} \frac{-\exp(-y_i \boldsymbol{\omega}^T \mathbf{x}_i)}{1 + \exp(-y_i \boldsymbol{\omega}^T \mathbf{x}_i)} y_i \mathbf{x}_i \\ &= \sum_{i=1}^{n} \left\{ \frac{1}{1 + \exp(-y_i \boldsymbol{\omega}^T \mathbf{x}_i)} - 1 \right\} y_i \mathbf{x}_i \end{aligned}$$

At each iteration, we calculate the gradient, and then update according to

$$\boldsymbol{\omega}_{(k+1)} \leftarrow \boldsymbol{\omega}_{(k)} - \gamma f'(\boldsymbol{\omega}_{(k)})$$

The next section would be . . . . . .

Assessment of classifiers

A classifier ϕ classifies any input $\boldsymbol{x}$ into a class label $\{1, \ldots, K\}$. How do we assess the performance of classifier ϕ?

- Misclassification occurs if $\boldsymbol{x}$ is classified to label j, but was actually from class $k \neq j$.
- For a mixed population $(\boldsymbol{X}, Y)$, the *total probability of misclassification* for classifier ϕ:

$$P(\phi(\boldsymbol{X}) \neq Y) = \sum_{k=1}^{K} P(\phi(\boldsymbol{X}) \neq k \mid Y = k) P(Y = k),$$

  where $P(\phi(\boldsymbol{X}) \neq k \mid Y = k)$ is the conditional probability of misclassification given $Y = k$. Here the probability P on the left side is with respect to the distribution of $(\boldsymbol{X}, Y)$.
- A classifier ϕ is *optimal* if it has the smallest t.p.m. compared to any other classifiers, i.e.,

$$P(\phi(\boldsymbol{X}) \neq Y) \leq P(\varphi(\boldsymbol{X}) \neq Y), \quad \varphi \in \Phi.$$

Total probability of misclassification

- Bayes rule classifier is optimal (when the distributions are known).
- If a classifier $\hat{\phi}$ is estimated from a finite sample $\mathcal{D} := \{(\boldsymbol{x}_i, y_i)\}$, the total probability of misclassification (t.p.m) is usually adapted to the t.p.m. conditional on $\mathcal{D}$ $P(\hat{\phi}_{\mathcal{D}}(\boldsymbol{X}) \neq Y \mid \mathcal{D})$.
- Clearly, this probability above is a random variable depending on $\mathcal{D}$ [different data $\mathcal{D}$ lead to different realized $\hat{\phi}_{\mathcal{D}}(\cdot)$ which have different (conditional) t.p.m.] When treating $\mathcal{D}$ as given (and fixed), this probability becomes a constant.
- Total probability of misclassification is also called generalization error, test error, etc.

Misclassification rates

- In practice, since the distribution of $(\boldsymbol{X}, Y)$ is unknown, one cannot compute P.
- Instead, we consider a "big" test data set $\mathcal{T} = \{(\boldsymbol{x}_j, y_j), j = 1, \ldots, M\}$, which induces an empirical distribution. This empirical distribution assigns $1/M$ probability mass to each point $(\boldsymbol{x}_j, y_j)$ in $\mathcal{T}$.
- The probability of $\phi(\boldsymbol{X}) \neq Y$ based on this empirical distribution is

$$\hat{P}(\hat{\phi}_{\mathcal{D}}(\boldsymbol{X}) \neq Y \mid \mathcal{D}) = \frac{1}{M} \sum_{j=1}^{M} \mathbb{1}_{\{\phi(\boldsymbol{x}_j) \neq y_j\}}$$

  which is an unbiased and consistent estimate of the true generalization error $P(\hat{\phi}_{\mathcal{D}}(\boldsymbol{X}) \neq Y \mid \mathcal{D})$
- This is sometimes called "misclassification rate".

- We usually do not have the luxury to have a big test data set.
- The same sample is used for estimation of the classification rule ϕ and also for evaluation of the performance of ϕ. In these cases, we could either
  1. divide the sample into training and testing sets; or
  2. use cross-validation.

Training and testing sets

- Binary classification ($K = 2$). Divide inputs $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m$ and $\boldsymbol{y}_1, \dots, \boldsymbol{y}_n$ into two groups

$$\boldsymbol{X}_{tr} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{m_1}, \boldsymbol{y}_1, \dots, \boldsymbol{y}_{n_1}\} \quad \text{(training)}$$

  and

$$\boldsymbol{X}_{test} = \{\boldsymbol{x}_{m_1+1}, \dots, \boldsymbol{x}_m, \boldsymbol{y}_{n_1+1}, \dots, \boldsymbol{y}_n\} \quad \text{(testing).}$$

  (Sorry for the abuse of notation. $\boldsymbol{y}$ is not the class label here.)
- Estimate the classifier ϕ using $\boldsymbol{X}_{tr}$.
- Estimate the misclassification rate using $\boldsymbol{X}_{test}$.
- Use Confusion matrix, defined in the next slide.

- Training confusion matrix

		True class	
		Class 1	Class 2
classified to	Class 1	r_{11}	r_{12}
	Class 2	r_{21}	r_{22}

where r_{ij} is the number of observations in the **training** sample which are classified to class i and are actually from class j.

- $\sum r_{ij} = n_1 + m_1 := N_{tr}$, thus

$$\text{Training misclassification rate} = (r_{12} + r_{21})/N_{tr}.$$

- Testing confusion matrix

		True class	
		Class 1	Class 2
classified to	Class 1	s_{11}	s_{12}
	Class 2	s_{21}	s_{22}

where r_{ij} is the number of observations in the **testing** sample which are classified to class i and are actually from class j.

- $\sum s_{ij} = n - n_1 + m - m_1 := N_{test}$, thus

$$\text{Testing misclassification rate} = (s_{12} + s_{21})/N_{test}.$$

Training and testing sets IRIS data example

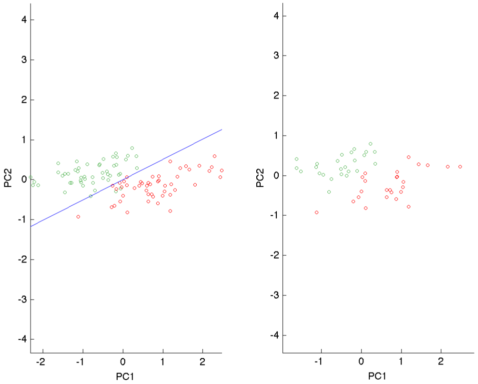


Figure: Left: Two classes (versicolor and virginica), $n = 50, m = 50$ total observations, together with the separating hyperplane by LDA. Right: $n_1 = 25, m_2 = 25$ training observations.

Training and testing sets
IRIS data example

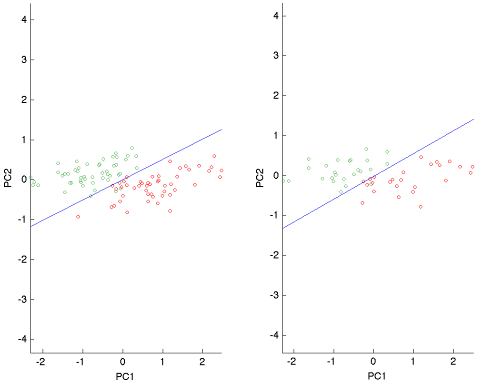


Figure: Right: $n_1 = 25, m_2 = 25$ training observations, together with the separating hyperplane by LDA, estimated using only training observations.

Training and testing sets IRIS data example

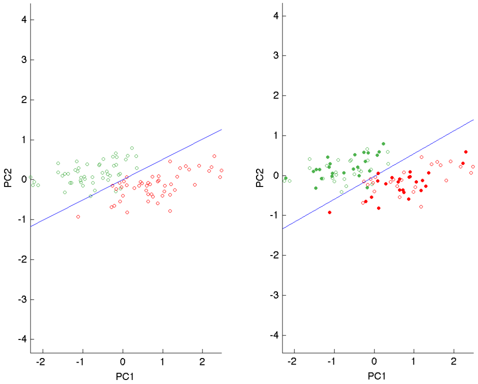


Figure: Right: The remaining points corresponding to the testing set are overlaid.

Training and testing sets
IRIS data example

- Training confusion matrix for this particular choice of training set.

		True class	
		Versicolor	Virginica
classified to	Versicolor	24	1
	Virginica	1	24

Training misclassification rate $= (r_{12} + r_{21})/N_{tr} = 2/50 = 4\%$.

Training and testing sets
IRIS data example

- Testing confusion matrix for this particular choice of training set.

		True class	
		Versicolor	Virginica
classified to	Versicolor	23	0
	Virginica	2	25

Testing misclassification rate $= (s_{12} + s_{21})/N_{test} = 2/50 = 4\%$.

- The testing misclassification rate is an estimate of the total probability of misclassification $P(\hat{\phi}(\boldsymbol{X}) \neq Y)$. May not be a good one since the sample size is too small.
- In most cases the tr. and testing sets are chosen in advance. In this example, testing and training sets are chosen at random. Different choice of sets will lead to different rates.

K-fold Cross Validation

- To minimize the randomness in the choice of training and testing sets, we repeat the process. One example is the K-fold cross validation.

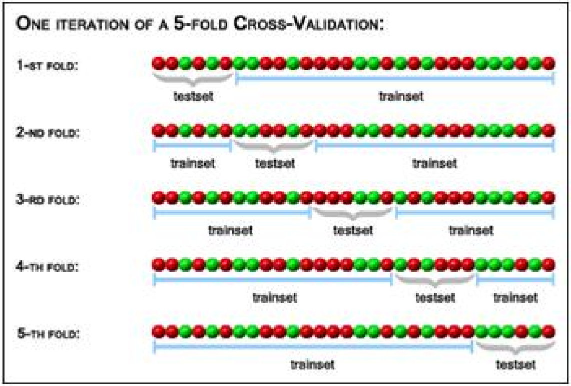


Figure: For ith fold, testing misclassification rate p_i is computed, average of those p_i is the estimate of the total probability of misclassification $P(\hat{\phi}(\boldsymbol{X}) \neq Y)$.

Leave-One-Out Cross Validation

- The N-fold cross validation, where N is the sample size, is called Leave-One-Out Cross Validation.

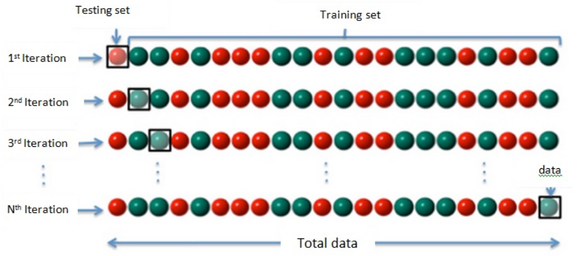


Figure: For ith iteration, testing misclassification rate p_i is computed, average of those p_i is the estimate of the total probability of misclassification $P(\hat{\phi}(\boldsymbol{X}) \neq Y)$.

- Note that for each fold, say V_1, and its complement V_{-1}, we are not estimating the performance of $\hat{\phi}_{\mathcal{D}}$ on V_1, but are estimating the performance of $\hat{\phi}_{V_{-1}}$ on V_1. Hence the estimated generalization error is biased.
    - Larger K means each fold is smaller, and since only one fold is left out, the training set V_{-1} is closer to the full set $\mathcal{D}$. Hence larger K means less biased estimate.
- On the other hand, as $K \to N$, V_{-i}'s become too similar to each other. An extreme case is $K = N$. This makes it difficult for V_{-i}'s to mimic the typical observations of a $(n-1)$ data set from the true distribution (especially when N is itself small). Moreover, this makes the resulting CV estimator too much depend on the data $\mathcal{D}$.
    - Larger K means the estimated generalization errors (before taking average) are highly correlated and taking average does not help to reduce the variance. Smaller K means that the errors are less dependent, and taking average does help to reduce the variance

How many folds are needed?

- Often $K = 10$
- For smaller data, perhaps 5 or 3.
- K-fold CV ($K \ll N$) is different from N-fold CV in that:
  - For a given data set, the N-fold CV is deterministic, since each observation is classified by a deterministic classifier.
  - For a given data set, the K-fold CV depends on how the data are split into folds. An observation may be judged by a different classifier (trained from a different set of training data) due to a different way of splitting.
- Find a balance between bias and variance.
- If time is not an issue, try repeat the K-fold many times with random splitting.

Cross Validated Misclassification Rates

10-fold cross validation is used to estimate the probability of misclassification, for each data:

		classification methods	
		LDA	QDA
	IRIS	0.04	0.05
	Ex. 1 (Equal Cov.)	0.02	0.02
Examples	Ex. 2 (Unequal Cov.)	0.445	0.145
	Ex. 3 (Unequal Cov.)	0.045	0.035
	Ex. 4 (Donut)	0.575	0.06

Expected generalization error

- $P(\phi(\boldsymbol{X}) \neq Y)$ is the generalization error for a given classifier $\phi(\cdot)$. This describes the performance of $\phi(\cdot)$.
- Often $\phi(\cdot)$ is trained from sample $\mathcal{D}$, hence we have $GE_{\mathcal{D}} := P(\hat{\phi}_{\mathcal{D}}(\boldsymbol{X}) \neq Y \mid \mathcal{D})$. This quantity measures the performance of $\hat{\phi}_{\mathcal{D}}(\boldsymbol{X})$, which is indirectly a measure of the performance of a classification procedure / method (through its behavior on $\mathcal{D}$.)
- A related quantity is

$$GE := \mathrm{E}[GE_{\mathcal{D}}] = \mathrm{E}[P(\hat{\phi}_{\mathcal{D}}(\boldsymbol{X}) \neq Y \mid \mathcal{D})]$$

  where the expectation is with respect to the distribution of $\mathcal{D}$ and the probability P is with respect to the distribution of $(\boldsymbol{X}, Y)$. This measures the average performance of the procedure / method. Hence this quantity is not specific to any sample.

The next section would be . . . . . .

LDA and QDA in R

```
library(MASS)
data(iris)
train <- sample(1:150, 75)
table(iris$Species[train])

z <- lda(Species ~ ., iris, prior = c(1,1,1)/3, subset = train)
# predictions
predict(z, iris[-train, ])$class
# true labels
iris$Species[-train]

z <- qda(Species ~ ., iris, prior = c(1,1,1)/3, subset = train)
# predictions
predict(z, iris[-train, ])$class
# true labels
iris$Species[-train]
```

Logistic Regression in R

Use `glm` function.

```
> z <- glm(Species ~ ., iris, family = 'binomial')
Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: fitted probabilities numerically 0 or 1 occurre
>
```

Data separation issue in this data set.

The next section would be . . . . . .

Various classifiers

- There are thousands of classification methods available.
- Examples of simpler methods
  1. Nearest Centroid classifier,
  2. Naive Bayes classifier,
  3. k-Nearest Neighbor classifier,
- Examples of advanced methods
  1. Support Vector Machines (SVM: Ch 11, Izenman),
  2. Distance Weighted Discrimination (DWD: Marron and his colleagues),
  3. Classification And Regression Trees (CART: Ch 9, Izenman).
- A nonlinear classifier can be obtained by using the *kernel* trick. (Section 11.3, Izenman)

Some advanced classifiers and nonlinear classifiers will be introduced later in this course.

Nearest Centroid/ Naive Bayes classifier

For two group classification ($\boldsymbol{x}_{11}, \ldots, \boldsymbol{x}_{1n}$ and $\boldsymbol{x}_{21}, \ldots, \boldsymbol{x}_{2n}$),

- the nearest centroid classifier is

$$\phi(\boldsymbol{x}) = 1 \text{ if } \boldsymbol{b}'(\boldsymbol{x} - \frac{\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2}{2}) < 0, \quad \boldsymbol{b} = (\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1),$$

  sometimes called mean difference classifier;

- the Naive Bayes classifier is

$$\phi(\boldsymbol{x}) = 1 \text{ if } \boldsymbol{b}'(\boldsymbol{x} - \frac{\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2}{2}) < 0, \quad \boldsymbol{b} = \boldsymbol{D}_P^{-1}(\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1),$$

  where $\boldsymbol{D}_P$ is the diagonal matrix consisting of diagonal elements of $\boldsymbol{S}_P$.
  This is a Bayes rule classifier (applied to Gaussian data; hence is LDA) assuming the (common) covariance matrix Σ is diagonal (a reason that it is called naive).

Nearest Centroid (Mean Difference) IRIS data example

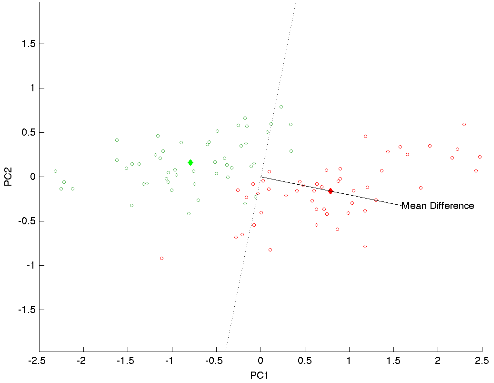


Figure: Mean difference direction $\boldsymbol{b} \propto \bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2$ and its separating hyperplane.

Naive Bayes classifier IRIS data example

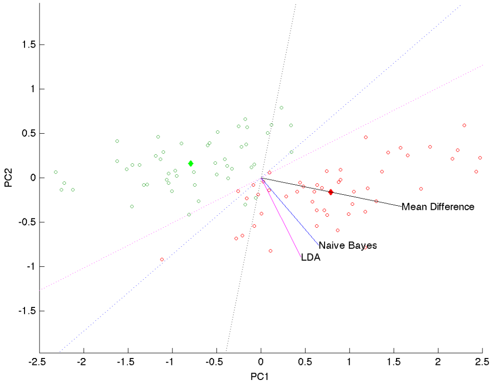


Figure: Better classification by Naive Bayes (Naive LDA) and by LDA.

Naive Bayes classifier
IRIS data example

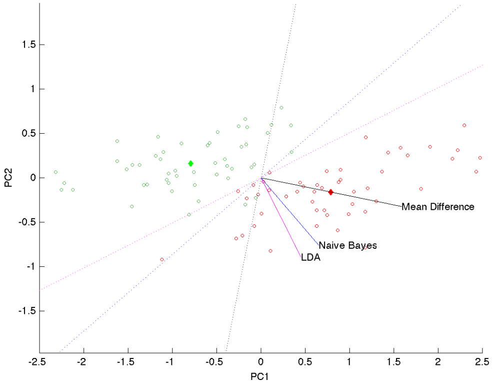


Figure: Better classification by Naive Bayes (Naive LDA) and by LDA.

k-nearest-neighbor (k-NN)

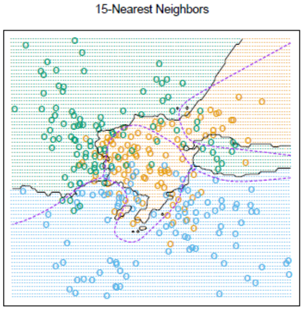
The k-nearest-neighbor classifiers are memory-based, and require no model to fit. Given a point $\boldsymbol{x}^*$, we find k points $\boldsymbol{x}_{(r)}, r = 1, \ldots, k$, among training inputs, closest in *distance* to $\boldsymbol{x}^*$. $\boldsymbol{x}^*$ is classified using majority vote among the k neighbors.

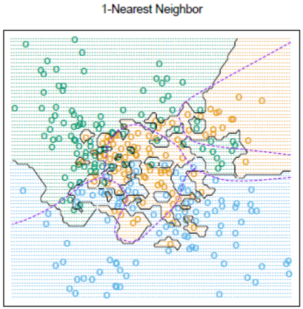
- simple to use.
- shown to be successful in examples.
- requires large memory if the dataset is huge.
- k chosen by comparing test error or cross validated error.
- not so useful for large p (as the concept of distance / neighbor becomes meaningless)

Example from ESL (Hastie, Ribshirani, Friedman)

k-Nearest Neighbor classifiers applied to a simulated data set, with three groups.

The decision boundary of a 15-Nearest Neighbor classifier (top) is fairly smooth compared to a 1-Nearest Neighbor classifier (bottom).





Plug-in classifier

- Given a good estimator for $\eta(\boldsymbol{x}) := P(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x})$, use a classifier defined as

$$\phi(\boldsymbol{x}) := \mathbb{1}_{[\widehat{\eta}(\boldsymbol{x})>1/2]}$$

- The idea of kNN is essential to estimate $P(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x})$ by $\frac{1}{k}\sum_{j=1}^{k} y_{(j)}$ where subscript (j) denotes the jth closest observation in the training data set to $\boldsymbol{x}$.
- If the sample is dense enough, and if the sample size is large, then we expect that $\boldsymbol{x}_{(j)}$ $(j = 1, \ldots, k)$ are all at $\boldsymbol{x}$. Remember that $Y \mid \boldsymbol{X} = \boldsymbol{x}$ is simply a (conditional) Bernoulli random variable and its (conditional) expectation can be estimated by the sample mean (conditioning on $\boldsymbol{X} = \boldsymbol{x}$), $\frac{1}{k}\sum_{j=1}^{k} Y_{(j)}$
- $k = 1$, small bias, but large variance.
- Larger k, larger bias (because some of these neighbors may be far away from $\boldsymbol{x}$), but smaller variance (because of the average).

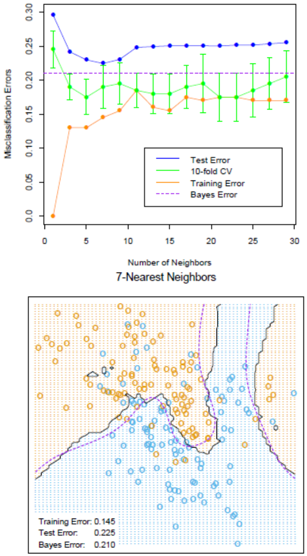


FIGURE 13.4. *k-nearest-neighbors on the two-class mixture data. The upper panel shows the misclassification errors as a function of neighborhood size. Standard error bars are included for 10-fold cross validation. The lower panel shows the decision boundary for 7-nearest-neighbors, which appears to be optimal for minimizing test error. The broken purple curve in the background is the Bayes decision boundary.*

k-NN in R

```
library(class)
data(iris3)

train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
class<-knn(train, test, cl, k = 3, prob=TRUE)

mcr <- 1 - sum(class == cl) / length(cl)
```