Outline

Linear Discriminant Analysis

LDA is the most well-known and simplest example of plug-in classification. Assume a parametric form for $f_k(x)$ where for each class k, the distribution of X, conditional on $Y = k$, is

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

i.e. classes have different means with the *same* covariance matrix Σ. For a new observation x,

$$\begin{aligned} P(Y = k|X = x) &\propto \pi_k f_k(x) \\ &\propto \frac{\pi_k}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k)\right\} \end{aligned}$$

As $\arg\max_{k=1,\ldots,K} g(k) = \arg\min_{k=1,\ldots,K} -2\log g(k)$ for any real-valued function g, choose k to minimize

$$-2\log P(Y=k|X=x) \quad \propto \quad (x-\mu_k)^T\Sigma^{-1}(x-\mu_k) - 2\log(\pi_k) + \text{const.}$$

where the constant does not depend on the class k.
The quantity $(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)$ is called the Malahanobis distance. It measures the distance between x and μ_k in the metric given by Σ.
Notice that if $\Sigma = I_p$ and $\pi_k = \frac{1}{K}$, $\hat{Y}(x)$ simply chooses the class k with the nearest (in the Euclidean sense) mean μ_k.

Expanding the discriminant $(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)$, the term $-2\log P(Y=k|X=x)$ is seen to be proportional to

$$\mu_k^T\Sigma^{-1}\mu_k - 2\mu_k^T\Sigma^{-1}x + x^T\Sigma^{-1}x - 2\log(\pi_k) + \text{const}$$
$$= \mu_k^T\Sigma^{-1}\mu_k - 2\mu_k^T\Sigma^{-1}x - 2\log(\pi_k) + \text{const},$$

where the constant does not depend on the class k.
Setting $a_k = \mu_k^T\Sigma^{-1}\mu_k - 2\log(\pi_k)$ and $b_k = -2\Sigma^{-1}\mu_k$, we obtain

$$-2\log P(Y=k|X=x) = a_k + b_k^T x + \text{const}$$

i.e. a *linear discriminant function*.
Considering when we choose class k over k',

$$\begin{aligned} a_k + b_k^T x + \text{const}(x) &< a_{k'} + b_{k'}^T x + \text{const} \\ \Leftrightarrow a_\star + b_\star^T x &< 0 \end{aligned}$$

where $a_\star = a_k - a_{k'}$ and $b_\star = b_k - b_{k'}$.
Shows that the Bayes Classifier 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 *Linear Discriminant Analysis Classifier*.

Parameter Estimation and 'Plug-In' Classifiers

Remember that upon assuming a parametric form for the $f_k(x)$'s, the optimal classification procedure under 0-1 loss is

$$\hat{Y}(x) = \underset{k=1,\ldots,K}{\arg\max} \quad \pi_k f_k(x)$$

LDA proposes multivariate normal distributions for $f_k(x)$.
However, we still don't know what the parameters μ_k, $k = 1, \ldots, K$ and Σ that determine f_k. The statistical task becomes one of finding good estimates for these quantities and plugging them into the derived equations to give the *'Plug-In' Classifier*

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

The a priori probabilities $\pi_k = P(Y = k)$ are simply estimated by the empirical proportion of samples of class k, $\hat{\pi}_k = |\{i : Y_i = k\}|/n$.

For estimation of Σ and μ, looking at the log-likelihood of the training set,

$$\begin{aligned}\ell(\mu_1, \dots, \mu_K) &= -\sum_{k=1}^{K} \sum_{j:Y_j=k} \frac{1}{2}(X_j - \mu_k)^T \Sigma^{-1} (X_j - \mu_k) \\ &\quad -\frac{1}{2} n \log |\Sigma| + \text{const.}\end{aligned}$$

Let $n_k = \#\{j : Y_j = k\}$ be the number of observations in class k. The log-likelihood is maximised by

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{j:Y_j=k} X_j, \qquad \hat{\Sigma} = \frac{1}{n} \sum_{k=1}^{K} \sum_{j:Y_j=k} (X_j - \hat{\mu}_k)(X_j - \hat{\mu}_k)^T.$$

The best classifier under the assumption that $X|Y = k \sim \mathcal{N}_p(\hat{\mu}_k, \hat{\Sigma})$ with plug-in estimates of μ and Σ is therefore given by

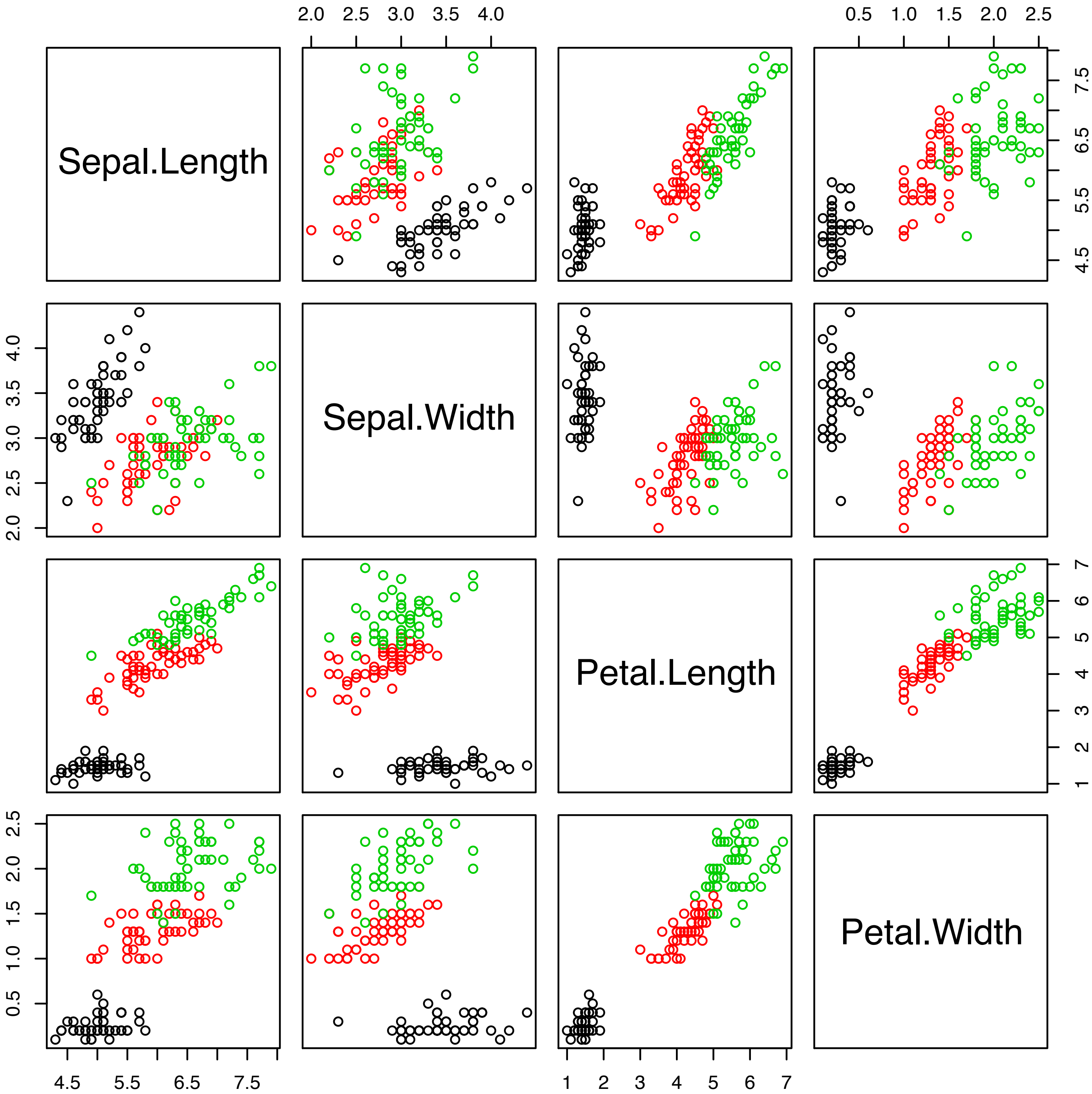
$$\hat{Y}_{lda}(x) = \underset{k=1,\dots,K}{\arg\min} \left\{ (x - \hat{\mu}_k)^T \hat{\Sigma}^{-1} (x - \hat{\mu}_k) - 2\log(\hat{\pi}_k) \right\}$$

for each point $x \in \mathcal{X}$.
Can also be written as

$$\hat{Y}_{lda}(x) = \underset{k=1,\dots,K}{\arg\min} \left\{ \hat{\mu}_k^T \hat{\Sigma}^{-1} \hat{\mu}_k - 2\hat{\mu}_k^T \hat{\Sigma}^{-1} x - 2\log(\hat{\pi}_k) \right\}.$$

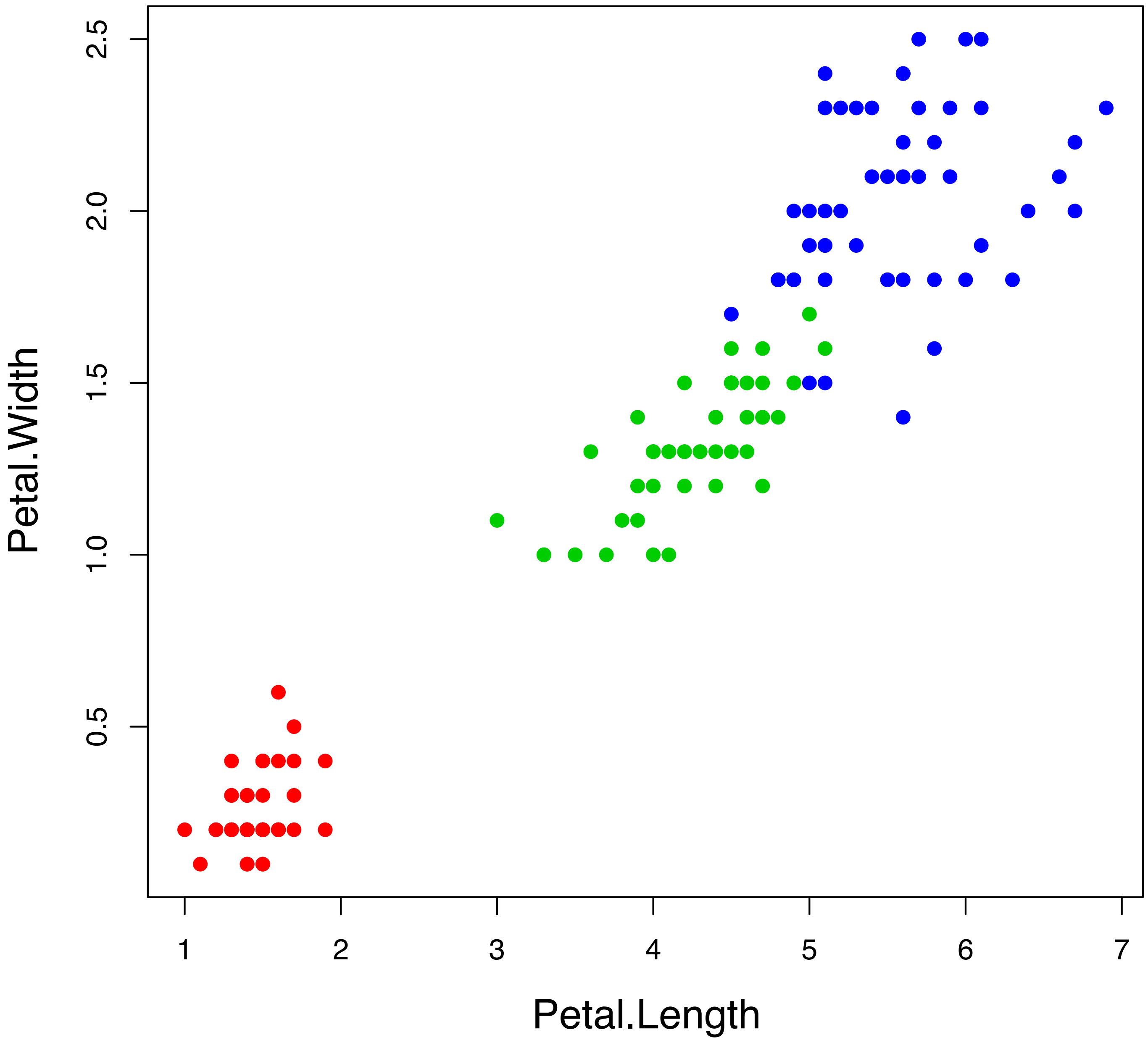
Iris example

```
library(MASS)
data(iris)

##save class labels
ct <- rep(1:3,each=50)
##pairwise plot
pairs(iris[,1:4],col=ct)

##save petal.length and petal.width
iris.data <- iris[,3:4]
plot(iris.data,col=ct+1,pch=20,cex=1.5,cex.lab=1.4)
```



Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
2.0 2.5 3.0 3.5 4.0
0.5 1.0 1.5 2.0 2.5
4.5 5.5 6.5 7.5
2.0 2.5 3.0 3.5 4.0
1 2 3 4 5 6 7
4.5 5.5 6.5 7.5
0.5 1.0 1.5 2.0 2.5
1 2 3 4 5 6 7

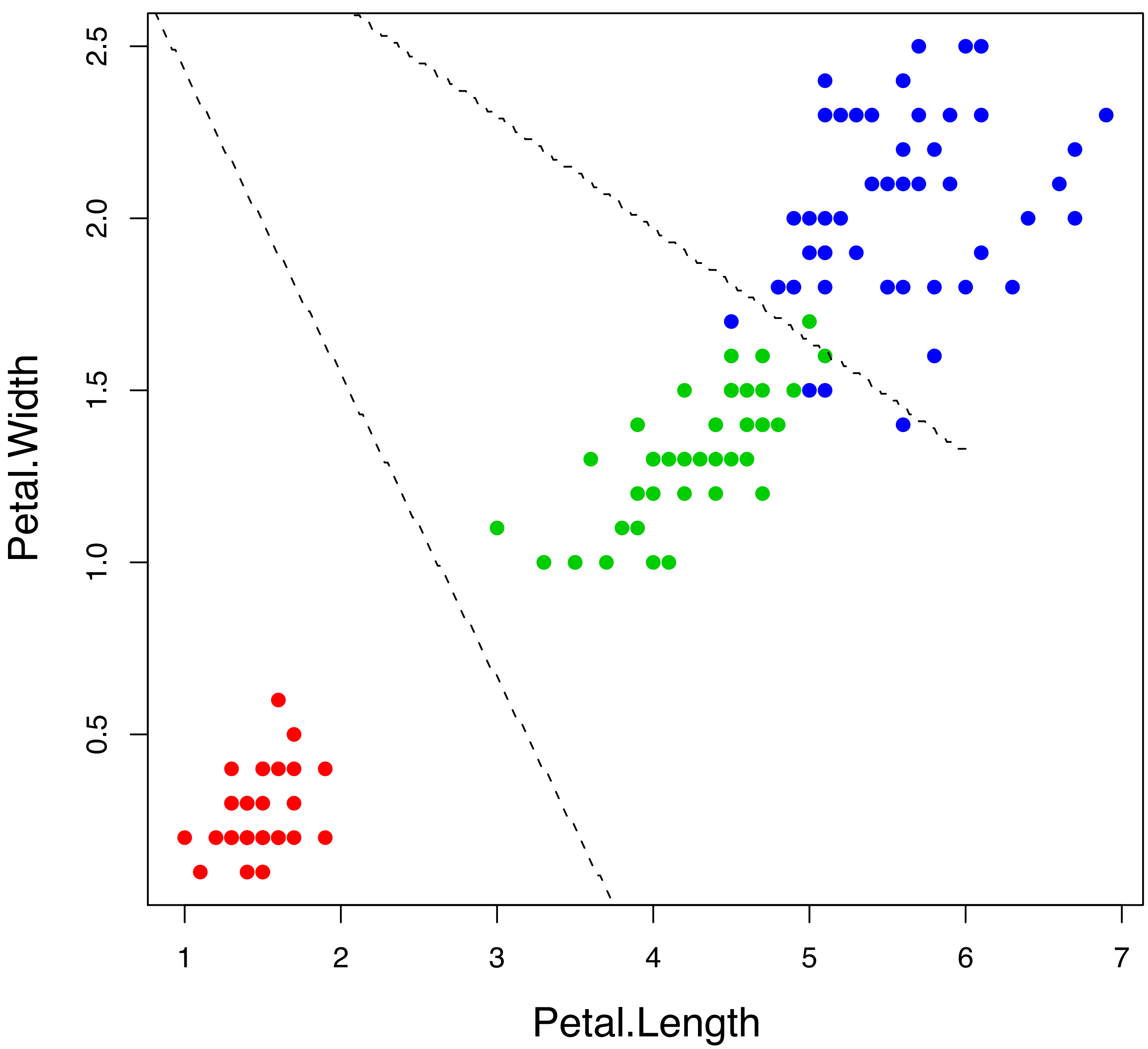
Just focus on two predictor variables.



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(-6,6,0.02)
y <- seq(-4,4,0.02)
z <- as.matrix(expand.grid(x,y),0)
m <- length(x)
n <- length(y)

##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)
```

LDA boundaries.



Fishers Linear Discriminant Analysis

We have derived LDA as the plug-in Bayes classifier under the assumption of multivariate normality for all classes with common covariance matrix.
Alternative view (without making any assumption on underlying densities):
Find a direction $a \in \mathbb{R}^p$ to maximize the variance ratio

$$\frac{a^T B a}{a^T \Sigma a},$$

where

$$\Sigma = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \mu_{Y_i})(X_i - \mu_{Y_i})^\top \qquad \text{(within class covariance)}$$

$$B = \frac{1}{n-1} \sum_{k=1}^{K} n_k (\mu_{Y_i} - \bar{X})(\mu_{Y_i} - \bar{X}))^\top \qquad \text{(between class covariance)}$$

B has rank at most $K-1$.

Discriminant Coordinates

The variance ratio satisfies

$$\frac{a^T B a}{a^T \Sigma a} = \frac{b^T (\Sigma^{-\frac{1}{2}})^T B \Sigma^{-\frac{1}{2}} b}{b^T b},$$

where $b = \Sigma^{\frac{1}{2}} a$ and $B^* = (\Sigma^{-\frac{1}{2}})^T B \Sigma^{-\frac{1}{2}}$.
The maximization over b is achieved by the first eigenvector v_1 of B^*. We also look at the remaining eigenvectors v_l associated to the non-zero eigenvalues and defined the discriminant coordinates as $a_l = \Sigma^{-\frac{1}{2}} v_l$.
These directions a_l span exactly the space of all linear discriminant functions for all pairwise comparisons and are often used for plotting (ie in the function `lda`).
Data are then projected onto these directions (these vectors are given as the "linear discriminant" functions in the R-function `lda`).

Crabs data example

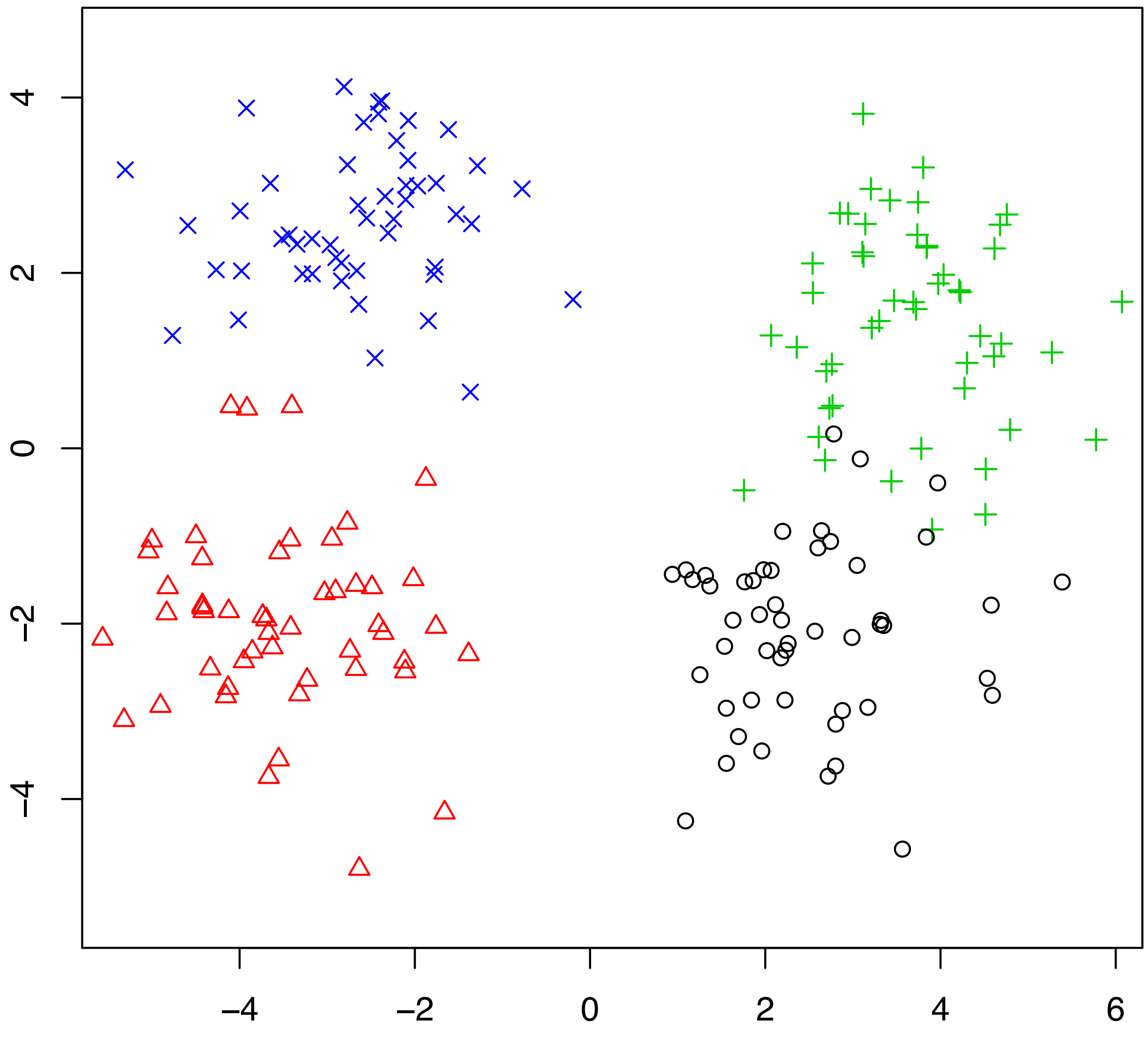
Crabs data, again.

```
library(MASS)
data(crabs)

## numeric and text class labels
ct <- as.numeric(crabs[,1])-1+2*(as.numeric(crabs[,2])-1)

## Projection on Fisher's linear discriminant directions
print(cb.lda <- lda(log(crabs[,4:8]),ct))
```

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

Prior probabilities of groups:
   0    1    2    3 
0.25 0.25 0.25 0.25 

Group means:
        FL       RW       CL       CW       BD
0 2.564985 2.475174 3.312685 3.462327 2.441351
1 2.852455 2.683831 3.529370 3.649555 2.733273
2 2.672724 2.443774 3.437968 3.578077 2.560806
3 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 
```

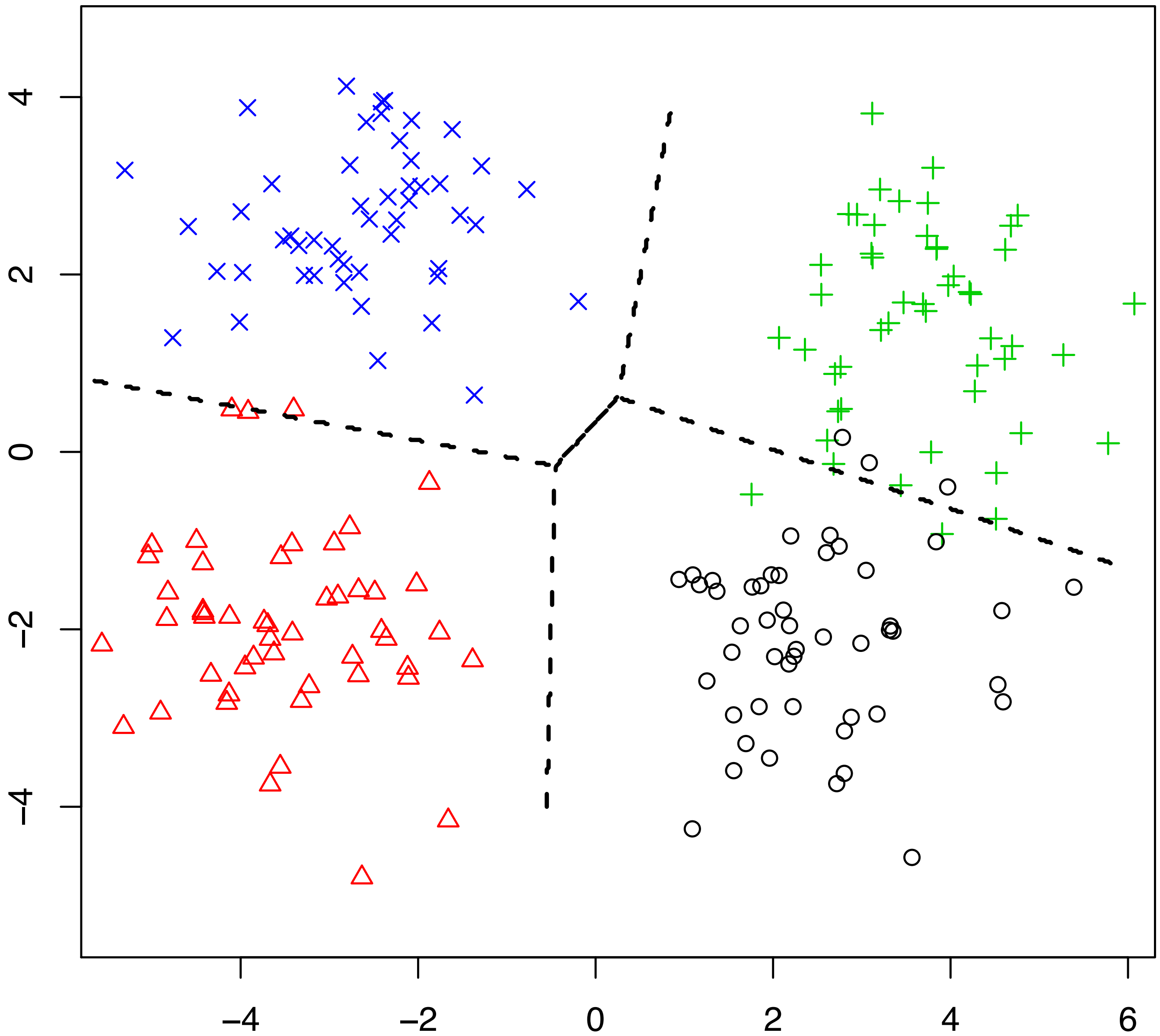
Plot predictions

```
cb.ldp <- predict(cb.lda)
eqscplot(cb.ldp$x,pch=ct+1,col=ct+1)
```



```
> ct
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 3 3 3 3 3 3
[112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[149] 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

> predict(cb.lda)
$class
  [1] 2 2 2 2 2 2 0 2 2 0 2 0 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 3 3 3 3 3 3
[112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[149] 3 3 1 3 3 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Levels: 0 1 2 3

$posterior
                0            1            2            3
1   4.058456e-02 1.579991e-10 9.594150e-01 4.367517e-07
2   4.912087e-01 2.057493e-09 5.087911e-01 2.314634e-07
3   2.001047e-02 4.368642e-16 9.799895e-01 2.087757e-13
4   7.867144e-04 9.148327e-15 9.992133e-01 2.087350e-09
5   2.094626e-03 2.381970e-11 9.979020e-01 3.335500e-06
6   3.740294e-03 3.170411e-13 9.962597e-01 2.545022e-08
7   7.291360e-01 1.625743e-09 2.708639e-01 6.637005e-08
```

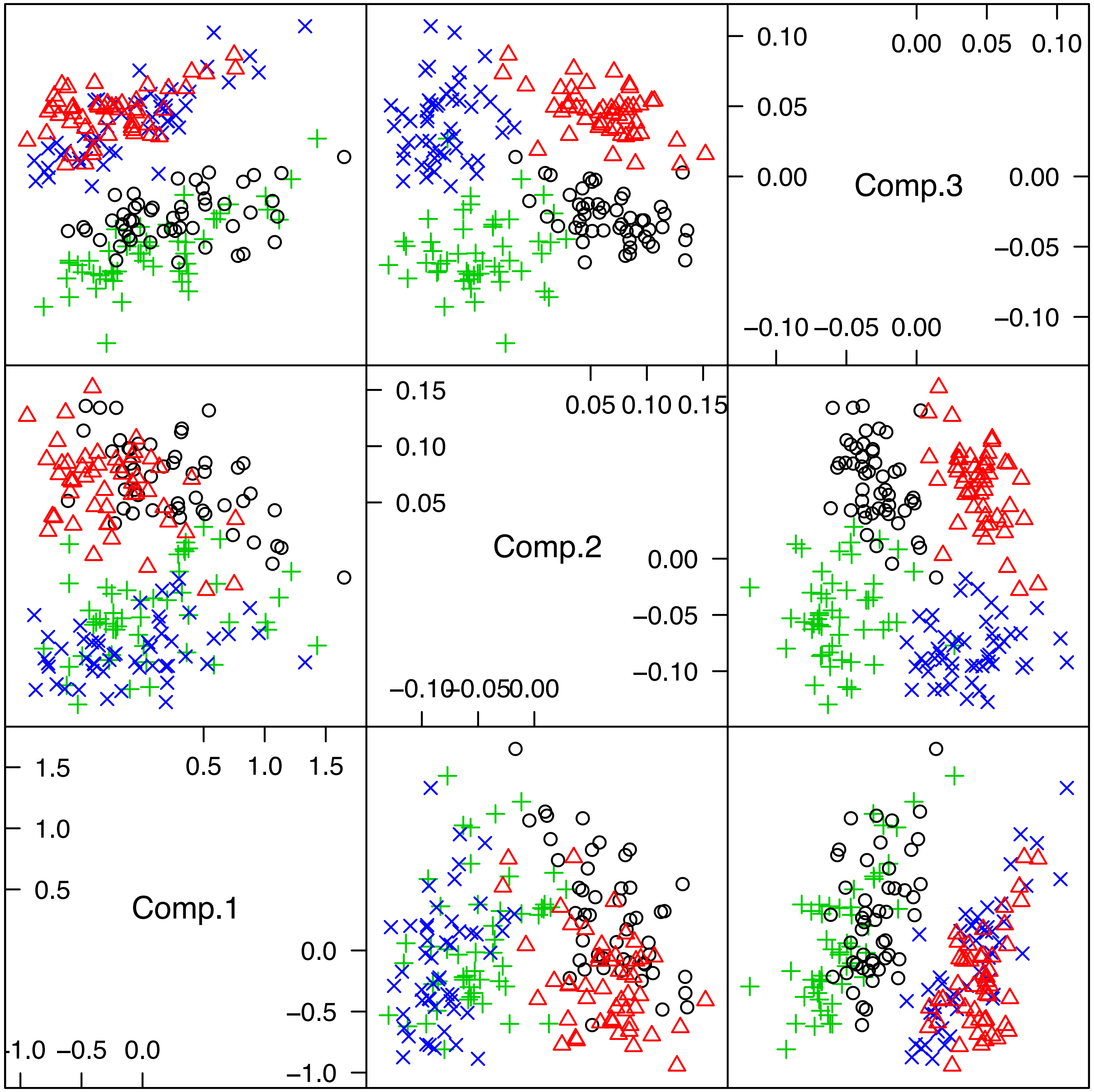
```
## display the decision boundaries
## take a lattice of points in LD-space
x <- seq(-6,6,0.02)
y <- seq(-4,4,0.02)
z <- as.matrix(expand.grid(x,y,0))
m <- length(x)
n <- length(y)


## predict onto the grid
cb.ldap <- lda(cb.ldp$x,ct)
cb.ldpp <- predict(cb.ldap,z)$class

## classes are 0,1,2 and 3 so set contours
## at 0.5,1.5 and 2.5
contour(x,y,matrix(cb.ldpp,m,n),
           levels=c(0.5,2.5),
           add=TRUE,d=FALSE,lty=2,lwd=2)
```



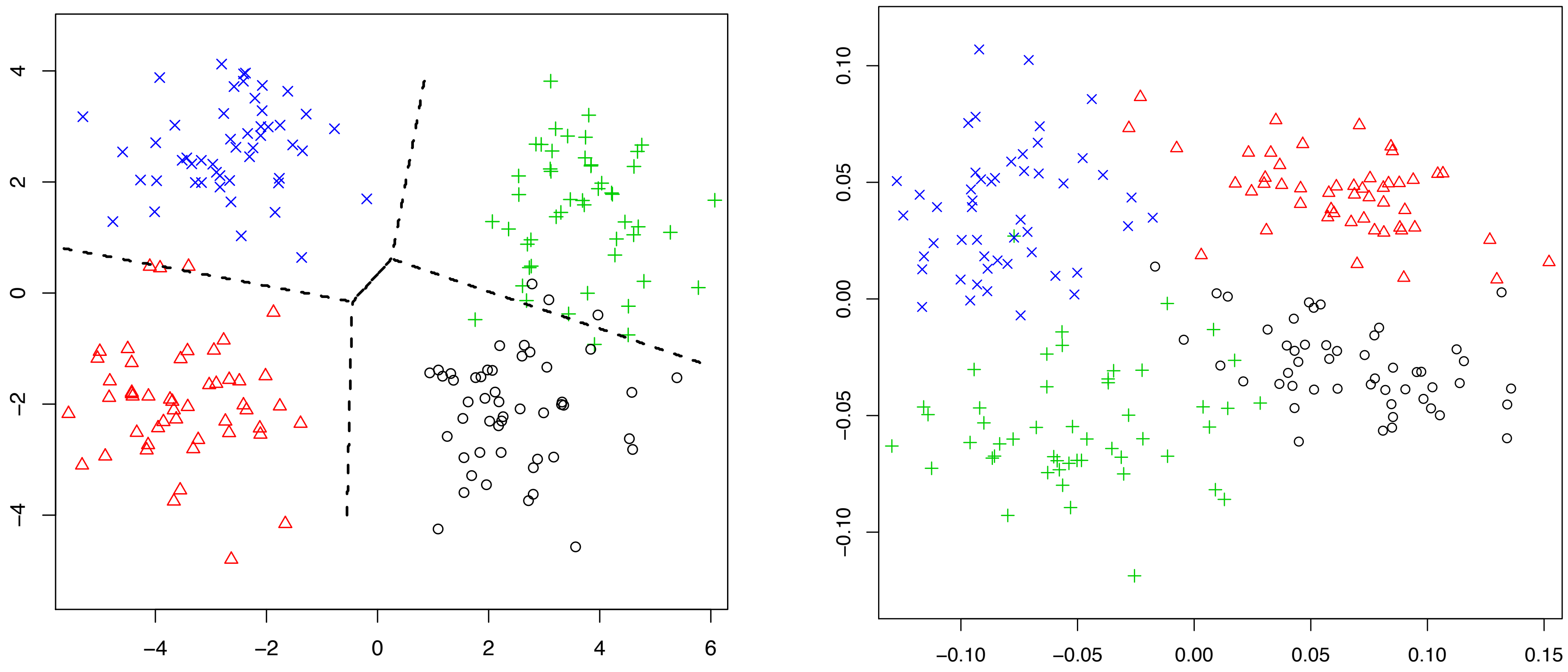
4
2
0
−2
−4
−4
−2
0
2
4
6

Compare with PCA plots.

```
library(lattice)
cb.pca <- princomp(log(crabs[,4:8]))
cb.pcp <- predict(cb.pca)
splom(~cb.pcp[,1:3],pch=ct+1,col=ct+1)
```



Scatter Plot Matrix



LDA separates the groups better.