Statistical Data Mining and Machine Learning Hilary Term 2016

Dino Sejdinovic
Department of Statistics
Oxford

Slides and other materials available at:
http://www.stats.ox.ac.uk/~sejdinov/sdmml

Last Time: **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 $\hat{\pi}_k$:

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

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 $\hat{\pi}_k$:

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

- Thus, LDA classification can be implemented as the following two steps:
  (1) **Sphere** the data with respect to **the common covariance estimate** $\hat{\Sigma} = \frac{1}{n} \sum_{k=1}^{K} \sum_{j:y_j=k} (x_j - \hat{\mu}_k)(x_j - \hat{\mu}_k)^\top$:

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

  (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

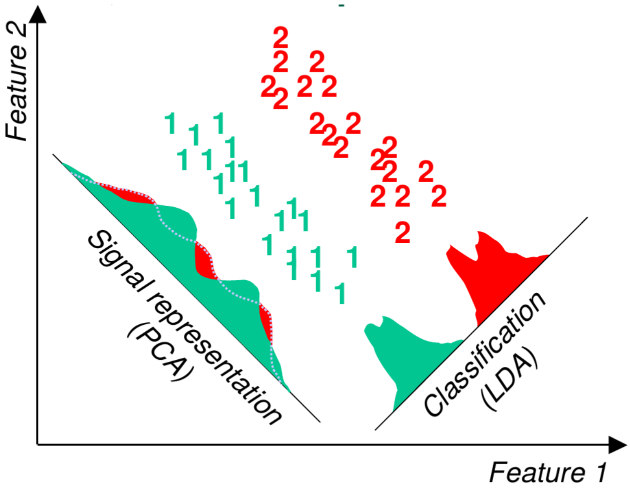
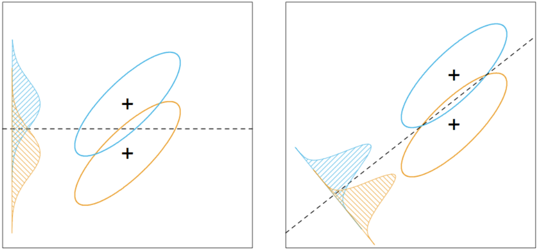


Figure by R. Gutierrez-Osuna

Discriminant Coordinates



- Find a direction $v \in \mathbb{R}^p$ to maximize the between-class variance relative to the within-class variance of the projection $v^\top X$:

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

where

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

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

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

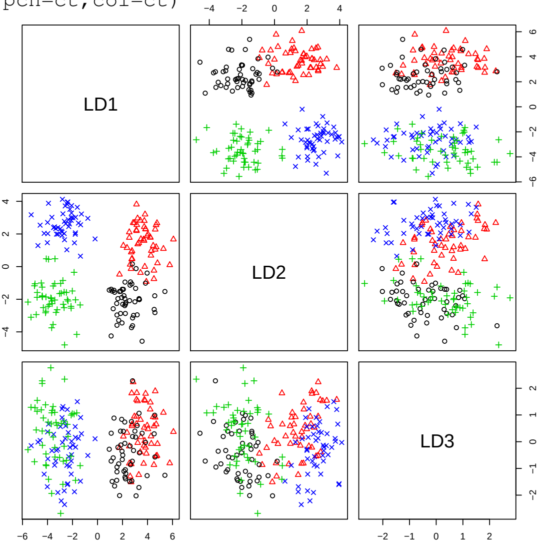
```
library(MASS)
data(crabs)

## 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
```

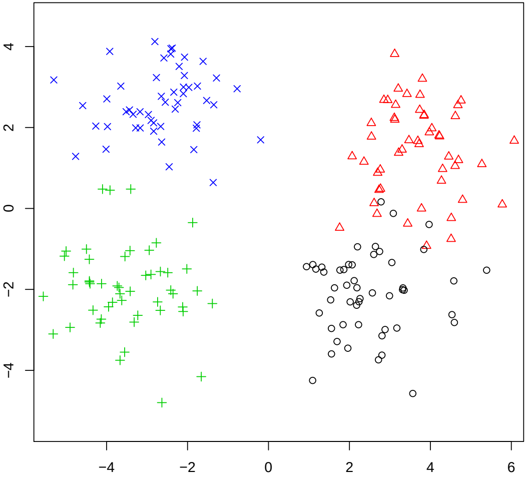
Crabs Dataset

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



Crabs Dataset

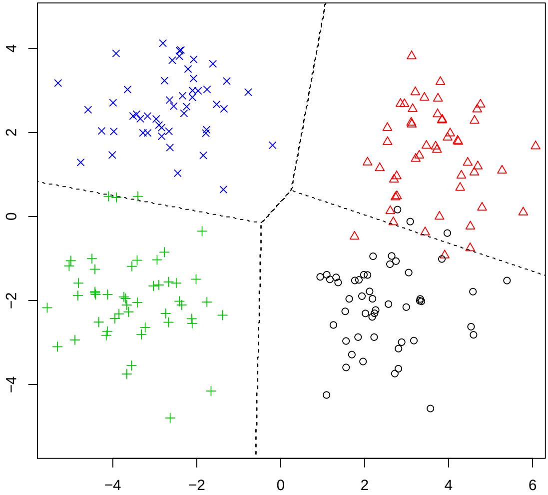
```
cb.ldp12 <- cb.ldp$x[,1:2]
eqscplot(cb.ldp12,pch=ct,col=ct)
```



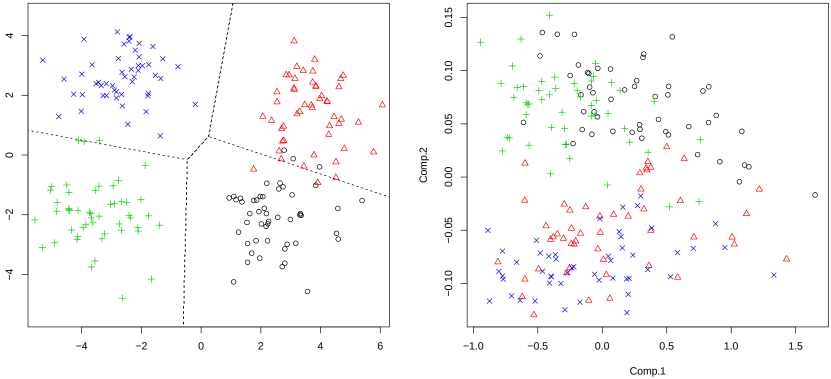
Crabs Dataset

```
## 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

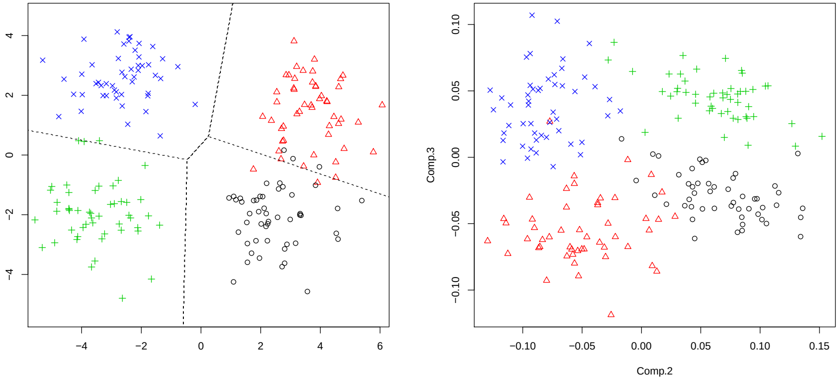


LDA vs PCA projections



LDA separates the groups better.

LDA vs PCA projections



LDA separates the groups better.

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

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

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,

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

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:

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

QDA classifier:

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

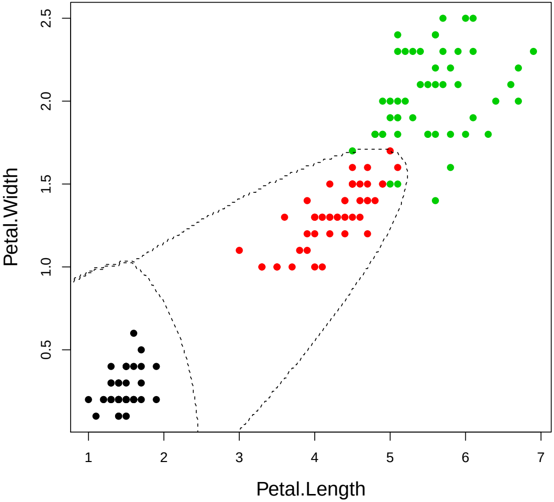
for each point $x \in \mathcal{X}$ where the plug-in estimate $\hat{\mu}_k$ is as before and $\hat{\Sigma}_k$ is (in contrast to LDA) estimated for each class $k = 1, \ldots, K$ separately:

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

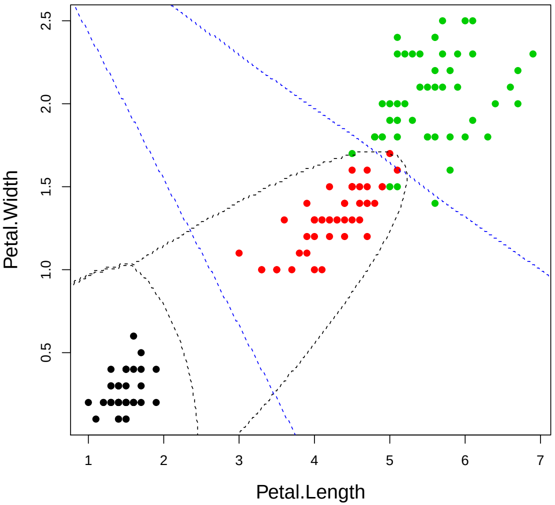
Computing and plotting the QDA boundaries.

```
##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**.