Supervised Learning

Unsupervised learning:

- To "extract structure" and postulate hypotheses about data generating process from observations $x_1, \ldots, x_n$.
- Visualize, summarize and compress data.

We have seen how response or grouping variables are used to validate the usefulness of the extracted structure.

Supervised learning:

- In addition to the n observations of X, we also have a response variable $Y \in \mathcal{Y}$.
- Techniques for predicting Y given X.
  - Classification: discrete responses, e.g. $\mathcal{Y} = \{+1, -1\}$ or $\{1, \ldots, K\}$.
  - Regression: a numerical value is observed and $\mathcal{Y} = \mathbb{R}$.

Given training data (x_i, y_i), $i = 1, \ldots, n$, the goal is to accurately predict the class or response Y on new observations of X.

Regression Example: Boston Housing

The original data are 506 observations on 13 variables X; medv being the response variable Y.

```
crim     per capita crime rate by town
zn       proportion of residential land zoned for lots
         over 25,000 sq.ft
indus    proportion of non-retail business acres per town
chas     Charles River dummy variable (= 1 if tract bounds river;
         0 otherwise)
nox      nitric oxides concentration (parts per 10 million)
rm       average number of rooms per dwelling
age      proportion of owner-occupied units built prior to 1940
dis      weighted distances to five Boston employment centers
rad      index of accessibility to radial highways
tax      full-value property-tax rate per USD 10,000
ptratio  pupil-teacher ratio by town
b        1000(B - 0.63)^2 where B is the proportion of blacks by t
lstat    percentage of lower status of the population
medv     median value of owner-occupied homes in USD 1000's
```


Regression Example: Boston Housing

```
> str(X)
'data.frame':   506 obs. of  13 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.8
 $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0
 $ rm     : num  6.58 6.42 7.18 7.00 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2
 $ black  : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...

> str(Y)
num[1:506] 24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

Goal: predict median house price $\hat{Y}(X)$, given 13 predictor variables X of a new district.

Classification Example: Lymphoma

We have gene expression measurements X of $n = 62$ patients for $p = 4026$ genes. For each patient, Y denotes one of two subtypes of cancer. Goal: predict cancer subtype $\hat{Y}(X) \in \{0, 1\}$, given gene expressions of a new patient.

```
> str(X)
'data.frame':   62 obs. of  4026 variables:
 $ Gene 1   : num  -0.344 -1.188  0.520 -0.748 -0.868 ...
 $ Gene 2   : num  -0.953 -1.286  0.657 -1.328 -1.330 ...
 $ Gene 3   : num  -0.776 -0.588  0.409 -0.991 -1.517 ...
 $ Gene 4   : num  -0.474 -1.588  0.219  0.978 -1.604 ...
 $ Gene 5   : num  -1.896 -1.960 -1.695 -0.348 -0.595 ...
 $ Gene 6   : num  -2.075 -2.117  0.121 -0.800  0.651 ...
 $ Gene 7   : num  -1.8755 -1.8187  0.3175  0.3873  0.0414 ...
 $ Gene 8   : num  -1.539 -2.433 -0.337 -0.522 -0.668 ...
 $ Gene 9   : num  -0.604 -0.710 -1.269 -0.832  0.458 ...
 $ Gene 10  : num  -0.218 -0.487 -1.203 -0.919 -0.848 ...
 $ Gene 11  : num  -0.340  1.164  1.023  1.133 -0.541 ...
 $ Gene 12  : num  -0.531  0.488 -0.335  0.496 -0.358 ...

> str(Y)
 num [1:62] 0 0 0 1 0 0 1 0 0 0 ...
```

Decision Theory

- Suppose we made a prediction $\hat{Y} \in \mathcal{Y}$ 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:
  - **Misclassification loss** (or **0-1 loss**) for classification

$$L(Y, \hat{Y}) = \begin{cases} 0 & Y = \hat{Y} \\ 1 & Y \neq \hat{Y} \end{cases} .$$

  - **Squared loss** for regression

$$L(Y, \hat{Y}) = (Y - \hat{Y})^2.$$

- Alternative loss functions are often useful (later). For example, **weighted misclassification error** often appropriate. Or **log-likelihood loss** (sometimes shortened as **log loss**) $L(Y, \hat{p}) = -\log \hat{p}(Y)$, where $\hat{p}(k)$ is the estimated probability of class $k \in \mathcal{Y}$.

Decision Theory

- For a given loss function L, the **risk** R of a learner is given by the expected loss

$$R(\hat{Y}) = \mathbb{E}(L(Y, \hat{Y}(X))),$$

  where the expectation is with respect to the true (unknown) joint distribution (X, Y).
- The risk is unknown, but we can estimate it by the **empirical risk**:

$$R(\hat{Y}) \approx R_n(\hat{Y}) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{Y}(x_i)).$$

The Bayes Classifier

- What is the optimal classifier if the joint distribution (X, Y) were known?
- The joint distribution f of X can be written as a mixture

$$f(X) = \sum_{k=1}^{K} f_k(X) \mathbb{P}(Y = k),$$

  where, for $k = 1, \dots, K$,
  - the prior probabilities over classes are $P(Y = k) = \pi_k$
  - and distributions of X, conditional on $Y = k$, is $f_k(X)$.
- The **Bayes classifier** $\hat{Y}(X) \mapsto \{1, \dots, K\}$ is the one with minimum risk:

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

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

The Bayes Classifier

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

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

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

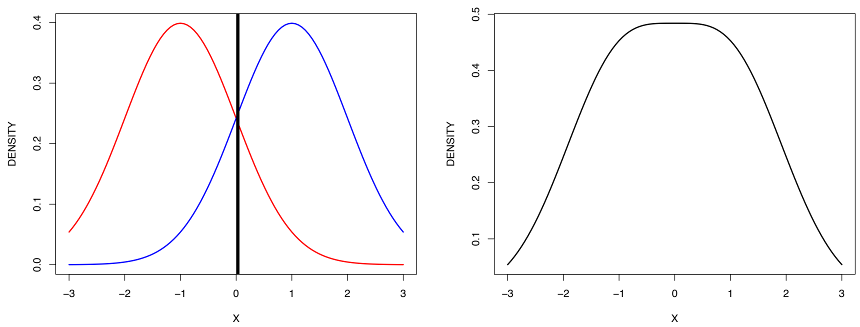
$$\begin{aligned} \hat{Y}(x) &= \underset{k=1,\dots,K}{\arg\max}\, \mathbb{P}(Y = k | X = x) = \underset{k=1,\dots,K}{\arg\max} \frac{\pi_k f_k(x)}{\sum_{k=1}^{K} \pi_k f_k(x)} \\ &= \underset{k=1,\dots,K}{\arg\max}\ \pi_k f_k(x). \end{aligned}$$

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

The Bayes Classifier

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

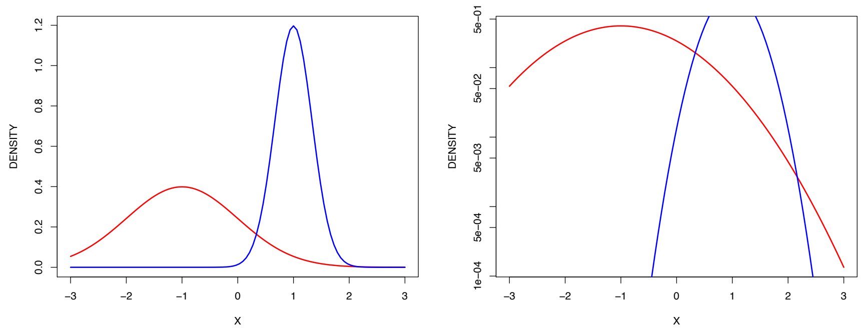
$$f_1(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-(-1))^2}{2}\right) \quad \text{and} \quad f_2(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-1)^2}{2}\right).$$



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

The Bayes Classifier

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 class 2 if and only if $x \in [-0.39, 2.15]$.

Plug-in Classification

- The Bayes Classifier chooses the class with the greatest posterior probability

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

- Unfortunately, we usually know neither the conditional class probabilities nor the prior probabilities.
- We can estimate the joint distribution with:
  - estimates $\hat{\pi}_k$ for π_k and $k = 1, \ldots, K$ and
  - estimates $\hat{f}_k(x)$ of conditional class densities,
- The **plug-in classifiers** chooses the class

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

- **Linear Discriminant Analysis** will be an example of plug-in classification.

Linear Discriminant Analysis

- LDA is the most well-known and simplest example of plug-in classification.
- Assume a multivariate Normal form for $f_k(x)$ for each class k:

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

  - each class can have a **different mean** μ_k
  - but all classes share the **same covariance** Σ.
- For an observation x,

$$\begin{aligned} \log \mathbb{P}(Y = k|X = x) &= \kappa + \log \pi_k f_k(x) \\ &= \kappa + \log \pi_k - \frac{1}{2}(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k) \end{aligned}$$

  The quantity $(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k)$ is the square of the **Mahalanobis distance**. It gives the distance between x and μ_k in the metric given by Σ.
- 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.

Linear Discriminant Analysis

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

$$\begin{aligned}\log \mathbb{P}(Y = k|x) &= \kappa + \log(\pi_k) - \frac{1}{2}\left(\mu_k^\top \Sigma^{-1}\mu_k - 2\mu_k^\top \Sigma^{-1}x + x^\top \Sigma^{-1}x\right)\\ &= \kappa + \log(\pi_k) - \frac{1}{2}\mu_k^\top \Sigma^{-1}\mu_k + \mu_k^\top \Sigma^{-1}x\end{aligned}$$

- 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

$$\log \mathbb{P}(Y = k|X = x) = \kappa + a_k + b_k^\top x$$

  i.e. a **linear** discriminant function.
- 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 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

- The final piece of the puzzle is 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.

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

  Then:

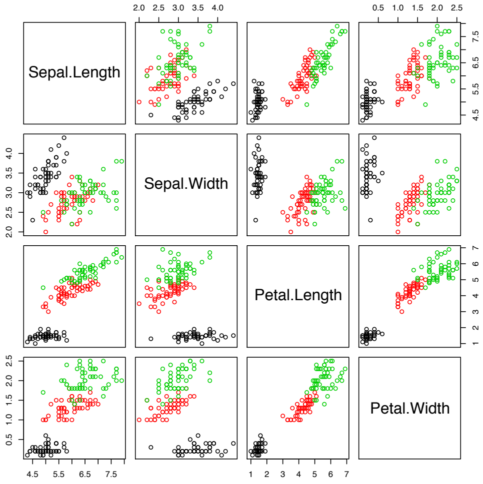
$$\hat{\pi}_k = \frac{n_k}{n} \qquad \hat{\mu}_k = \frac{1}{n_k}\sum_{j:y_j=k} x_j$$

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

- Note: the ML estimate of Σ is not unbiased. For an unbiased estimate we need to divide by $n - K$.

Iris Dataset

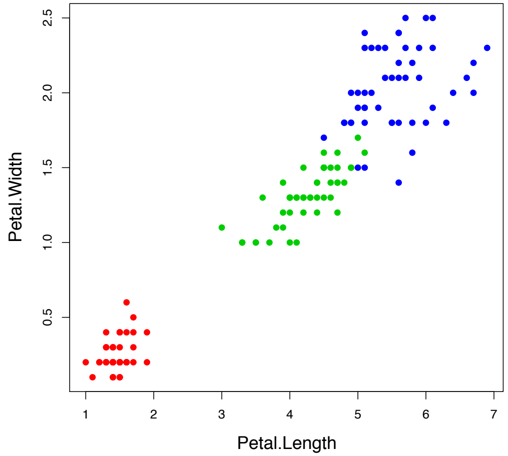
```
library(MASS)
data(iris)
##save class labels
ct <- rep(1:3,each=50)
##pairwise plot
pairs(iris[,1:4],col=ct)
```



Iris Dataset

Just focus on two predictor variables.

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

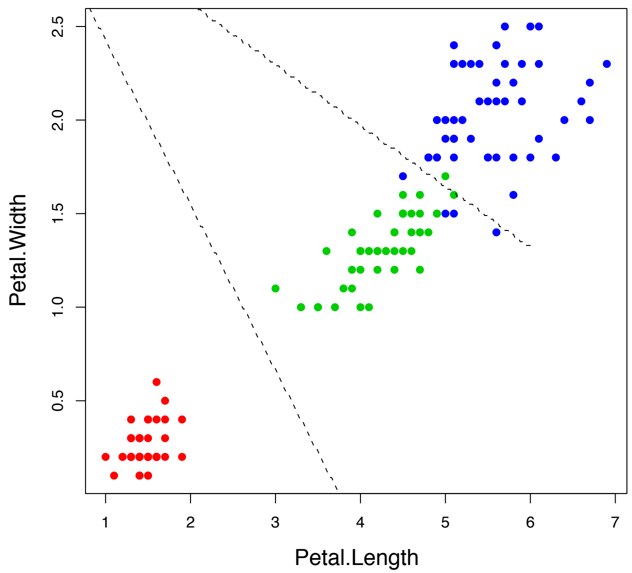


Iris Dataset

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(-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)
```


Iris Dataset

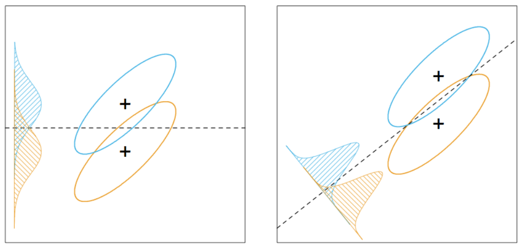


Fisher's Linear Discriminant Analysis

- In LDA, data vectors are classified based on Mahalanobis distance from cluster means, which lie on a $K-1$ affine subspace.
- In measuring these distances, directions orthogonal[5] to the subspace can be ignored.
- Projecting data vectors onto the subspace can be viewed as a dimensionality reduction technique that preserves discriminative information about $(y_i)_{i=1}^n$.
- As with PCA, we can visualize the structure in the data by choosing an appropriate basis for the subspace and projecting data onto it.
- Choose a basis by finding directions that are separate classes best.

[5] Orthogonality defined in terms of the inner product corresponding to Mahalanobis distance: $\langle x, y\rangle = x\Sigma^{-1}y$.

Fisher's Linear Discriminant Analysis



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

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

where

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

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

B has rank at most $K-1$.

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 defined 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

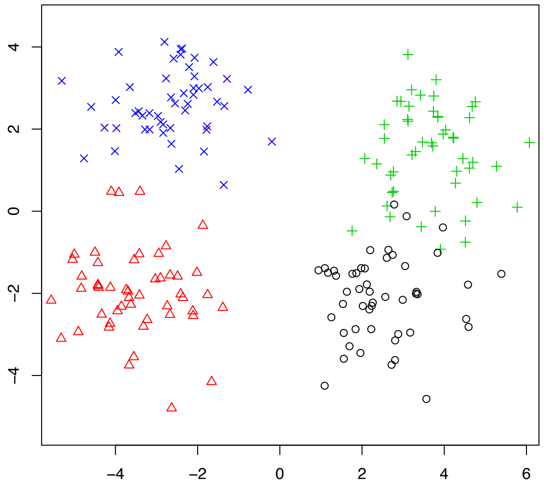
```
library(MASS)
data(crabs)

## 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))
```


Crabs Dataset

```
> > > > > > > > > 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
```


Crabs Dataset

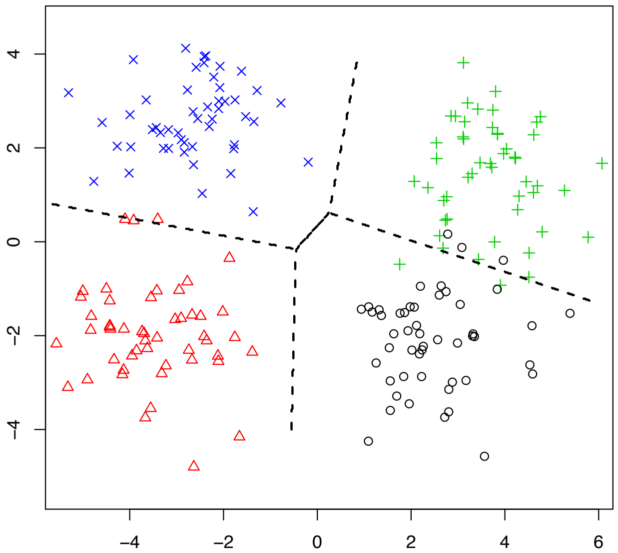
```
cb.ldp <- predict(cb.lda)
eqscplot(cb.ldp$x,pch=ct+1,col=ct+1)
```



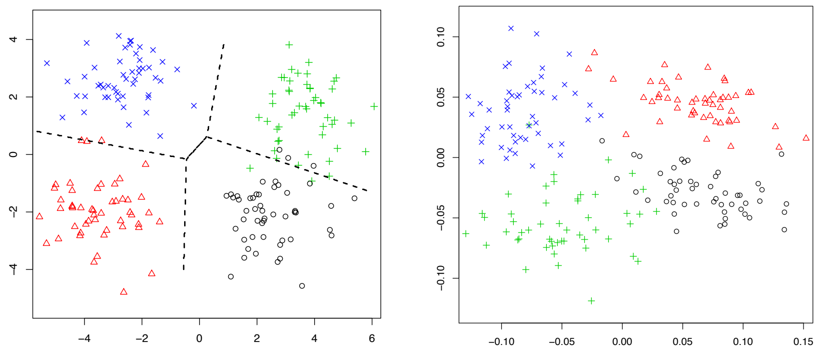
Crabs Dataset

```
## 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)
```


Crabs Dataset



Crabs Dataset



LDA separates the groups better.

Naïve Bayes

- Assume we are interested in classifying documents; e.g. scientific articles or emails.
- A basic but standard model for text classification consists of considering a pre-specified dictionary of p words (including say physics, calculus.... or dollars, sex etc.) and summarizing each document i by a binary vector x_i where

$$x_{ij} = \begin{cases} 1 & \text{if word } j \text{ is present in document} \\ 0 & \text{otherwise.} \end{cases}$$

- To implement a probabilistic classifier, we need to model $f_k(x|\phi_k)$ for each class $k = 1, ..., K$.

Naïve Bayes

- A Naïve Bayes approach ignores feature correlations and assumes $f_k(x) = f(x|\phi_k)$ where
$$f_k(x_i) = f(x_i|\phi_k) = \prod_{j=1}^{p} (\phi_{kj})^{x_{ij}} (1 - \phi_{kj})^{1-x_{ij}}$$
- Given dataset, the MLE is easily obtained
$$\hat{\pi}_k = \frac{n_k}{n} \qquad \hat{\phi}_{kj} = \frac{\sum_{i:y_i=k} x_{ij}}{n_k}$$
- One problem: if word j did not appear in documents labelled as class k then $\hat{\phi}_{kj} = 0$ and
$$\mathbb{P}(Y = k|X = x \text{ with } j\text{th entry equal to } 1) = 0$$
i.e. we will never attribute a new document containing word j to class k.
- This problem is called **overfitting**, and is a major concern in modelling high-dimensional datasets common in machine learning.