HT2015: SC4
Statistical Data Mining and Machine Learning

Dino Sejdinovic
Department of Statistics
Oxford

http://www.stats.ox.ac.uk/~sejdinov/sdmml.html

Fisher's 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 estimated class probability $\hat{\pi}_k$:

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

Fisher's Linear Discriminant Analysis

- In LDA, data vectors are classified based on Mahalanobis distance to class means.
- All class means lie on a $(K-1)$-dimensional affine subspace: Decisions are 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 $\mathbb{R}^{K-1}$.
- As with PCA, we can visualize the structure in the data by choosing an appropriate basis for the subspace and projecting data onto it.
- Change of basis that finds **directions that best separate classes**.

LDA projections

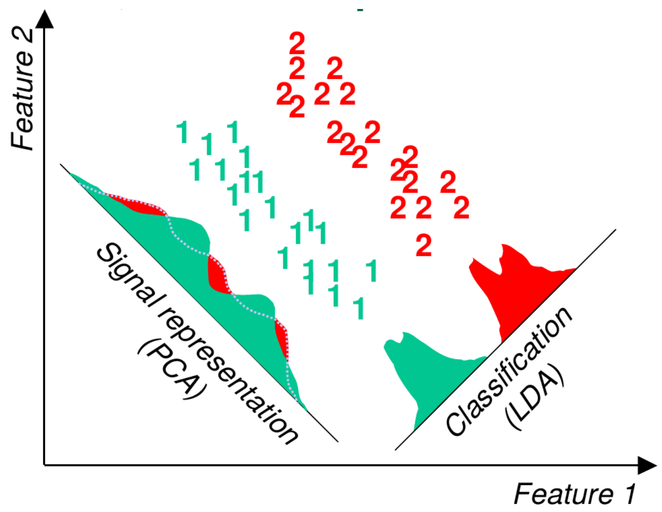
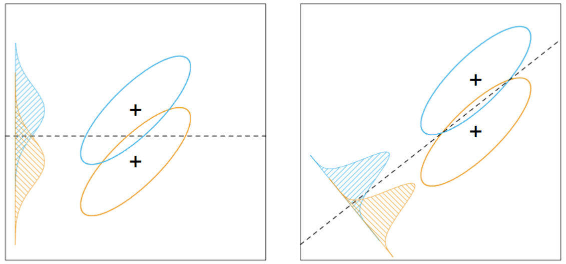


Figure by R. Gutierrez-Osuna

Discriminant Coordinates



- Find a direction $v \in \mathbb{R}^p$ to maximize the variance ratio

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

where

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

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

B has rank at most $K-1$.

Figure from Hastie et al.

Discriminant Coordinates

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

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

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

Crabs Dataset

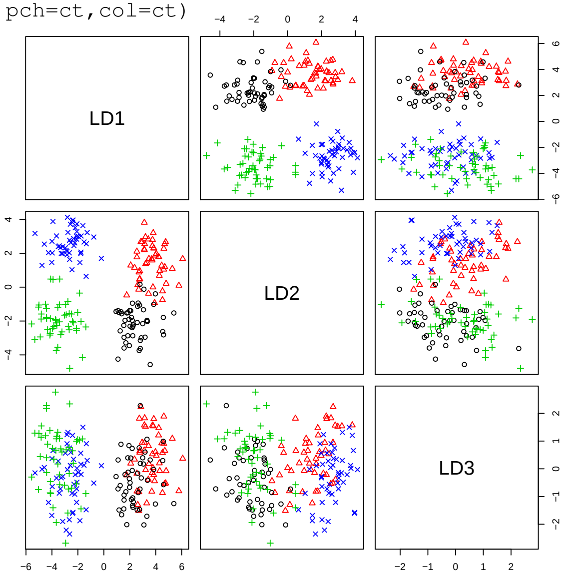
```
library(MASS)
data(crabs)

## 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
```

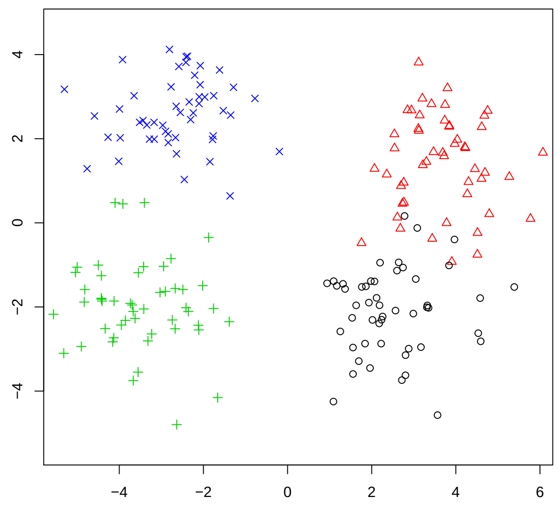
Crabs Dataset

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



Crabs Dataset

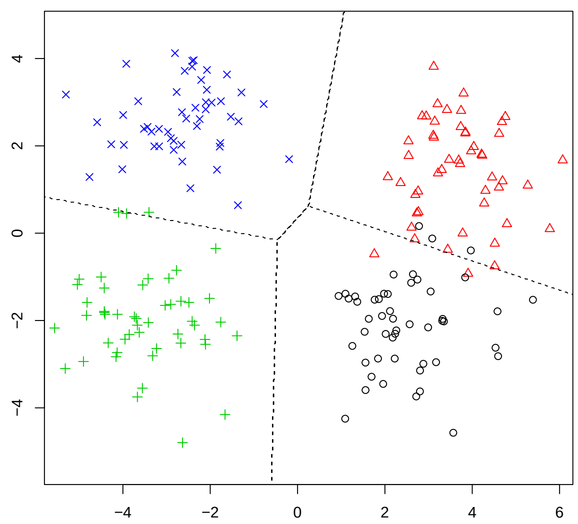
```
cb.ldp12 <- cb.ldp$x[,1:2]
eqscplot(cb.ldp12,pch=ct,col=ct)
```



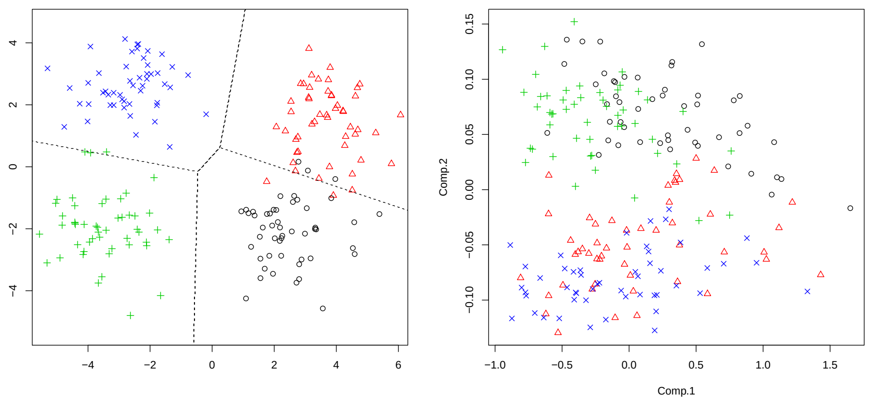
Crabs Dataset

```
## 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

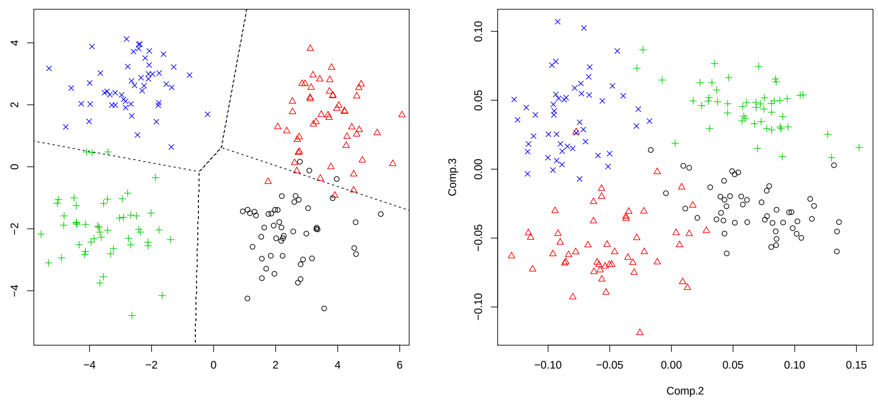


LDA vs PCA projections



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,

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

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

Quadratic decision boundaries

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

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

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:

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

QDA classifier:

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

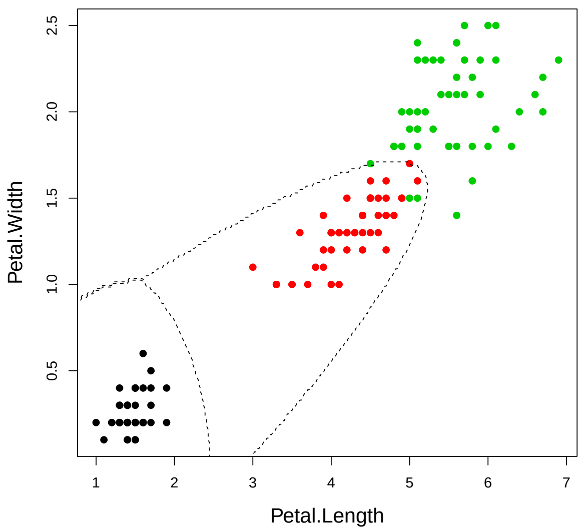
for each point $x \in \mathcal{X}$ where the plug-in estimate $\hat{\mu}_k$ is as before and $\hat{\Sigma}_k$ is (in contrast to LDA) estimated for each class $k = 1, \ldots, K$ separately:

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

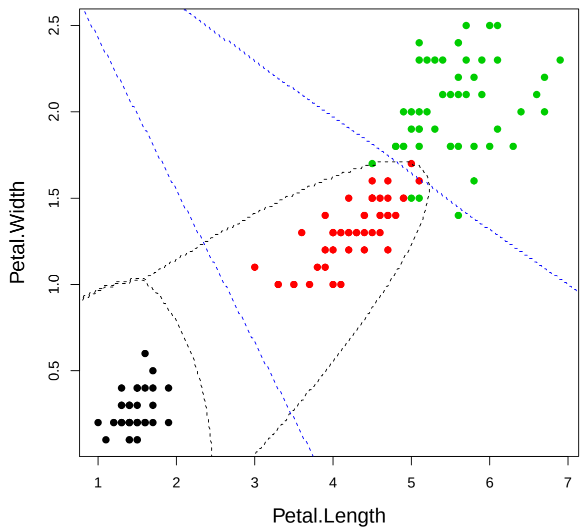
Computing and plotting the QDA boundaries.

```
##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**.

Naïve Bayes

- Assume we are interested in classifying documents, e.g., scientific articles or emails.
- A basic standard model for text classification consists of considering a pre-specified dictionary of p words and summarizing each document i by a binary vector x_i where

$$x_i^{(j)} = \begin{cases} 1 & \text{if word } j \text{ is present in document} \\ 0 & \text{otherwise.} \end{cases}$$

- Presence of the word j is the j-the feature/dimension.
- To implement a probabilistic classifier, we need to model for the conditional probability mass function $g_k(x) = \mathbb{P}(X = x | Y = k)$ for each class $k = 1, ..., K$.

Naïve Bayes

- Naïve Bayes is a plug-in classifier which **ignores feature correlations**[1] and assumes:

$$\begin{aligned} g_k(x_i) = \mathbb{P}(X = x_i | Y = k) &= \prod_{j=1}^{p} \mathbb{P}(X^{(j)} = x_i^{(j)} | Y = k) \\ &= \prod_{j=1}^{p} (\phi_{kj})^{x_i^{(j)}} (1 - \phi_{kj})^{1 - x_i^{(j)}}, \end{aligned}$$

  where we denoted parametrized conditional PMF with $\phi_{kj} = \mathbb{P}(X^{(j)} = 1 | Y = k)$ (probability that j-th word appears in class k document).
- Given dataset, the MLE of the parameters is:

$$\hat{\pi}_k = \frac{n_k}{n}, \qquad \hat{\phi}_{kj} = \frac{\sum_{i:y_i=k} x_i^{(j)}}{n_k}.$$

[1] given the class, it assumes each word appears in a document independently of all others

Naïve Bayes

- MLE:

$$\hat{\pi}_k = \frac{n_k}{n}, \qquad \hat{\phi}_{kj} = \frac{\sum_{i:y_i=k} x_i^{(j)}}{n_k}.$$

- One problem: if the ℓ-th word did not appear in documents labelled as class k then $\hat{\phi}_{k\ell} = 0$ and

$$\begin{aligned} &\mathbb{P}(Y = k | X = x \text{ with } \ell\text{-th entry equal to 1}) \\ &\propto \hat{\pi}_k \prod_{j=1}^{p} \left(\hat{\phi}_{kj}\right)^{x^{(j)}} \left(1 - \hat{\phi}_{kj}\right)^{1 - x^{(j)}} = 0 \end{aligned}$$

  i.e. we will never attribute a new document containing word ℓ to class k (regardless of other words in it).
- An example of **overfitting**.