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

Supervised Learning

Unsupervised learning:

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

Supervised learning:

- In addition to the observations of X, we have access to their response variables / labels $Y \in \mathcal{Y}$: we observe $\{(x_i, y_i)\}_{i=1}^n$.
- Types of supervised learning:
  - Classification: discrete responses, e.g. $\mathcal{Y} = \{+1, -1\}$ or $\{1, \ldots, K\}$.
  - Regression: a numerical value is observed and $\mathcal{Y} = \mathbb{R}$.

The goal is to accurately predict the response Y on new observations of X, i.e., to **learn a function** $f : \mathbb{R}^p \to \mathcal{Y}$, such that $f(X)$ will be close to the true response Y.

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:
  - **Misclassification loss** (or **0-1 loss**) for classification

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

  - **Squared loss** for regression

$$L(Y, f(X)) = (f(X) - Y)^2 .$$

- 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.

Risk

- paired observations $\{(x_i, y_i)\}_{i=1}^n$ viewed as i.i.d. realizations of a random variable (X, Y) on $\mathcal{X} \times \mathcal{Y}$ with joint distribution P_{XY}

Risk

For a given loss function L, the **risk** R of a learned function f is given by the expected loss

$$R(f) = \mathbb{E}_{P_{XY}}\left[L(Y, f(X))\right],$$

where the expectation is with respect to the true (unknown) joint distribution of (X, Y).

- The risk is unknown, but we can compute the **empirical risk**:

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)).$$

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:

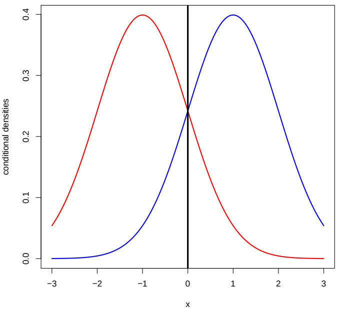
$$\begin{aligned}f_{\text{Bayes}}(x) &= \arg\max_{k=1,\dots,K} \mathbb{P}(Y=k|X=x)\\ &= \arg\max_{k=1,\dots,K} \frac{\pi_k g_k(x)}{\sum_{j=1}^{K}\pi_j g_j(x)} = \arg\max_{k=1,\dots,K}\ \pi_k g_k(x).\end{aligned}$$

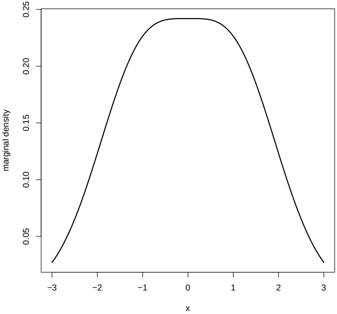
- The functions $x \mapsto \pi_k g_k(x)$ are called **discriminant functions**. The discriminant function with maximum value determines the predicted class of x.

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 $\pi_1 = \pi_2 = 1/2$.

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

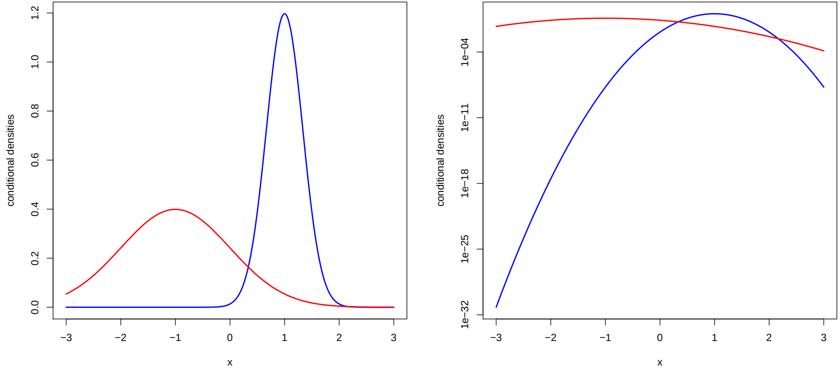




Optimal classification is $f_{\text{Bayes}}(x) = \underset{k=1,\ldots,K}{\arg\max}\ \pi_k g_k(x) = \begin{cases} 1 & \text{if } x < 0, \\ 2 & \text{if } x \geq 0. \end{cases}$

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:

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

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

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

- 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

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

  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:

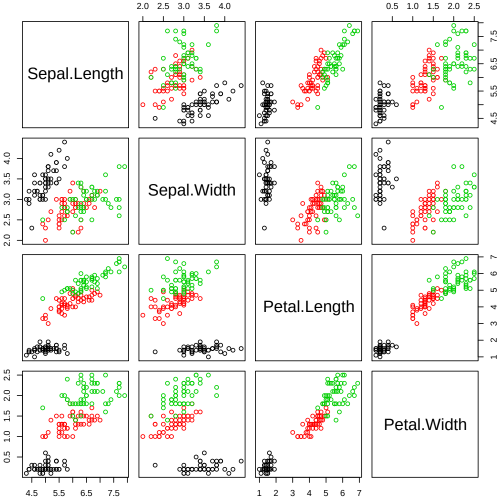
$$\hat{\pi}_k = \frac{n_k}{n} \qquad \hat{\mu}_k = \frac{1}{n_k}\sum_{j:y_j=k} x_j$$

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

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

Iris Dataset

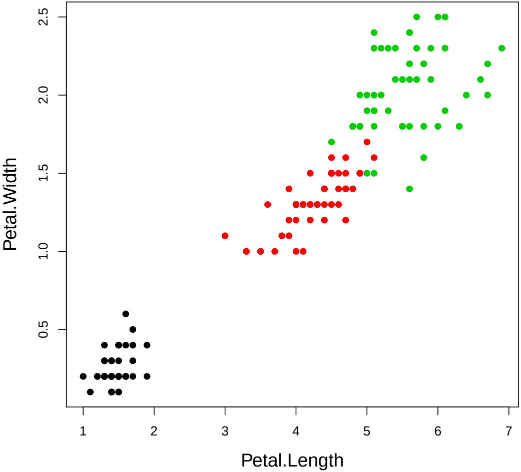
```
library(MASS)
data(iris)
##save class labels
ct <- unclass(iris$Species)
##pairwise plot
pairs(iris[,1:4],col=ct)
```



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

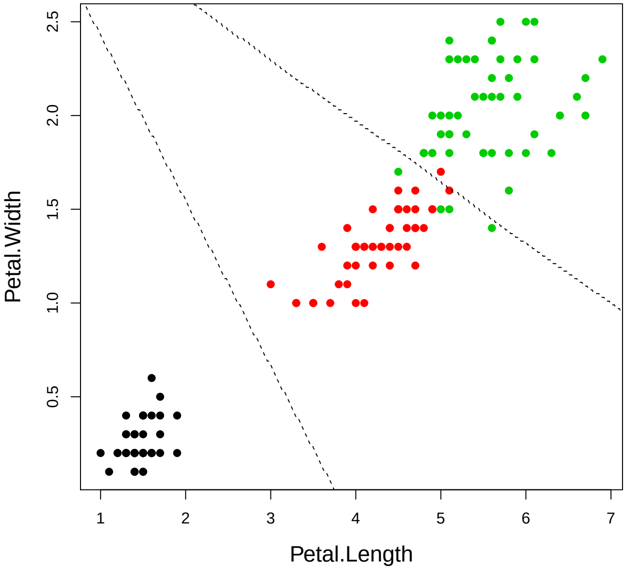
Computing and plotting the LDA boundaries.

```
##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)

##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


Petal.Width
Petal.Length