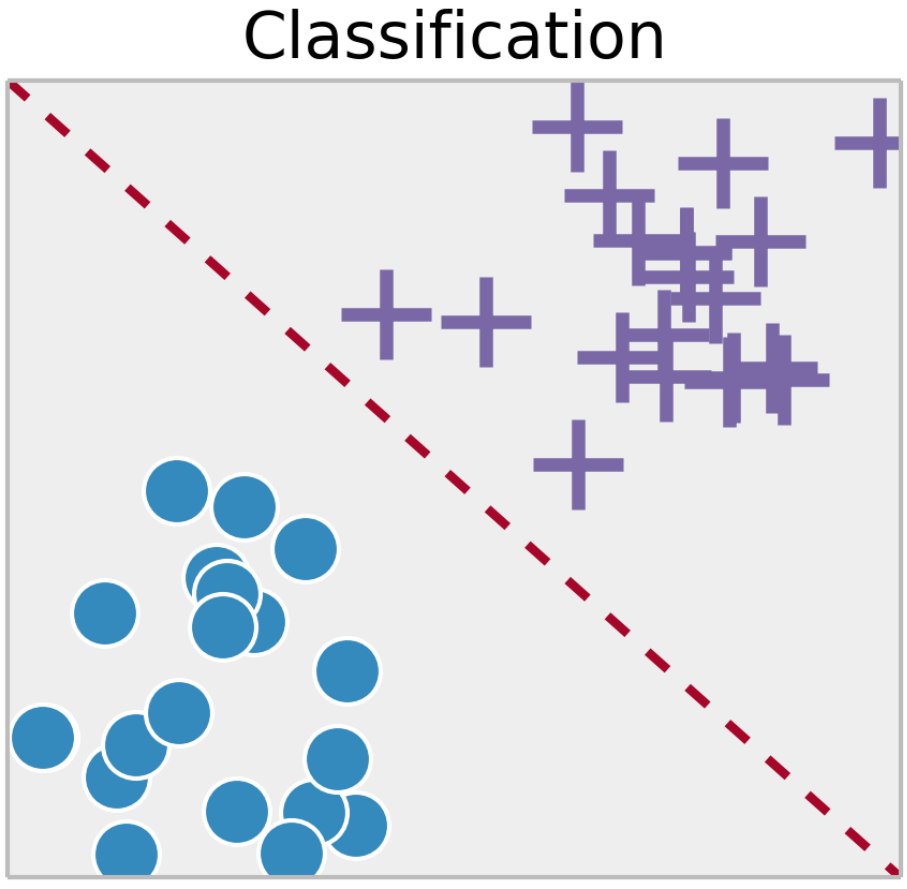
Statistical Machine Learning
Hilary Term 2018

Pier Francesco Palamara
Department of Statistics
University of Oxford

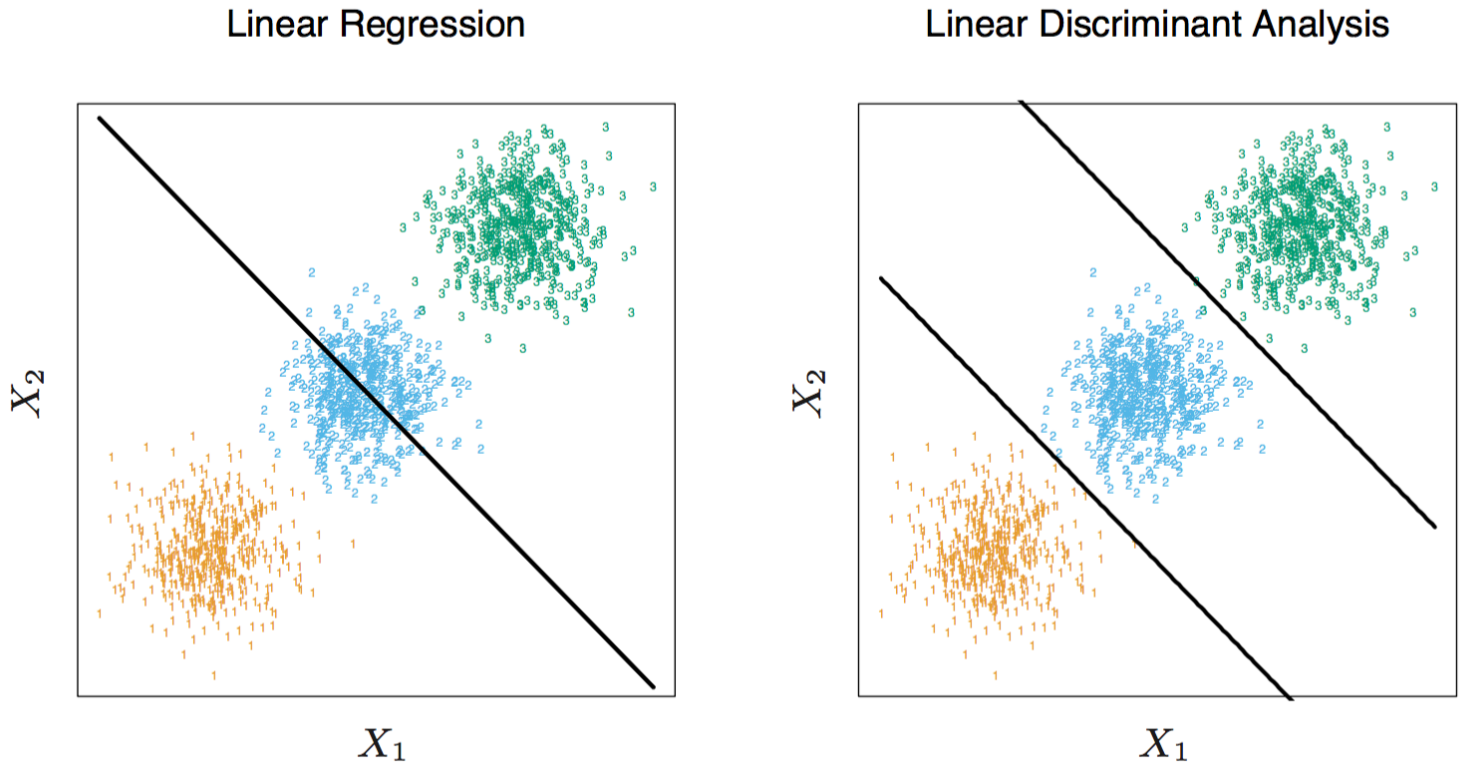
Slide credits and other course material can be found at:
http://www.stats.ox.ac.uk/~palamara/SML18.html

February 9, 2018

Classification



Linear models for classification



Sometimes using regression in a classification setting leads to acceptable results. But the model is usually misspecified, and leads to bad performance (left) compared to methods that are explicitly designed for categorical labels (right).

Figure 4.2 from Hastie, Tibshirani and Friedman, Section 4.2

Recall: Loss function

- Suppose we made a prediction $\hat{Y} = f(X) \in \mathcal{Y}$ based on observation of X.
- How good is the prediction? We can use a **loss function** $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$ to formalize the quality of the prediction.
- Typical loss functions:
  - **Squared loss** for regression
  $$L(Y, f(X)) = (f(X) - Y)^2 .$$
  - **Absolute loss** for regression
  $$L(Y, f(X)) = |f(X) - Y| .$$
  - **Misclassification loss** (or **0-1 loss**) for classification
  $$L(Y, f(X)) = \begin{cases} 0 & f(X) = Y \\ 1 & f(X) \neq Y \end{cases} .$$

  Many other choices are possible, e.g., **weighted misclassification loss**.
- In classification, if estimated probabilities $\hat{p}(k)$ for each class $k \in \mathcal{Y}$ are returned, **log-likelihood loss** (or **log loss**) $L(Y, \hat{p}) = -\log \hat{p}(Y)$ is often used.

The Bayes Classifier

- What is the optimal classifier if the joint distribution (X, Y) were known?
- The density g of X can be written as a mixture of K components (corresponding to each of the classes):

$$g(x) = \sum_{k=1}^{K} \pi_k g_k(x),$$

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

  - $\mathbb{P}(Y = k) = \pi_k$ are the class probabilities,
  - $g_k(x)$ is the conditional density of X, given $Y = k$.

- The **Bayes classifier** $f_{\text{Bayes}} : x \mapsto \{1, \ldots, K\}$ is the one with minimum risk:

$$\begin{aligned} R(f) =& \mathbb{E}\left[L(Y, f(X))\right] = \mathbb{E}_X\left[\mathbb{E}_{Y|X}[L(Y, f(X))|X]\right] \\ =& \int_{\mathcal{X}} \mathbb{E}\left[L(Y, f(X))|X = x\right] g(x) dx \end{aligned}$$

- The minimum risk attained by the Bayes classifier is called **Bayes risk**.
- Minimizing $\mathbb{E}[L(Y, f(X))|X = x]$ separately for each x suffices.

The Bayes Classifier

- Consider the 0-1 loss.
- The risk simplifies to:

$$\begin{aligned}\mathbb{E}\Big[L(Y, f(X))\Big|X = x\Big] &= \sum_{k=1}^{K} L(k, f(x))\mathbb{P}(Y = k|X = x) \\ &= 1 - \mathbb{P}(Y = f(x)|X = x)\end{aligned}$$

- The risk is minimized by choosing the class with the greatest probability given the observation:

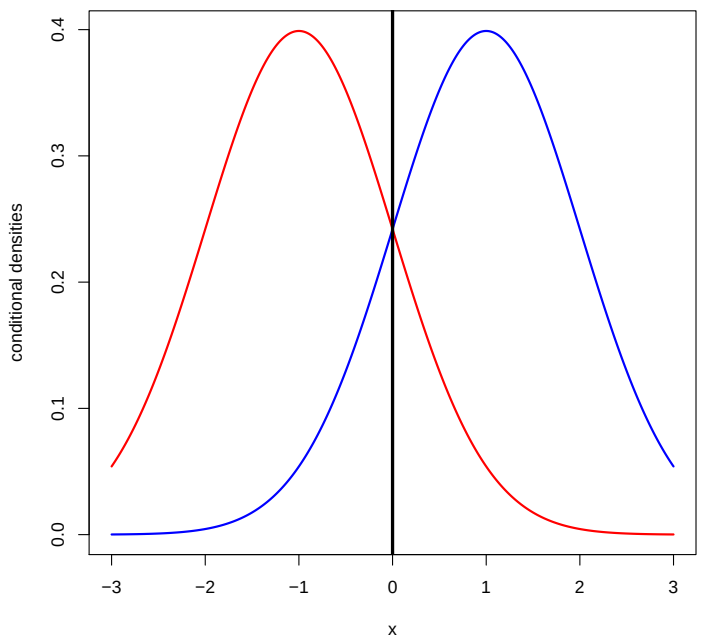
$$\begin{aligned}f_{\text{Bayes}}(x) &= \underset{k=1,\ldots,K}{\arg\max}\, \mathbb{P}(Y = k|X = x) \\ &= \underset{k=1,\ldots,K}{\arg\max}\, \frac{\pi_k g_k(x)}{\sum_{j=1}^{K} \pi_j g_j(x)} = \underset{k=1,\ldots,K}{\arg\max}\, \pi_k g_k(x).\end{aligned}$$

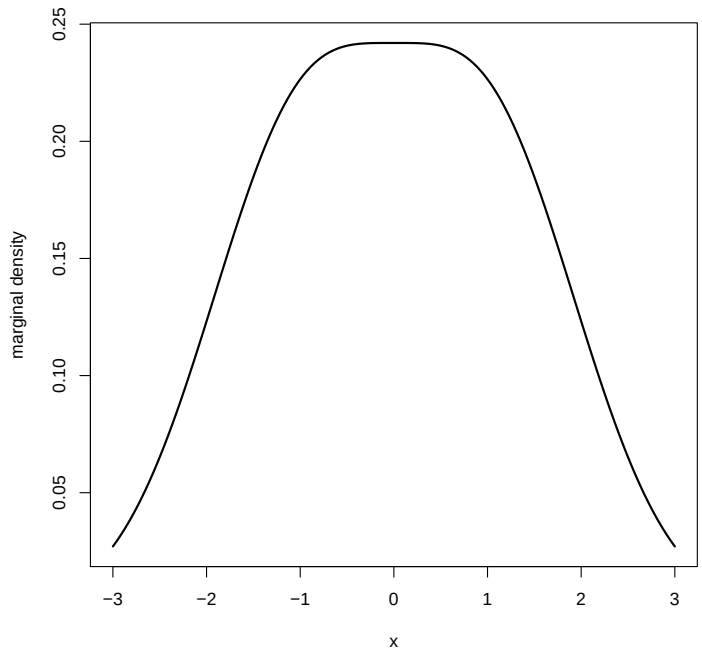
- The functions $x \mapsto \pi_k g_k(x)$ are called **discriminant functions**. The discriminant function with maximum value determines the predicted class of x.

The Bayes Classifier: Example

A simple two Gaussians example: Suppose $X \sim \mathcal{N}(\mu_Y, 1)$, where $\mu_1 = -1$ and $\mu_2 = 1$ and assume equal class probabilities $\pi_1 = \pi_2 = 1/2$.

$$g_1(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x+1)^2}{2}\right) \quad \text{and} \quad g_2(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-1)^2}{2}\right).$$

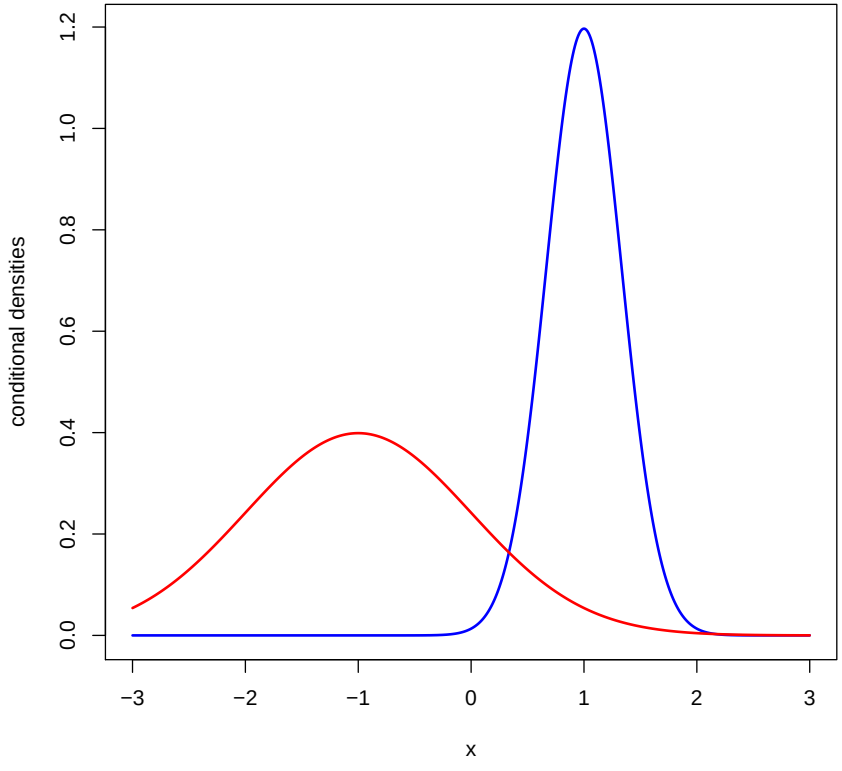


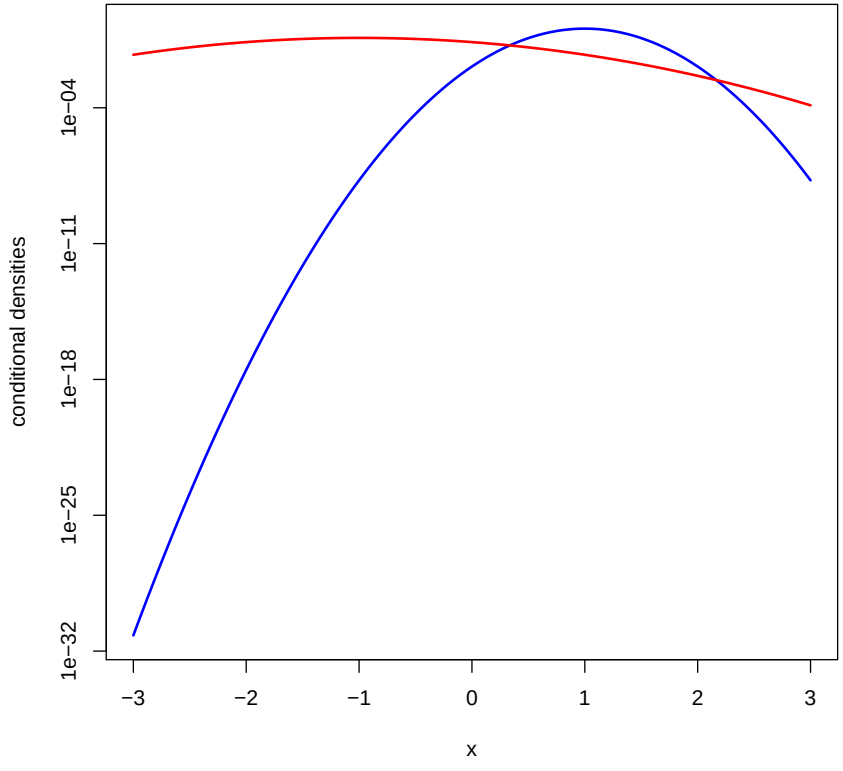


Optimal classification is $f_{\text{Bayes}}(x) = \underset{k=1,\ldots,K}{\arg\max}\ \pi_k g_k(x) = \begin{cases} 1 & \text{if } x < 0, \\ 2 & \text{if } x \geq 0. \end{cases}$

The Bayes Classifier: Example

How do you classify a new observation x if now the standard deviation is still 1 for class 1 but $1/3$ for class 2?





Looking at density in a log-scale, optimal classification is to select class 2 if and only if $x \in [0.34, 2.16]$.

Plug-in Classification

- The Bayes Classifier:

$$f_{\text{Bayes}}(x) \quad = \quad \underset{k=1,\ldots,K}{\arg\max}\, \pi_k g_k(x).$$

- We know neither the conditional densities g_k nor the class probabilities π_k!
- The **plug-in classifier** chooses the class

$$f(x) = \underset{k=1,\ldots,K}{\arg\max}\, \hat{\pi}_k \hat{g}_k(x),$$

- where we plugged in
  - estimates $\hat{\pi}_k$ of π_k and $k = 1, \ldots, K$ and
  - estimates $\hat{g}_k(x)$ of conditional densities,
- **Linear Discriminant Analysis** is an example of plug-in classification.

Linear Discriminant Analysis

- **LDA** is the most well-known and simplest example of plug-in classification.
- Assume multivariate normal conditional density $g_k(x)$ for each class k:

$$X|Y = k \sim \mathcal{N}(\mu_k, \Sigma),$$

$$g_k(x) = (2\pi)^{-p/2}|\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k)\right),$$

  - each class can have a **different mean** μ_k,
  - all classes share the **same covariance** Σ.
- For an observation x, the k-th log-discriminant function is

$$\log \pi_k g_k(x) = c + \log \pi_k - \frac{1}{2}(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k)$$

  The quantity $(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k)$ is the squared **Mahalanobis distance** between x and μ_k.
- If $\Sigma = I_p$ and $\pi_k = \frac{1}{K}$, LDA simply chooses the class k with the nearest (in the Euclidean sense) class mean.

Linear Discriminant Analysis

- Expanding the term $(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k)$,

$$\log \pi_k g_k(x) = c + \log \pi_k - \frac{1}{2}\left(\mu_k^\top \Sigma^{-1}\mu_k - 2\mu_k^\top \Sigma^{-1}x + x^\top \Sigma^{-1}x\right)$$
$$= c' + \log \pi_k - \frac{1}{2}\mu_k^\top \Sigma^{-1}\mu_k + \mu_k^\top \Sigma^{-1}x$$

- Setting $a_k = \log(\pi_k) - \frac{1}{2}\mu_k^\top \Sigma^{-1}\mu_k$ and $b_k = \Sigma^{-1}\mu_k$, we obtain

$$\log \pi_k g_k(x) = c' + a_k + b_k^\top x$$

  i.e. a **linear** discriminant function in x.
- Consider choosing class k over k':

$$a_k + b_k^\top x > a_{k'} + b_{k'}^\top x \qquad \Leftrightarrow \qquad a_\star + b_\star^\top x > 0$$

  where $a_\star = a_k - a_{k'}$ and $b_\star = b_k - b_{k'}$.
- The Bayes classifier thus partitions $\mathcal{X}$ into regions with the same class predictions via **separating hyperplanes**.
- The Bayes classifier under these assumptions is more commonly known as the **LDA classifier**.

Parameter Estimation

- How to estimate the parameters of the LDA model?
- We can achieve this by maximum likelihood (EM algorithm is not needed here since the class variables y_i are observed!).
- Let $n_k = \#\{j : y_j = k\}$ be the number of observations in class k.

$$\ell(\pi, (\mu_k)_{k=1}^K, \Sigma) = \log p\left((x_i, y_i)_{i=1}^n \,|\pi, (\mu_k)_{k=1}^K, \Sigma\right) = \sum_{i=1}^n \log \pi_{y_i} g_{y_i}(x_i)$$

$$= c + \sum_{k=1}^K \sum_{j:y_j=k} \log \pi_k - \frac{1}{2}\left(\log|\Sigma| + (x_j - \mu_k)^\top \Sigma^{-1}(x_j - \mu_k)\right)$$

ML estimates:

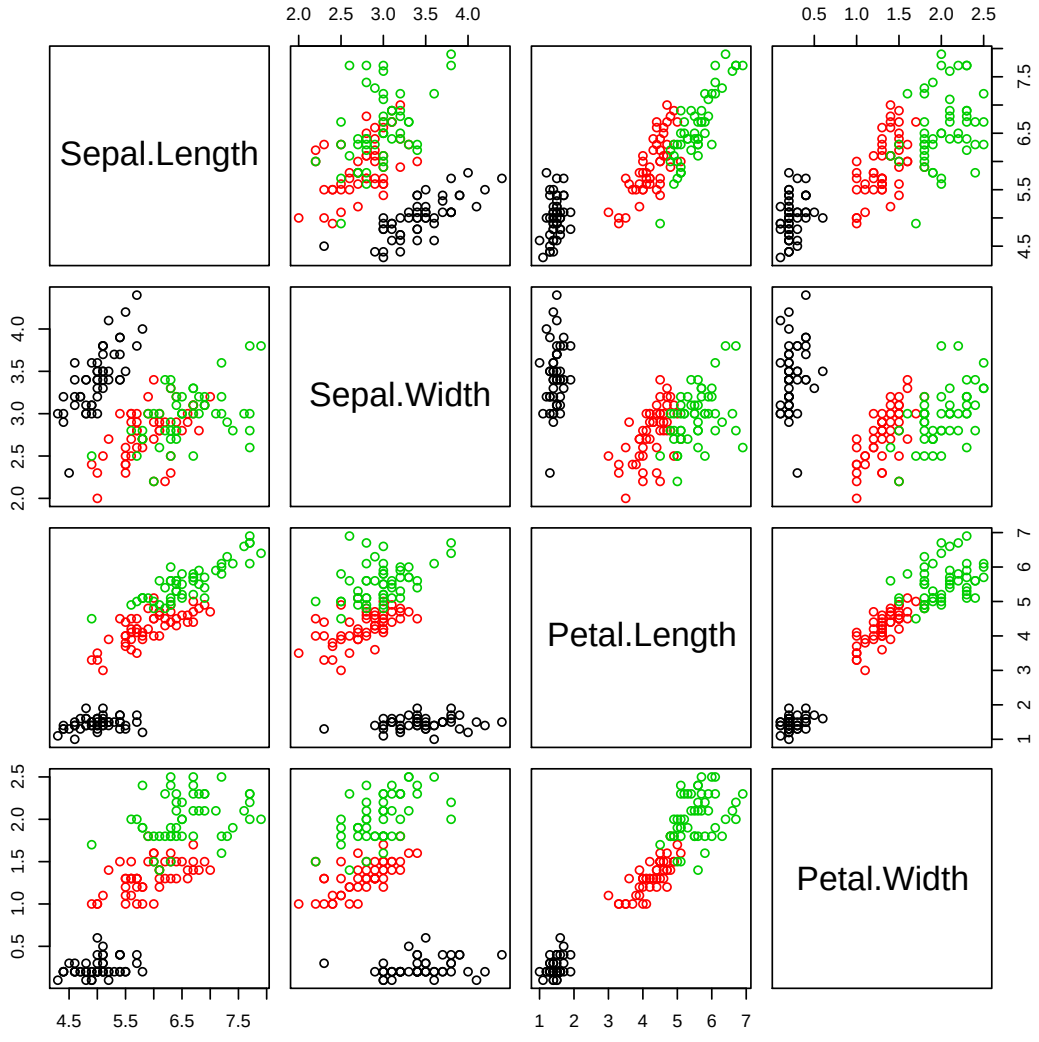
$$\hat{\pi}_k = \frac{n_k}{n} \qquad \hat{\mu}_k = \frac{1}{n_k} \sum_{j:y_j=k} x_j$$

$$\hat{\Sigma} = \frac{1}{n} \sum_{k=1}^K \sum_{j:y_j=k} (x_j - \hat{\mu}_k)(x_j - \hat{\mu}_k)^\top$$

- Note: the ML estimate of Σ is biased. For an unbiased estimate we need to divide by $n - K$.

Iris Dataset

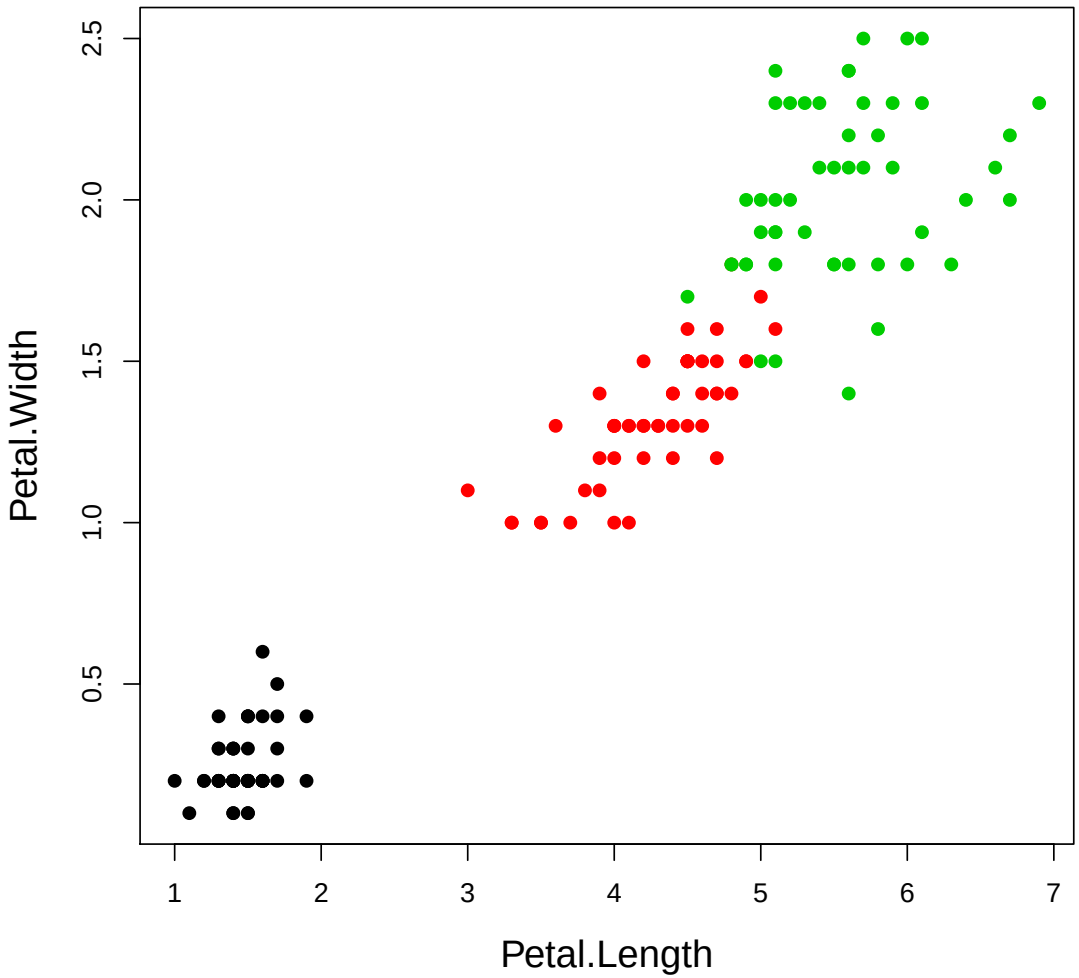
```
library(MASS)
data(iris)
##save class labels
ct <- unclass(iris$Species)
##pairwise plot
pairs(iris[,1:4],col=ct)
```



Iris Dataset

Just focus on two predictor variables.

```
iris.data <- iris[,3:4]
plot(iris.data,col=ct,pch=20,cex=1.5,cex.lab=1.4)
```

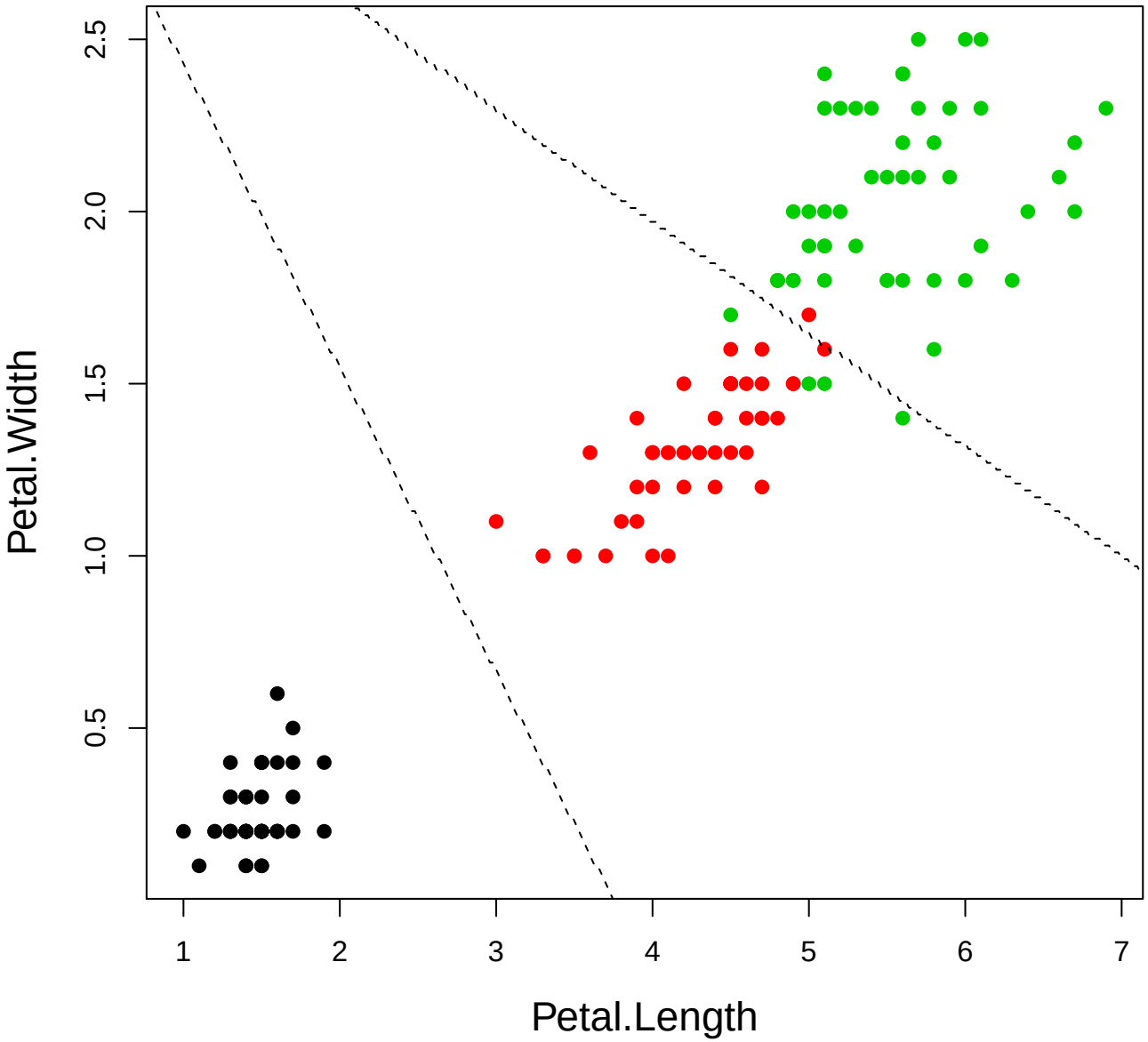


Iris Dataset

Computing and plotting the LDA boundaries.

```
##fit LDA
iris.lda <- lda(x=iris.data,grouping=ct)

##create a grid for our plotting surface
x <- seq(0,8,0.02)
y <- seq(0,3,0.02)
m <- length(x)
n <- length(y)
z <- as.matrix(expand.grid(x,y),0)
colnames(z) = colnames(iris.data)

##classes are 1,2 and 3, so set contours at 1.5 and 2.5
iris.ldp <- predict(iris.lda,z)$class
contour(x,y,matrix(iris.ldp,m,n),
        levels=c(1.5,2.5), add=TRUE, d=FALSE, lty=2)
```

Iris Dataset



Summary: **Linear Discriminant Analysis**

- **LDA**: a plug-in classifier assuming multivariate normal conditional density $g_k(x) = g_k(x|\mu_k, \Sigma)$ for each class k sharing the **same covariance** Σ:

$$X|Y = k \sim \mathcal{N}(\mu_k, \Sigma),$$

$$g_k(x|\mu_k, \Sigma) = (2\pi)^{-p/2}|\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k)\right).$$

- LDA minimizes the squared **Mahalanobis distance** between x and $\hat{\mu}_k$, offset by a term depending on the estimated class proportion $\hat{\pi}_k$:

$$\begin{aligned} f_{\text{LDA}}(x) &= \underset{k\in\{1,\ldots,K\}}{\operatorname{argmax}} \log \hat{\pi}_k g_k(x|\hat{\mu}_k, \hat{\Sigma}) \\ &= \underset{k\in\{1,\ldots,K\}}{\operatorname{argmax}} \underbrace{\left(\log \hat{\pi}_k - \frac{1}{2}\hat{\mu}_k^\top \hat{\Sigma}^{-1}\hat{\mu}_k\right) + \left(\hat{\Sigma}^{-1}\hat{\mu}_k\right)^\top x}_{\text{terms depending on } k \text{ linear in } x} \\ &= \underset{k\in\{1,\ldots,K\}}{\operatorname{argmin}} \frac{1}{2}\underbrace{(x-\hat{\mu}_k)^\top \hat{\Sigma}^{-1}(x-\hat{\mu}_k)}_{\text{squared Mahalanobis distance}} - \log \hat{\pi}_k. \end{aligned}$$

Computations for LDA

- LDA minimizes the squared **Mahalanobis distance** between x and $\hat{\mu}_k$, offset by a term depending on the estimated class proportion $\hat{\pi}_k$:

$$f_{\text{LDA}}(x) \quad = \quad \underset{k \in \{1,\ldots,K\}}{\operatorname{argmin}} \; \frac{1}{2} \underbrace{(x - \hat{\mu}_k)^\top \hat{\Sigma}^{-1} (x - \hat{\mu}_k)}_{\text{squared Mahalanobis distance}} - \log \hat{\pi}_k.$$

- Thus, LDA classification can be implemented as the following two steps:

  (1) **Sphere** the data with respect to **the common covariance estimate** $\hat{\Sigma} = \frac{1}{n} \sum_{k=1}^{K} \sum_{j:y_j=k} (x_j - \hat{\mu}_k)(x_j - \hat{\mu}_k)^\top$:

$$x^\bullet \leftarrow D^{-\frac{1}{2}} U^\top x, \quad \text{where} \quad \hat{\Sigma} = U D U^\top.$$

  (2) Classify to the closest class mean $\hat{\mu}_k^\bullet$ in the transformed space, modulo the effect of the estimated class proportions $\hat{\pi}_k$.

Fisher's Reduced-Rank Linear Discriminant Analysis

- In LDA, data vectors are classified based on Mahalanobis distance to class means.
- There is K class means and they lie on a $(K-1)$-dimensional affine subspace of ambient space $\mathbb{R}^p$: Decision function is unaffected by the directions orthogonal to this subspace.
- Projecting data vectors onto the subspace can be viewed as a dimensionality reduction technique that preserves discriminative information about the labels $\{y_i\}_{i=1}^n$: going from $\mathbb{R}^p$ to $\mathbb{R}^{K-1}$ and potentially $K-1 \ll p$.
- Just like in PCA, we can visualise the structure in the data by choosing an appropriate basis for the subspace and projecting data onto it - immediate visualisation fully describing LDA for $K=3$.
- For $K>3$, Fisher proposed to look for the change of basis that finds **directions that best separate the classes** - the largest possible spread of the centroids after sphering.

LDA projections

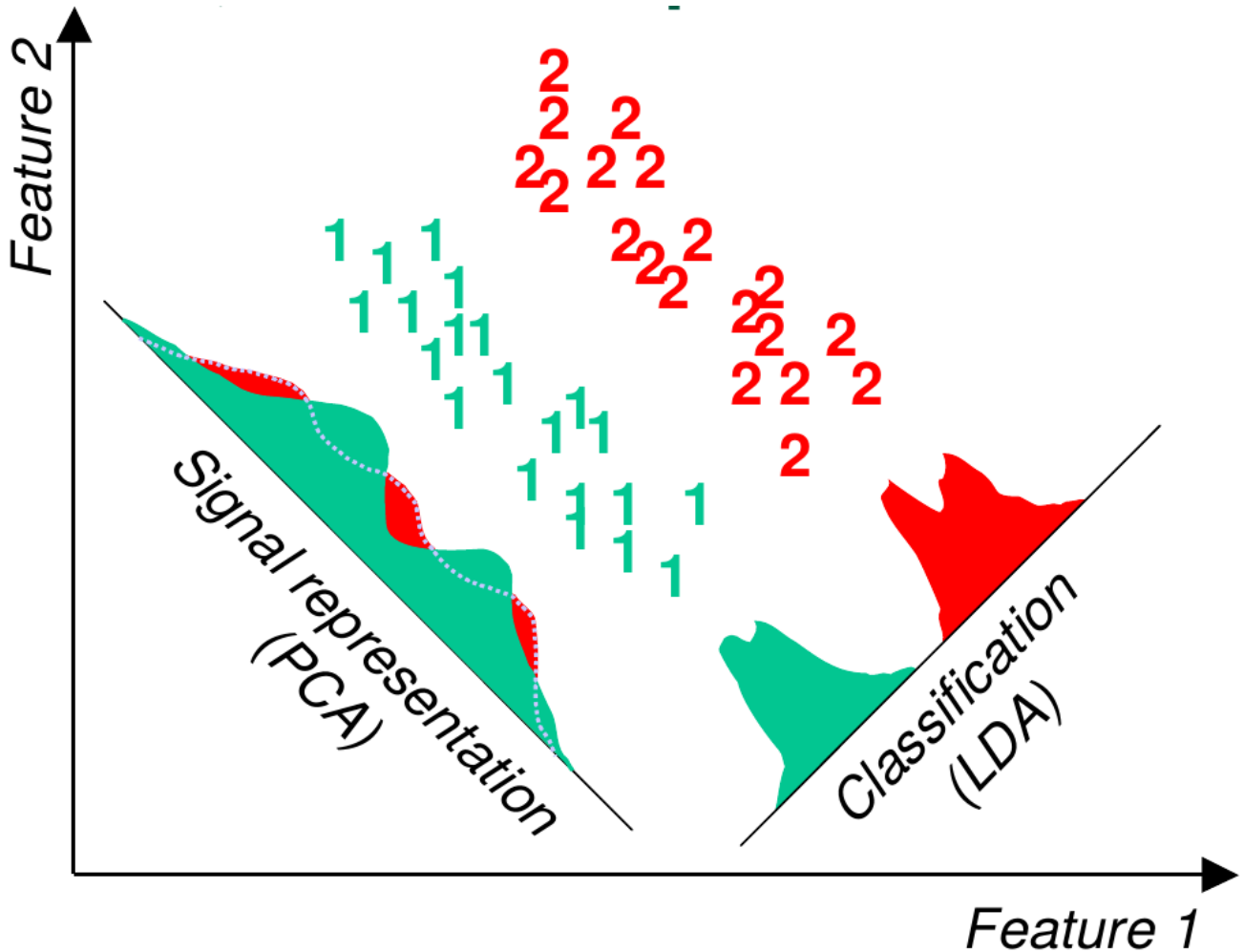


Figure by R. Gutierrez-Osuna

Discriminant Coordinates: 2-classes

- Centroids are $\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$.
- Centroids projected on a vector v are given by $m_k = v^\top \hat{\mu}_k$.
- Variance of projected data on v given by $s_k^2 = \sum_{i:y_i=k} (v^\top x_i - m_k)^2$.
- Goal: find v such that the distance between centroids is maximized, while projected clusters are "tight":

$$\frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

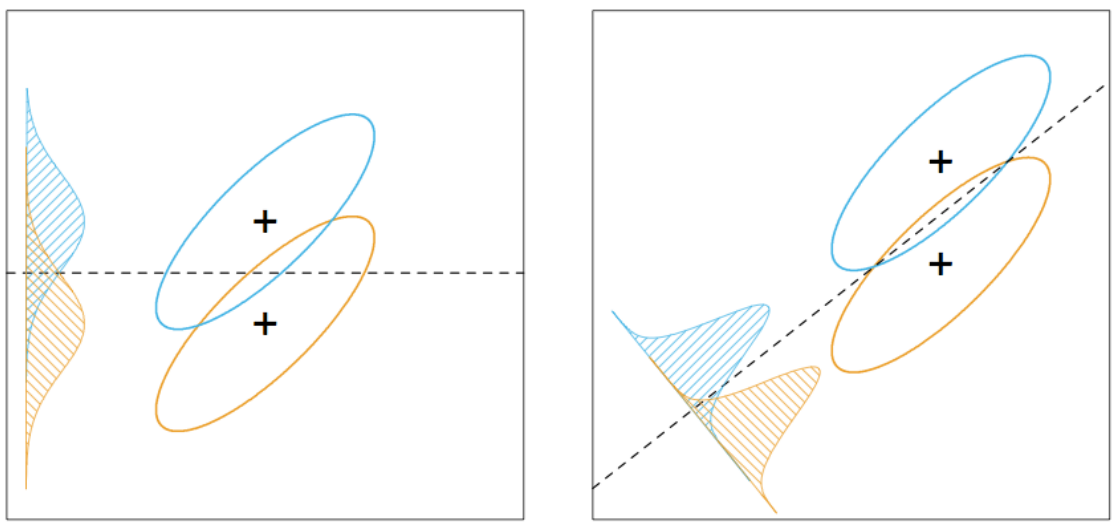


Figure from Hastie, Tibshirani and Friedman, Section 4.3.3

Discriminant Coordinates: 2-classes

- Centroids are $\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$.
- Centroids projected on a vector v are given by $m_k = v^\top \hat{\mu}_k$.
- Variance of projected data on v given by $s_k^2 = \sum_{i:y_i=k} (v^\top x_i - m_k)^2$.
- Goal: find v such that the distance between centroids is maximized, while projected clusters are "tight":

$$\frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} = \frac{v^\top B v}{v^\top \hat{\Sigma} v}$$

where

$$B = (\hat{\mu}_2 - \hat{\mu}_1)(\hat{\mu}_2 - \hat{\mu}_1)^\top \qquad \text{(between-class covariance)}$$

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu}_{y_i})(x_i - \hat{\mu}_{y_i})^\top \qquad \text{(within-class covariance)}$$

(verify above calculations).

Discriminant Coordinates

- More generally:

$$\frac{v^\top B v}{v^\top \hat{\Sigma} v}$$

where

$$B = \frac{1}{n}\sum_{k=1}^{K} n_k(\hat{\mu}_k - \bar{x})(\hat{\mu}_k - \bar{x})^\top \qquad \text{(between-class covariance)}$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu}_{y_i})(x_i - \hat{\mu}_{y_i})^\top \qquad \text{(within-class covariance)}$$

and B has rank at most $K - 1$.

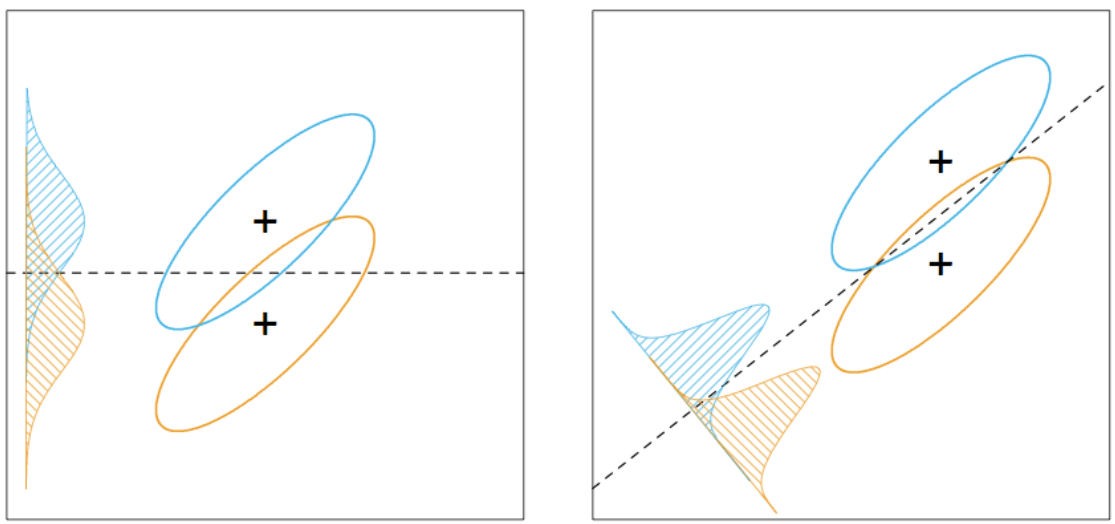


Figure from Hastie, Tibshirani and Friedman, Section 4.3.3

Discriminant Coordinates

- To solve for the optimal v, we first reparameterize it as $u = \hat{\Sigma}^{\frac{1}{2}} v$.

$$\frac{v^\top B v}{v^\top \hat{\Sigma} v} = \frac{u^\top (\hat{\Sigma}^{-\frac{1}{2}})^\top B \hat{\Sigma}^{-\frac{1}{2}} u}{u^\top u} = \frac{u^\top B^\bullet u}{u^\top u}$$

  where $B^\bullet = (\hat{\Sigma}^{-\frac{1}{2}})^\top B \hat{\Sigma}^{-\frac{1}{2}}$.
- We have solved something similar before. The maximization over u is achieved by the first eigenvector u_1 of $B^\bullet$.
- We also look at the remaining eigenvectors u_l associated to the non-zero eigenvalues and define the **discriminant coordinates** as $v_l = \hat{\Sigma}^{-\frac{1}{2}} u_l$.
- The v_l's span exactly the affine subspace spanned by $(\hat{\Sigma}^{-1}\hat{\mu}_k)_{k=1}^K$ (these vectors are given as the "linear discriminants" in the R-function `lda`).

Crabs Dataset

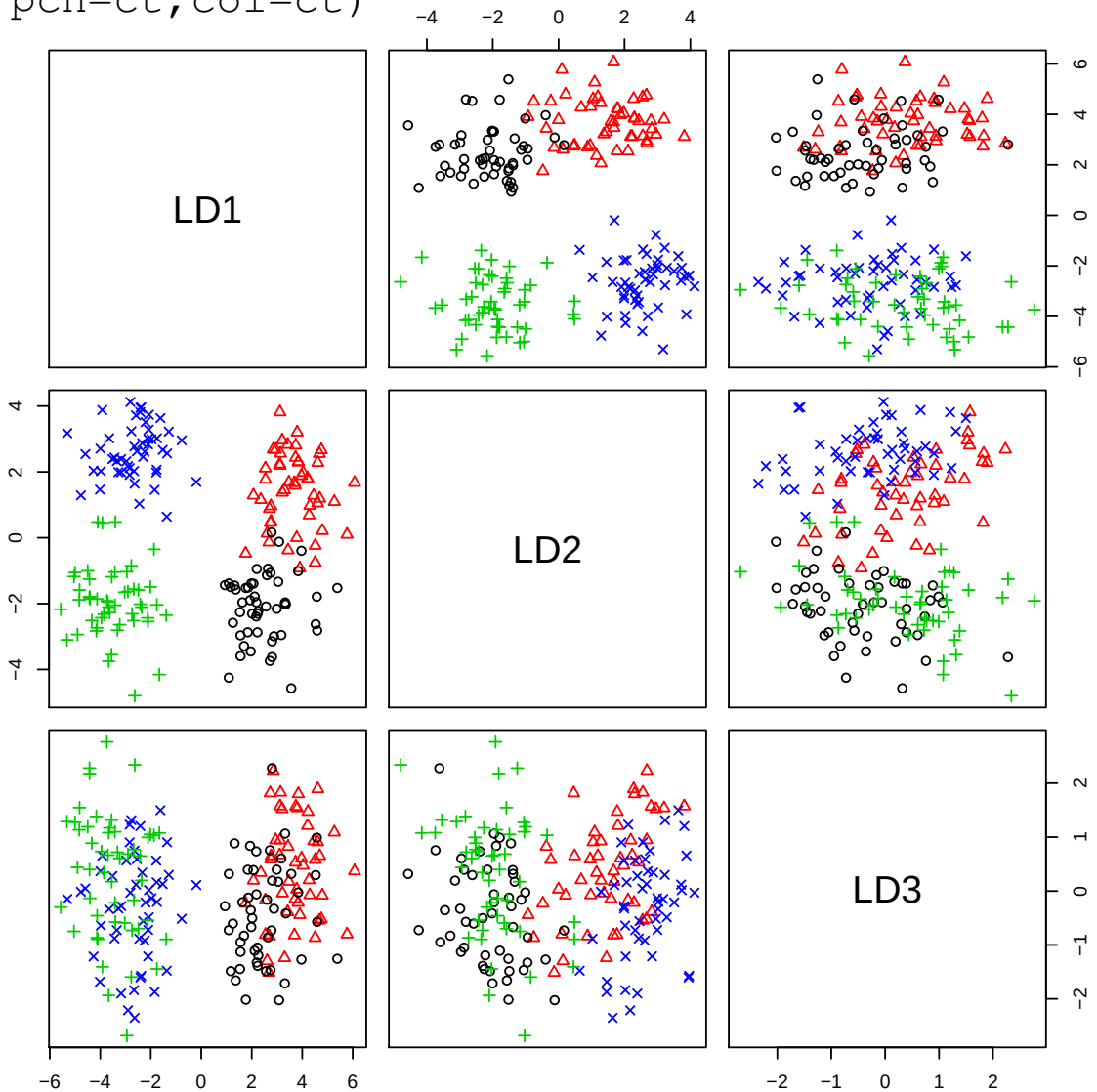
```
library(MASS)
data(crabs)

## create class labels (species+sex)
crabs$spsex=factor(paste(crabs$sp,crabs$sex,sep=""))
ct <- unclass(crabs$spsex)

## LDA on crabs in log-domain
cb.lda <- lda(log(crabs[,4:8]),ct)
```

Crabs Dataset

```
> cb.lda
Call:
lda(log(crabs[, 4:8]), ct)

Prior probabilities of groups:
   1    2    3    4
0.25 0.25 0.25 0.25

Group means:
        FL       RW       CL       CW       BD
1 2.564985 2.475174 3.312685 3.462327 2.441351
2 2.672724 2.443774 3.437968 3.578077 2.560806
3 2.852455 2.683831 3.529370 3.649555 2.733273
4 2.787885 2.489921 3.490431 3.589426 2.701580

Coefficients of linear discriminants:
          LD1        LD2        LD3
FL -31.217207  -2.851488  25.719750
RW  -9.485303 -24.652581  -6.067361
CL  -9.822169  38.578804 -31.679288
CW  65.950295 -21.375951  30.600428
BD -17.998493   6.002432 -14.541487

Proportion of trace:
   LD1    LD2    LD3
0.6891 0.3018 0.0091
```

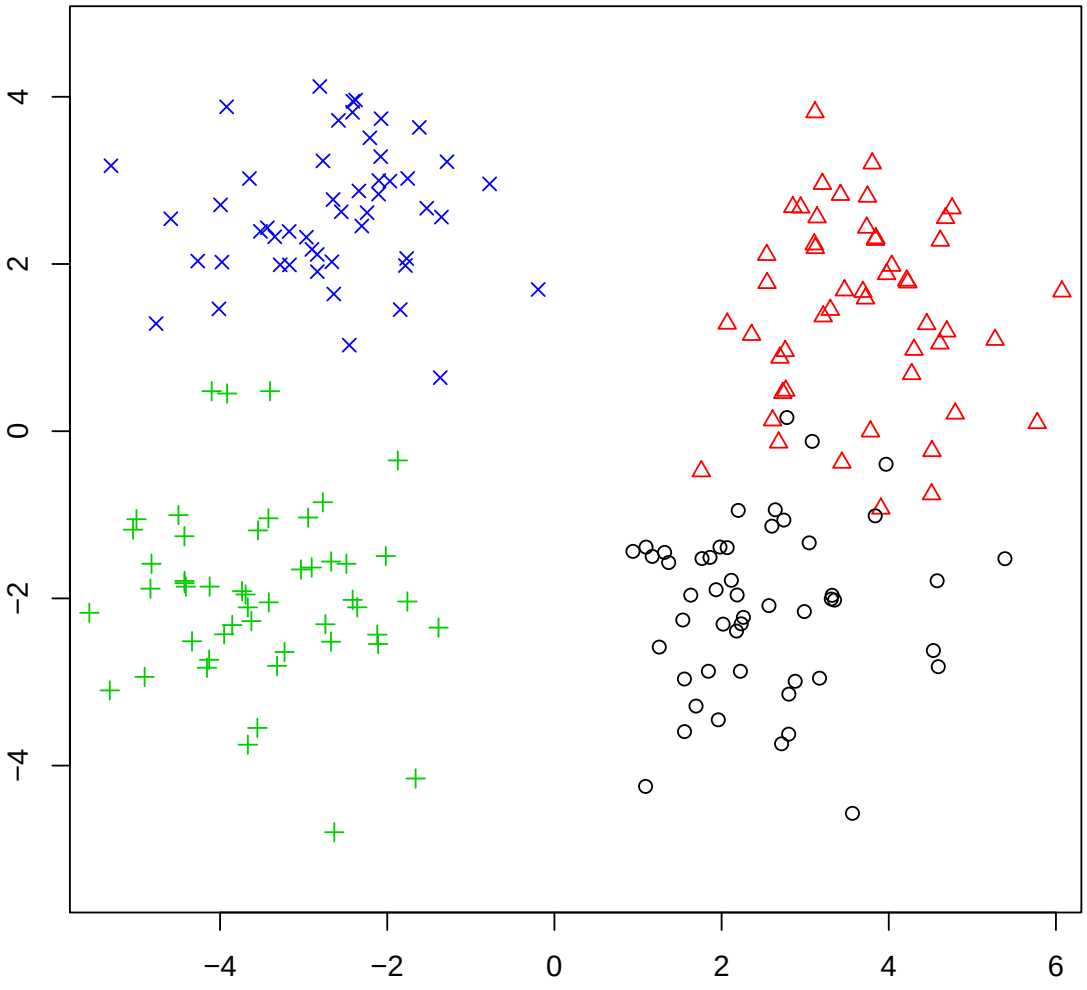
Crabs Dataset

```
cb.ldp <- predict(cb.lda)
pairs(cb.ldp$x,pch=ct,col=ct)
```



Crabs Dataset

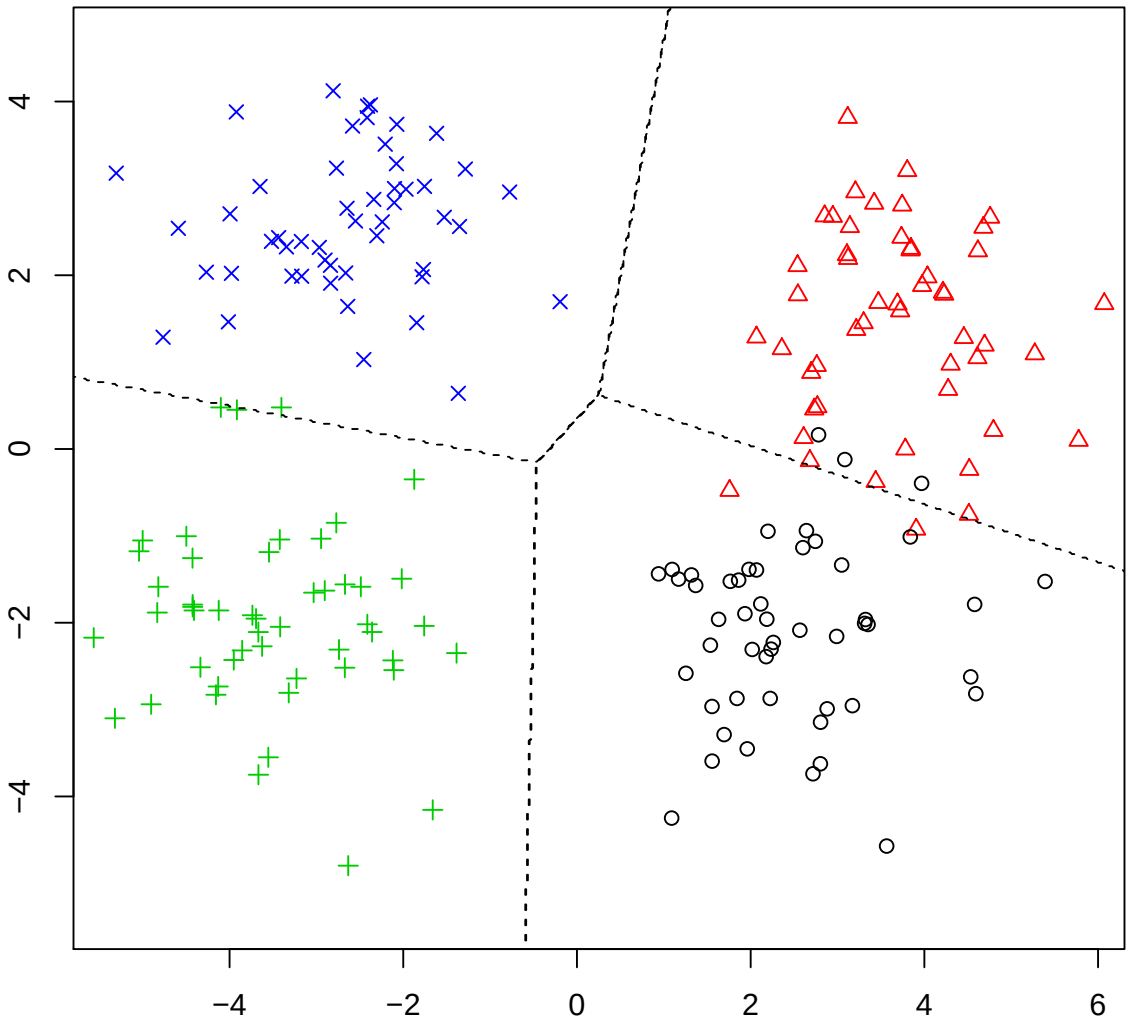
```
cb.ldp12 <- cb.ldp$x[,1:2]
eqscplot(cb.ldp12,pch=ct,col=ct)
```



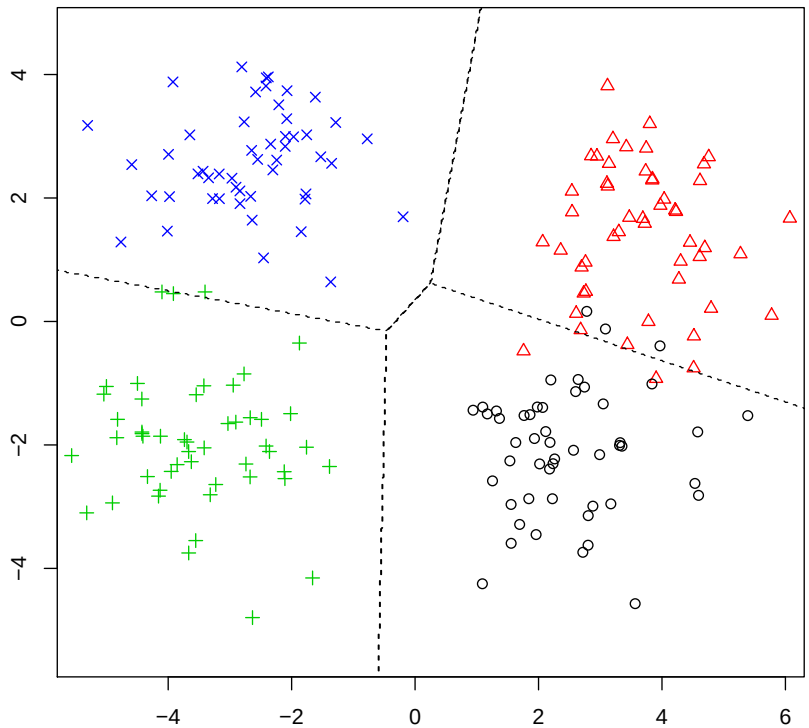
Crabs Dataset

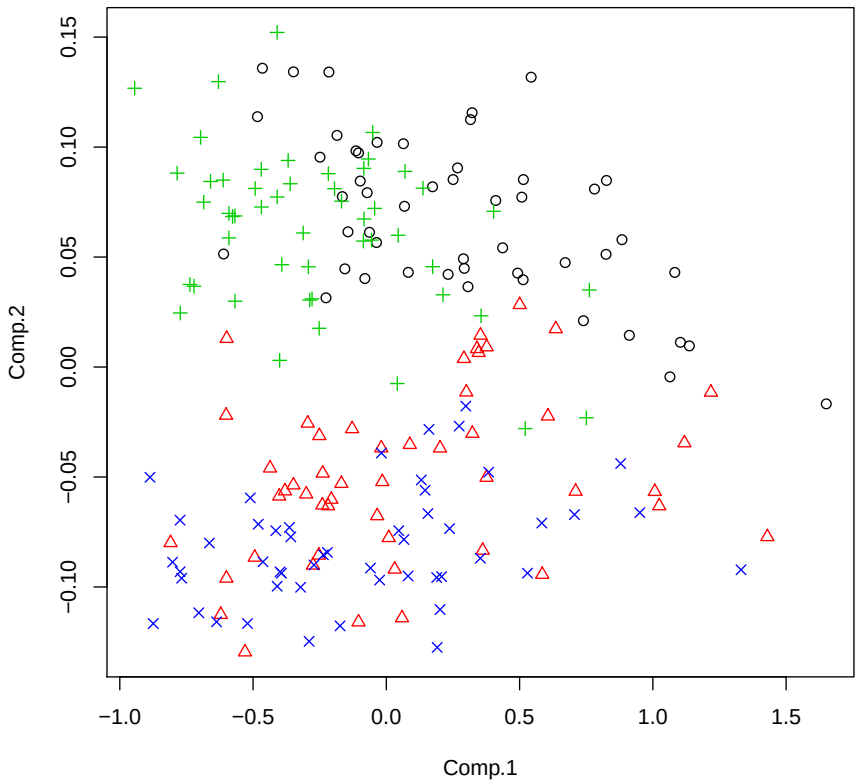
```
## display the decision boundaries
## take a lattice of points in LD-space
x <- seq(-6,7,0.02)
y <- seq(-6,7,0.02)
z <- as.matrix(expand.grid(x,y))
m <- length(x)
n <- length(y)

## perform LDA on first two discriminant directions
cb.lda_new <- lda(cb.ldp12,ct)
## predict onto the grid
cb.ldpp <- predict(cb.lda_new,z)$class

## classes are 1,2,3 and 4 so set contours
## at 1.5,2.5 and 3.5
contour(x,y,matrix(cb.ldpp,m,n),
        levels=c(1.5,2.5,3.5),
        add=TRUE,d=FALSE,lty=2)
```

Crabs Dataset



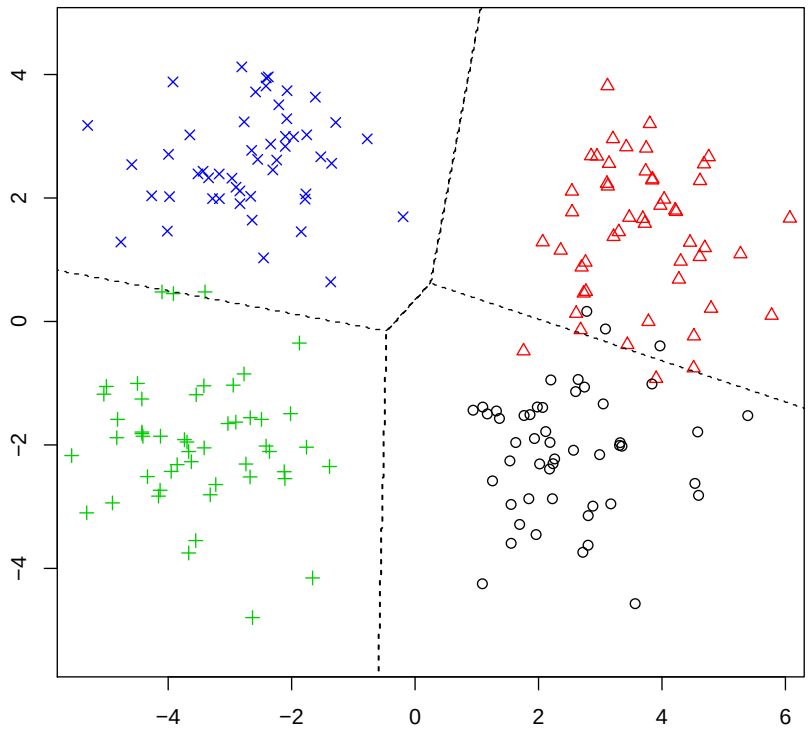
LDA vs PCA projections

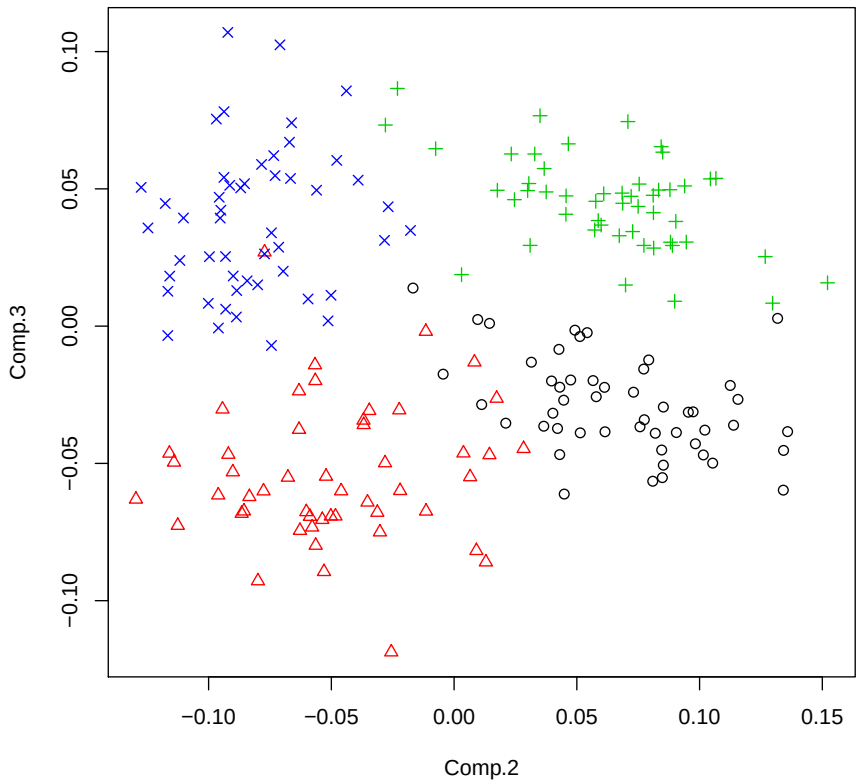




LDA separates the groups better.

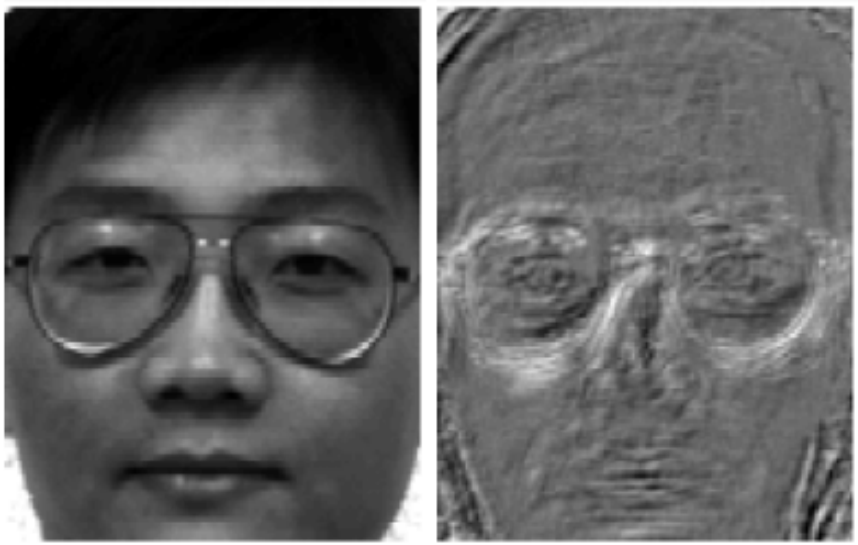
LDA vs PCA projections





LDA separates the groups better.

Fisherfaces



Eigenfaces vs. Fisherfaces, Belhumeur et al. 1997

http://ieeexplore.ieee.org/document/598228/

Conditional densities with different covariances

Given training data with K classes, assume a parametric form for conditional density $g_k(x)$, where for each class

$$X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k),$$

i.e., instead of assuming that every class has a different mean μ_k with the **same** covariance matrix Σ (LDA), we now allow each class to have its own covariance matrix.
Considering $\log \pi_k g_k(x)$ as before,

$$\begin{aligned}
\log \pi_k g_k(x) &= \text{const} + \log(\pi_k) - \frac{1}{2}\left(\log|\Sigma_k| + (x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right) \\
&= \text{const} + \log(\pi_k) - \frac{1}{2}\left(\log|\Sigma_k| + \mu_k^T\Sigma_k^{-1}\mu_k\right) \\
&\quad + \mu_k^T\Sigma_k^{-1}x - \frac{1}{2}x^T\Sigma_k^{-1}x \\
&= a_k + b_k^T x + x^T c_k x.
\end{aligned}$$

A **quadratic** discriminant function instead of linear.

Quadratic decision boundaries

Again, by considering that we choose class k over k',

$$\begin{aligned} a_k + b_k^T x + x^T c_k x - (a_{k'} + b_{k'}^T x + x^T c_{k'} x) \\ = a_\star + b_\star^T x + x^T c_\star x > 0 \end{aligned}$$

we see that the decision boundaries of the Bayes Classifier are quadratic surfaces.

- The plug-in Bayes Classifer under these assumptions is known as the **Quadratic Discriminant Analysis** (QDA) Classifier.

QDA

LDA classifier:

$$f_{\text{LDA}}(x) = \underset{k \in \{1,\ldots,K\}}{\arg\min} \left\{ (x - \hat{\mu}_k)^T \hat{\Sigma}^{-1} (x - \hat{\mu}_k) - 2 \log(\hat{\pi}_k) \right\}$$

QDA classifier:

$$f_{\text{QDA}}(x) = \underset{k \in \{1,\ldots,K\}}{\arg\min} \left\{ (x - \hat{\mu}_k)^T \hat{\Sigma}_k^{-1} (x - \hat{\mu}_k) - 2 \log(\hat{\pi}_k) + \log(|\hat{\Sigma}_k|) \right\}$$

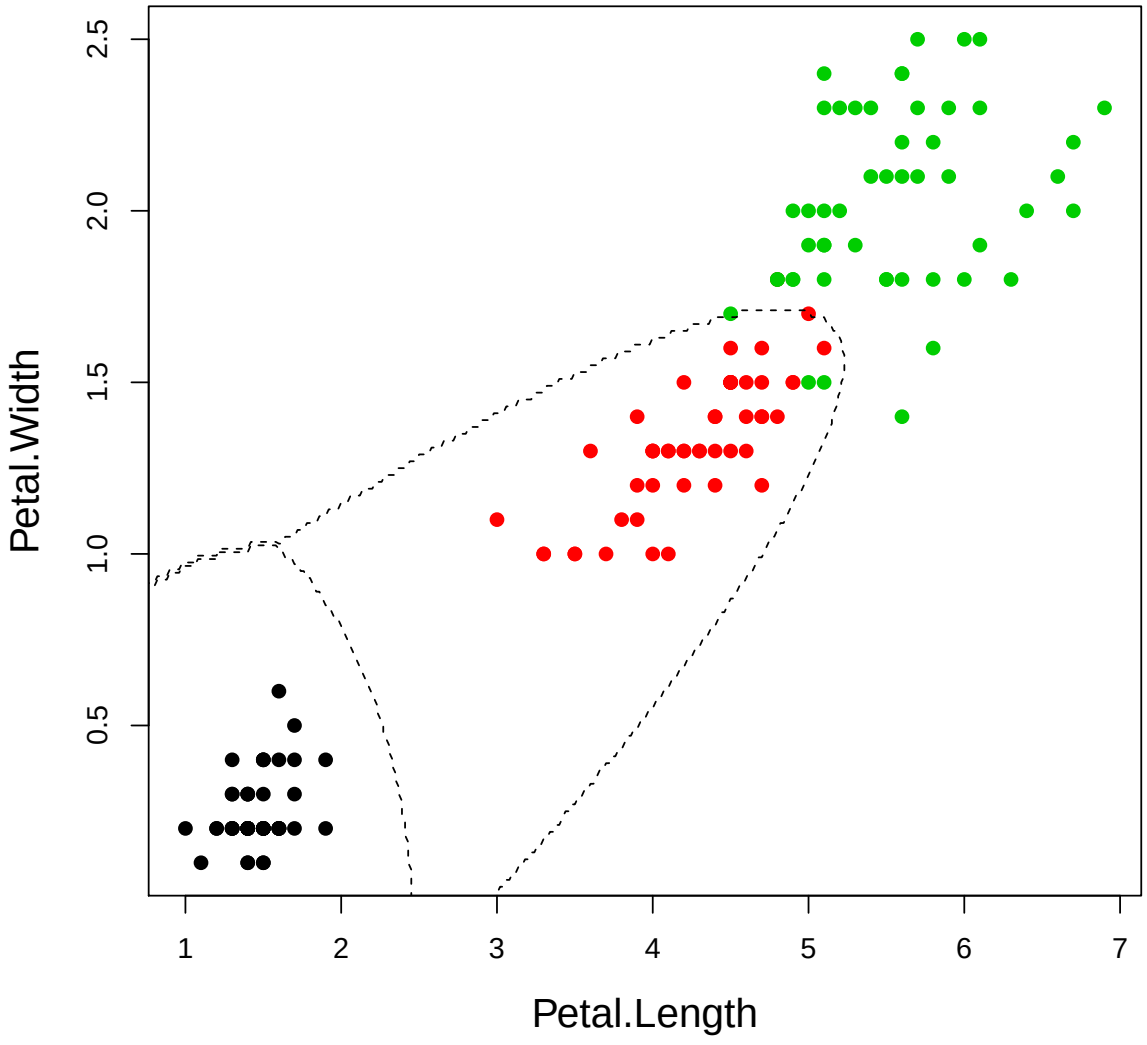
for each point $x \in \mathcal{X}$ where the plug-in estimate $\hat{\mu}_k$ is as before and $\hat{\Sigma}_k$ is (in contrast to LDA) estimated for each class $k = 1, \ldots, K$ separately:

$$\hat{\Sigma}_k = \frac{1}{n_k} \sum_{j : y_j = k} (x_j - \hat{\mu}_k)(x_j - \hat{\mu}_k)^T.$$

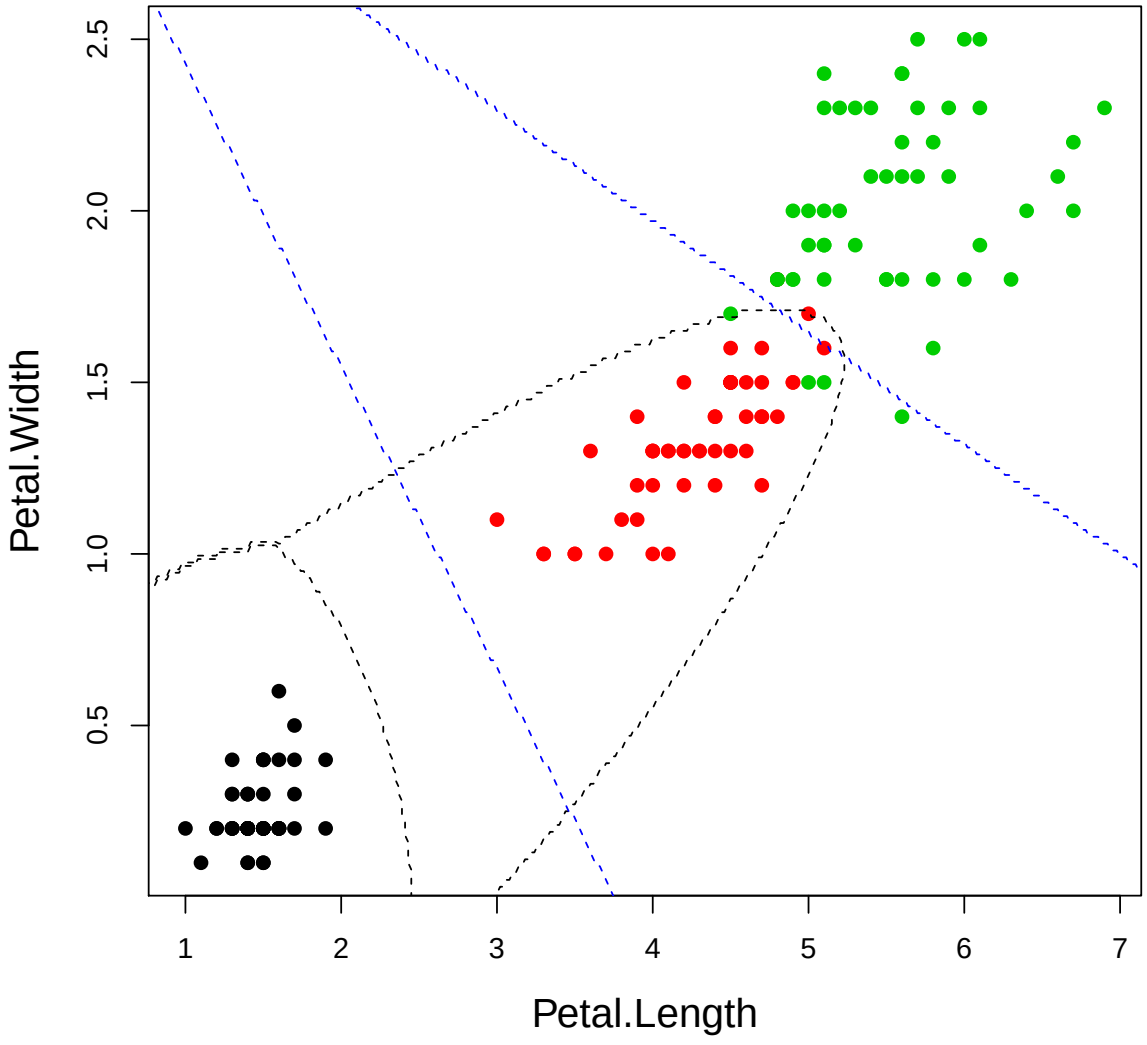
Computing and plotting the QDA boundaries.

```
##fit QDA
iris.qda <- qda(x=iris.data,grouping=ct)

##create a grid for our plotting surface
x <- seq(-6,6,0.02)
y <- seq(-4,4,0.02)
z <- as.matrix(expand.grid(x,y),0)
m <- length(x)
n <- length(y)


iris.qdp <- predict(iris.qda,z)$class
contour(x,y,matrix(iris.qdp,m,n),
        levels=c(1.5,2.5), add=TRUE, d=FALSE, lty=2)
```

Iris example: QDA boundaries



Iris example: QDA boundaries



LDA or QDA?

- Having seen both LDA and QDA in action, it is natural to ask which is the "better" classifier.
- If the covariances of different classes are very distinct, QDA will probably have an advantage over LDA.
- Parametric models are only ever approximations to the real world, allowing **more flexible decision boundaries** (QDA) may seem like a good idea. However, there is a price to pay in terms of increased variance and potential **overfitting**.

Regularized Discriminant Analysis

In the case where data is scarce , to fit

- LDA, need to estimate $K \times p + p \times p$ parameters
- QDA, need to estimate $K \times p + K \times p \times p$ parameters.

Using LDA allows us to better estimate the covariance matrix Σ. Though QDA allows more flexible decision boundaries, the estimates of the K covariance matrices Σ_k are more variable.
RDA combines the strengths of both classifiers by regularizing each covariance matrix Σ_k in QDA to the single one Σ in LDA

$$\Sigma_k(\alpha) = \alpha\Sigma_k + (1-\alpha)\Sigma \qquad \text{for some } \alpha \in [0,1].$$

This introduces a new parameter α and allows for a continuum of models between LDA and QDA to be used. Can be selected by Cross-Validation for example.