Module 4: Linear Classification

TMA4268 Statistical Learning V2020

Stefanie Muff, Department of Mathematical Sciences, NTNU

February xx, 2020

Introduction

Learning material for this module

- James et al (2013): An Introduction to Statistical Learning. Chapter 4.

We need more statistical theory than is presented in the textbook, which you find in this module page.
Some of the figures and slides in this presentation are taken (or are inspired) from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

Topics in this module

Part A: Introduction to classification, and modelling class densities

- Aim of this module
- What is classification and what is discrimination?
- Loss function and the Bayes classifier
- Modelling class densities
    - Linear discriminant analysis
    - Quadratic discriminant analysis
    - Naive Bayes (optional)
- Modelling posterior probabilities
    - KNN-classifier

Part B: Modelling posterior probabilites, ROC/AUC and comparisions

- Modelling posterior probabilities (cont.)
    - Linear regression for classification problems
    - Logistic regression
- Sensitivity, specificity, ROC and AUC
- Comparisons
- Extensions

What is classification?

- By now our responses Y was assumed *continuous*, while covariates were allowed to be *categorical*.
- Now we allow the response to be *categorical*.
- This is even more common than continuous responses. Examples:
    - Spam filters **`email`** $\in$ {spam, ham},
    - **`Eye color`** $\in$ {blue, brown, green}.
    - **`Medical condition`** $\in$ {disease1, disease2, disease3}.

- Suppose we have a qualititative response value that can be a member in one of K classes $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$.
- In classification we build a function $f(X)$ that takes a vector of input variables X and predicts its class membership, such that $Y \in \mathcal{C}$.
- We would also assess the *uncertainty* in this classification. Sometimes the role of the different predictors may be of main interest.
- We often build models that **predict probabilities of categories**, *given* certain covariates X.

Three methods for classification

- K-nearest neighbours / Bayes classification
- Logistic regression
- Linear and quadratic discriminant analysis

Example: Which type of iris species?

The `iris` flower data set was introduced by the British statistician and biologist Ronald Fisher in 1936.

- Three plant species {setosa, virginica, versicolor} (50 observation of each), and
- four features: `Sepal.Length`, `Sepal.Width`, `Petal.Length` and `Petal.Width`.

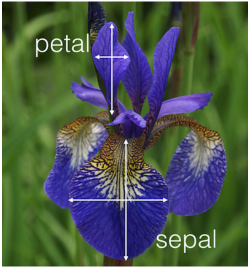


Figure 1: Iris plant with sepal and petal leaves

<http://blog.kaggle.com/2015/04/22/scikit-learn-video-3-machine-learning-first-steps-with-the-iris-dataset/>

Set-up: Training observations $\{(x_1, y_1), ..., (x_n, y_n)\}$ where the response variable Y is categorical, e.g $Y \in \mathcal{C} = \{0, 1, ..., 9\}$ or $Y \in \mathcal{C} = \{dog, cat, ..., horse\}$.

Aim: To *build* a classifier $f(x)$ that assigns a class label from $\mathcal{C}$ to a future unlabelled observation x and to asses the *uncertainty* in this classification.

Performance measure: Most popular is the misclassification error rate (training and test version)..

Bayes classifier

- The *Bayes classifier assigns an observation to the most likely class*, given its predictor values.
- Suppose we have a quantitative response value that can be a member in one of K classes $\mathcal{C} = \{c_1, c_2, ..., c_k, ..., c_K\}$. Further, suppose these elements are numbered $1, 2, ..., K$. The probability of that a new observation x_0 belongs to class k is

$$p_k(x_0) = \Pr(Y = k|X = x_0), \quad k = 1, 2, ...K.$$

  This is the conditional class probability: the probability that $Y = k$ given the observation x_0.
- Two-class example for two groups $\{A, B\}$. A new observation x_0 will be classified to A if $\Pr(Y = A|X = x_0) > 0.5$ and to class B otherwise.

Given a continuous random vector X and categorical random variable Y, how do we get $\Pr(Y = k|X = x_0)$?

Bayes theorem

$$\begin{aligned} P(Y = k \mid X = x) &= \frac{P(\mathbf{X} = \mathbf{x} \cap Y = k)}{f(\mathbf{x})} \\ &= \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l} \end{aligned}$$

Here $f_k(x) = P(X = x \mid Y = k)$ is the pdf for X in class k and $\pi_k = P(Y = k)$ is the prior probability for class k.

Properties of the Bayes classifier

- It has the *smallest test error rate*.
- The class boundaries using the Bayes classifier is called the *Bayes decision boundary*.
- The overall Bayes error rate is given as

  $$1 - \mathrm{E}(\max \Pr(Y = j \mid X))$$

  where the expectation is over X.
- The Bayes error rate is comparable to the *irreducible error* in the regression setting.
- Caveat: we never (or very seldom) know the conditional distribution of Y given X for real data

→ The K-nearest neighbor classifier estimates this conditional distribution and then classifies a new observation based on this estimated probability.

K-nearest neighbour classifier

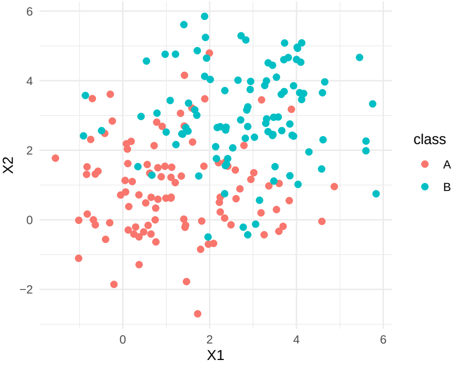
The K-nearest neighbour classifier (KNN) works in the following way:

- Given a new observation x_0 it searches for the K points in our training data that are closest to it (Euclidean distance).
- These points make up the neighborhood of x_0, $\mathcal{N}_0$.
- The point x_0 is classified by taking a majority vote of the neighbors.
- That means that x_0 is classified to the most occurring class among its neighbors

$$\Pr(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j).$$

A synthetic example

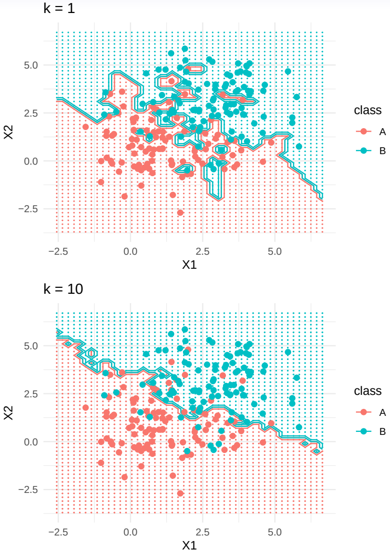
- Simulate 2×100 observations from a bivariate normal distribution with mean vectors $\mu_A = (1,1)^T$, $\mu_B = (3,3)^T$, and covariance matrix $\Sigma_A = \Sigma_B = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$.
- Aim: Find a rule to classify a new observation to A or B.



- Assume we have a new observation $X_0 = (x_{01}, x_{02})^T$ which we want to classify as belonging to the class A or B.
- To illustrate this problem we fit the K-nearest neighbor classifier to our simulated data set with $K = 1, 3, 10$ and 150 (next slide).

Interpretation:

- The small colored dots show the predicted classes for an evenly-spaced grid.
- The lines show the decision boundaries.
- If our new observation falls into the region within the red decision boundary, it will be classified as A. If it falls into the region within the green decision boundary, it will be classified as B.



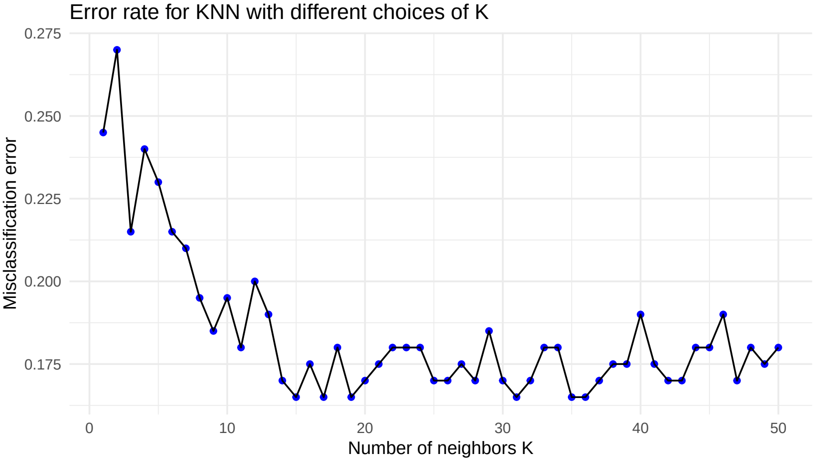
k = 1

k = 3

k = 10

k = 150

- The choice of K has a big influence on the result of our classification. For $K = 1$ the classification is made to the same class as the one nearest neighbor. As K gets very large, the decision boundary tends towards a straight line (which is the Bayes boundary in this set-up).
- To find the optimal value of K the typical procedure is to try different values of K and then test the predictive power of the different classifiers, for example by cross-validation, which will be discussed in M5.
- After trying all choices for K between 1 and 50, a few choices of K gave the smallest misclassification error rate, estimating by leave-one out cross-validation (see M5). The smallest error rate is equal to 0.165 (16.5% misclassification).



Error rate for KNN with different choices of K

Bias-variance trade-off in a classification setting

- A too low value of K will give a very flexible classifier (with high variance and low bias) which will fit the training set too well (it will overfit) and make poor predictions for new observations.
- Choosing a high value for K makes the classifier loose its flexibility and the classifier will have low variance but high bias.

$\rightarrow$ Critical to the success of the classifier to choose the correct level of flexibility (K).

The curse of dimensionality

- The nearest neighbor classifier can be quite good if the number of predictor p is small and the number of observations n is large. We need enough close neighbors to make a good classification.
- The effectiveness of the KNN classifier falls quickly when the dimension of the preditor space is high.
- Why? Because the nearest neighbors tend to be far away in high dimensions and the method no longer is local. This is referred to as the *curse of dimensionality*.

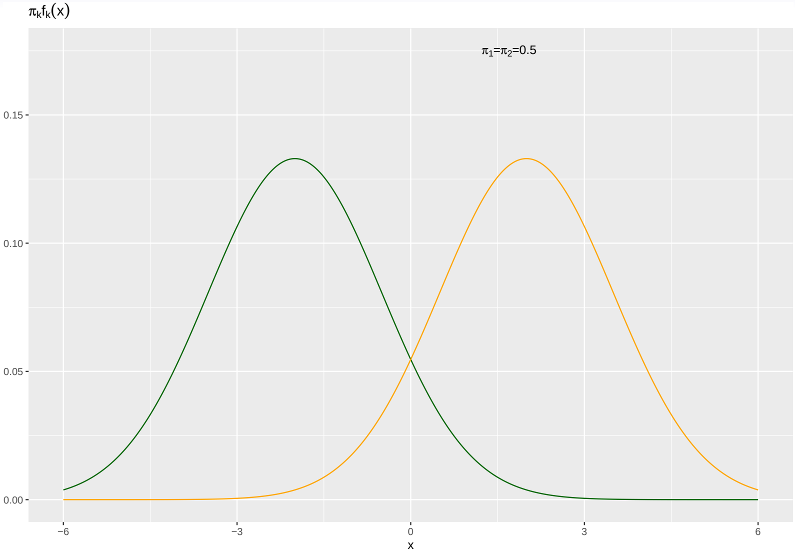
Synthetic example: what is the Bayes error?

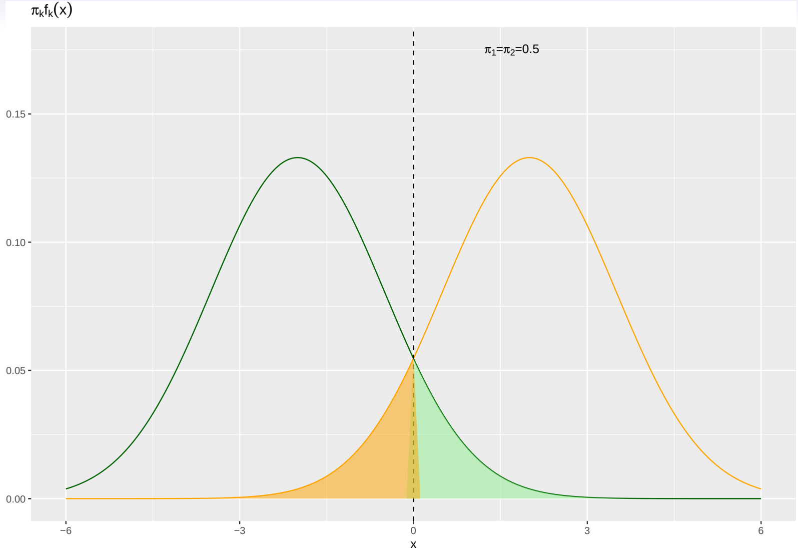
Suppose we have observations coming from two classes: {green, orange}, where

$$X_{\text{green}} \sim \mathcal{N}(-2, 1.5^2) \text{ and } X_{\text{orange}} \sim \mathcal{N}(2, 1.5^2)$$

and that the probability of observing each class is equal.

- Where is the Bayes decision boundary here?
- How can we calculate the Bayes error rate (intuitively- see the graph)?
- What would you estimate the Bayes error rate to be?
- What if someone constructs a classifier and it has a lower error rate than the Bayes error rate - is this possible?


$\pi_k f_k(x)$
$\pi_1=\pi_2=0.5$
0.15
0.10
0.05
0.00
−6
−3
0
3
6
x



Bayes error: `round(0.5*2*pnorm(0,mean=2,sd=1.5),2)`$=0.09$

Bayes decision rule - two paradigms

The most popular approach to classication is to use the Bayes decision rule with the 0/1 loss function= classify to the class with the largest $P(Y = k \mid X = x_0)$.
Two approaches:

- The **diagnostic paradigm**: We focus on *directly* estimating the posterior distribution for the classes $P(Y = k \mid X = x)$.
- The **sampling paradigm**: There focus is on estimating the prior probabilities π_k for the classes and the class conditional distributions $f_k(x)$. We classify to the class with the maximal product $\pi_k f_k(x)$.

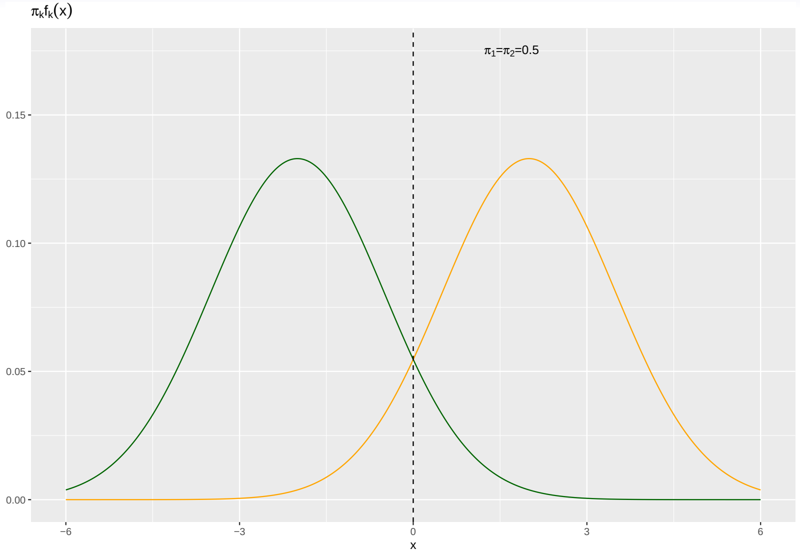
We *first* look at the sampling paradigm - and then we need to model the pdf for each class. Popular: the multivariate normal distribution!

Univariate normal class distributions - the role of the prior

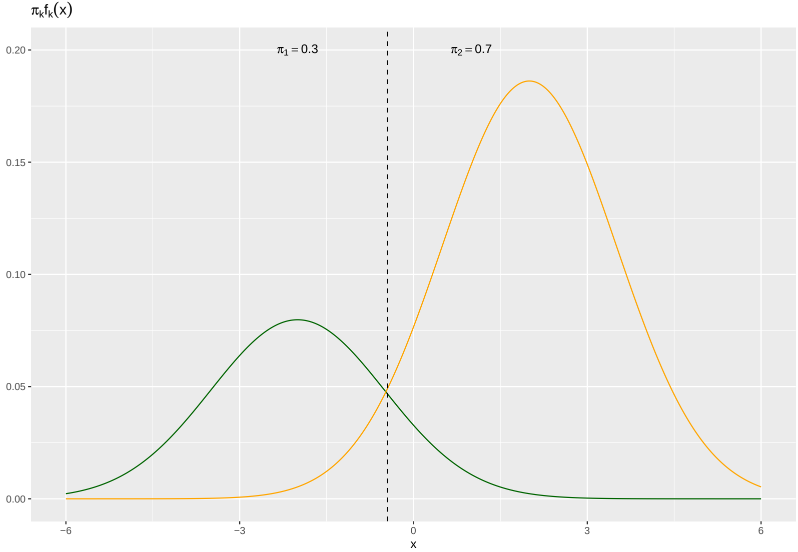
Suppose (again) we have observations coming from two classes: {green, orange}, where

$$X_{\text{green}} \sim \mathcal{N}(-2, 1.5^2) \text{ and } X_{\text{orange}} \sim \mathcal{N}(2, 1.5^2)$$

In the figure below we have specified the prior probabilities to be equal, $\pi_1 = \pi_2 = 0.5$ and have plotted $\pi_k f_k(x)$ for the two classes. The decision boundary is where the point of intersection of the two lines is, because here $\pi_1 f_1(x) = \pi_2 f_2(x)$. Thus all points to left of the decision boundary will be classified as green and similarly, all points to the right of the decision boundary will be classified as orange.


$\pi_k f_k(x)$
π_1=π_2=0.5
0.15
0.10
0.05
0.00
−6
−3
0
3
6
x

We now specify different priors, such that $\pi_1 = 0.3$ and $\pi_2 = 0.7$. We see that this has affected the decision boundary, which now has shifted to the left.



Linear and quadratic discriminant analysis

Linear discriminant analysis (LDA)

Using Linear discriminant analysis we assume the class conditional distributions are normal (Gaussian).

Univariate (p=1)

The univariate normal pdf

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2},$$

has parameters μ (mean) and σ (standard deviation).
Assume that we observe K classes, where each class conditional distribution is normal, where the classes have mean μ_k and standard deviation σ_k:

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma_k}\right)^2}$$

With LDA we assume that all of the classes have the *same standard deviation* $\sigma_k = \sigma$.
In addition we have prior class probabilites $\pi_k = P(Y = k)$, so that $\sum_{k=1}^{K} \pi_k = 1$.

We can insert the expression for each class distribution into Bayes formula to obtain the posterior probability $p_k(x) = P(Y = k|X = x)$:

$$p_k(x) = \frac{f_k(\mathbf{x})\pi_k}{f(\mathbf{x})} = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma}\right)^2}}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu_l}{\sigma}\right)^2}}$$

Our rule is to classify to the class for which $p_k(x)$ is largest.

It can be shown that this is equivalent to assigning x to the class with the largest *discriminant score* $\delta_k(x)$:

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k).$$

This decision boundaries between the classes are *linear* in x.

Q: So, what do we need to use this result in practice?

Parameter estimators

In real life situations we will not know the prior probabilities, the class mean or standard deviation=need parameter estimators.

- Prior probability for class k is (often) estimated by taking the fraction of observations n_k (out of n) coming from class k: $\hat{\pi}_k = \frac{n_k}{n}$.
- The mean value for class k is simply the sample mean of all observations from class k:

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i.$$

- The standard deviation: sample standard deviation across all classes:

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2 = \sum_{k=1}^{K} \frac{n_k - 1}{n-K} \cdot \hat{\sigma}_k^2.$$

$\hat{\sigma}_k$: estimated standard deviation of all observations from class k.

How would the estimation affect our misclassification rate?

If $\mu_1 = -2$, $\mu_2 = 2$, $\sigma = 1.5$, $\pi_1 = \pi_2 = 0.5$ we found that the class boundary is at 0 and the Bayes error is `round(2*0.5*pnorm(0,2,1.5))`$=0.09$.
But, in a real life situation

- we estimate the class boundary
- and we do not know the true distribution for the classes.

How can we then estimate the goodness of our estimator?

1. Use the training set to estimate parameters and class boundary:
2. Use the test set to estimate misclassification rate.

```r
n = 1000
pi1 = pi2 = 0.5
mu1 = -2
mu2 = 2
sigma = 1.5
set.seed(1)
n1train = rbinom(1, n, pi1)
n2train = n - n1train
n1test = rbinom(1, n, pi1)
n2test = n - n1test
train1 = rnorm(n1train, mu1, sigma)
train2 = rnorm(n2train, mu2, sigma)
test1 = rnorm(n1test, mu1, sigma)
test2 = rnorm(n2test, mu2, sigma)
sigma2.1 = var(train1)
sigma2.2 = var(train2)
estsigma2 = ((n1train - 1) * sigma2.1 + (n2train - 1) * sigma2.2)/(n -
    2)

rule = 0.5 * (mean(train1) + mean(train2)) + estsigma2 * (log(n2train/n) -
    log(n1train/n))/(mean(train1) - mean(train2))

c((sum(train1 > rule) + sum(train2 < rule))/n, (sum(test1 > rule) + sum(test2 <
    rule))/n)
```

Training error rate

the proportion of mistakes that are made if we apply classifier $\hat{f}$ to the training observations, i.e. $\hat{y}_i = \hat{f}(x_i)$.

$$\frac{1}{n}\sum_{i=1}^{n} \mathrm{I}(y_i \neq \hat{y}_i).$$

Here I is the indicator function (to give our 0/1 loss) which is defined as:

$$\mathrm{I}(a \neq \hat{a}) = \begin{cases} 1 \text{ if } a \neq \hat{a} \\ 0 \text{ else} \end{cases}$$

The indicator function counts the number of times our model has made a wrong classification. The training error rate is the fraction of misclassifications made on our training set.
A very low training error rate may imply overfitting.

Test error rate

Here the fraction of misclassifications is calculated when our model is applied on a test set. From what we have learned about regression we can deduct that this gives a better indication of the true performance of the classifier (than the training error).

$$\text{Ave}(I(y_0 \neq \hat{y}_0))$$

where the average is over all the test observations (x_0, y_0).
We assume that a *good* classifier is a classifier that has a *low* test error.

The confusion matrix

The confusion matrix is a table that can show the performance of classifier, given that the true values are known.

We can make a confusion matrix from the training or test set, but will in most cases do so for the test set only.

The rows represent the true classes, while the columns represent the predicted classes. Inside the table we have counts (just labelled "correct" and "wrong" below - but should be numbers). The sum of the diagonal is the total number of correct classifications. The sum of all elements off the diagonal is the total number of misclassifications.

Predicted 1

Predicted 2

...

Predicted K

True 1

correct

wrong

...

wrong

True 2

wrong

correct

The confusion matrix can be obtained in `R` by using the `table` function with the predicted classes and the true classes need to be given as function arguments, or directly using the `caret` package.

We will soon come back to the special case of two classes - where we may think of this as “-” (non disease) and “+” (disease).

Multivariate LDA (p>1)

Linear discriminant analysis can be generalized to situations when $p > 1$ covariates are used. The decision boundaries are still linear. The multivariate normal distribution function:

$$f(x) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))$$

This gives the following expression for the discriminant function:

$$\delta_k(x) = \mathbf{x}^T \mathbf{\Sigma}^{-1} \boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T \mathbf{\Sigma}^{-1} \boldsymbol{\mu}_k + \log \pi_k.$$

Some details on the derivation of the discriminant function - when $K = 2$ classes, and the classes are denoted 0 and 1

$$P(Y = 0|X = \mathbf{x}) = P(Y = 1|X = \mathbf{x})$$

$$\frac{\pi_0 f_0(x)}{\pi_0 f_0(x) + \pi_1 f_1(x)} = \frac{\pi_1 f_0(1)}{\pi_0 f_0(x) + \pi_1 f_1(x)}$$

$$\pi_0 f_0(\mathbf{x}) = \pi_1 f_1(\mathbf{x})$$

$$\pi_0 \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} e^{\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_0)} = \pi_1 \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} e^{\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)}$$

$$log(\pi_0) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_0) = log(\pi_1) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_1)$$

$$log(\pi_0) - \frac{1}{2} x^T \boldsymbol{\Sigma}^{-1}\mathbf{x} + \mathbf{x}^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 - \frac{1}{2}\boldsymbol{\mu}_0^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 = log(\pi_1) - \frac{1}{2}\mathbf{x}^T \boldsymbol{\Sigma}^{-1}\mathbf{x} + \mathbf{x}^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$$

$$log(\pi_0) + \mathbf{x}^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 - \frac{1}{2}\boldsymbol{\mu}_0^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 = log(\pi_1) + \mathbf{x}^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \frac{1}{2}\mu_1^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1$$

$$\delta_0(\mathbf{x}) = \delta_1(\mathbf{x})$$

Estimators for p>1:

- Prior probability for class k (unchanged from p=1): $\hat{\pi}_k = \frac{n_k}{n}$.
- The mean value for class k is simply the sample mean of all observations from class k (but now these are vectors):

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i:y_i=k} \mathbf{X}_i.$$

- The covariance matrices for each class:

$$\hat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k - 1} \sum_{i:y_i=k} (\mathbf{X}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{X}_i - \hat{\boldsymbol{\mu}}_k)^T$$

- Pooled version:

$$\hat{\boldsymbol{\Sigma}} = \sum_{k=1}^{K} \frac{n_k - 1}{n - K} \cdot \hat{\boldsymbol{\Sigma}}_k.$$

Optional: Proof that the estimator $\hat{\boldsymbol{\Sigma}}_k$ is unbiased for each class (from TMA4267). Proof for pooled version not provided.

Posterior probabilites

Sometimes the probability that an observation comes from a class k is more interesting than the actual classification itself. These class probabilities can be estimated from the priors and class conditional distributions, or from the discriminant functions:

$$\begin{aligned}\hat{P}(Y=k|X=x) &= \frac{\hat{\pi}_k \cdot \frac{1}{(2\pi)^{p/2}|\hat{\boldsymbol{\Sigma}}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}-\hat{\boldsymbol{\mu}}_k)^T \hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}-\hat{\boldsymbol{\mu}}_k))}{\sum_{l=1}^K \hat{\pi}_l \frac{1}{(2\pi)^{p/2}|\hat{\boldsymbol{\Sigma}}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}-\hat{\boldsymbol{\mu}}_l)^T \hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}-\hat{\boldsymbol{\mu}}_l))} \\ &= \frac{e^{\hat{\delta}_k(x)}}{\sum_{l=1}^K e^{\hat{\delta}_l(x)}}.\end{aligned}$$

Quadratic Discriminant Analysis (QDA)

In quadratic discriminant analysis we assume that the distributions of the classes is multivariate normal (Gaussian), but with covariance matrix $\mathbf{\Sigma}_k$ for each class.

The discriminant functions are now given by:

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\mathbf{\Sigma}_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\mathbf{\Sigma}_k| + \log\pi_k \\ &= -\frac{1}{2}x^T\mathbf{\Sigma}_k^{-1}x + x^T\mathbf{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\mathbf{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\log|\mathbf{\Sigma}_k| + \log\pi_k.\end{aligned}$$

These decision boundaries are *quadratic* functions of x.

LDA vs QDA

In QDA we are allowed for the covariance matrices to be different for the classes, but for LDA they are assumed to be the same, so QDA is more flexible than LDA.

Q:

- But, if the covariance matrices in theory are equal - will they not be estimated equal?
- Should we not always prefer QDA to LDA?

A:

Explanation similar to a “Bias-variance trade-off”:

If the assumption of equal covariance matrices is wrong

- then LDA may suffer from high bias for the parameter estimators.
- and QDA is better off. But, for small sample sizes the covariance matrices might be poorly estimated (high variance of estimators).

If the number of covariates is high:

- then QDA requires estimating $K \cdot p \cdot (p+1)/2$ parameters,
- while LDA only requires $p \cdot (p+1)/2$.

Therefore, LDA is less flexible than QDA and might therefore have much less variance.

Naive Bayes (optional)

This is method that is popular when p is large.
In Naive Bayes (Idiot's Bayes) we assume that each class density is the product of marginal densities - i.e. inputs are conditionally independent in each class.

$$f_k(x) = \prod_{j=1}^{p} f_{kj}(x_j)$$

This is generally not true, but it simplifies the estimation dramatically.
The original naive Bayes used univariate normal marginal distributions, but generalizations can be made.
Instead of estimating a full covariance matrix, only the diagonal elements are estimated.
This method often produces good results, even though the joint pdf is not the product of the marginal pdf. This might be because we are not focussing on estimation of class pdfs, but class boundaries.

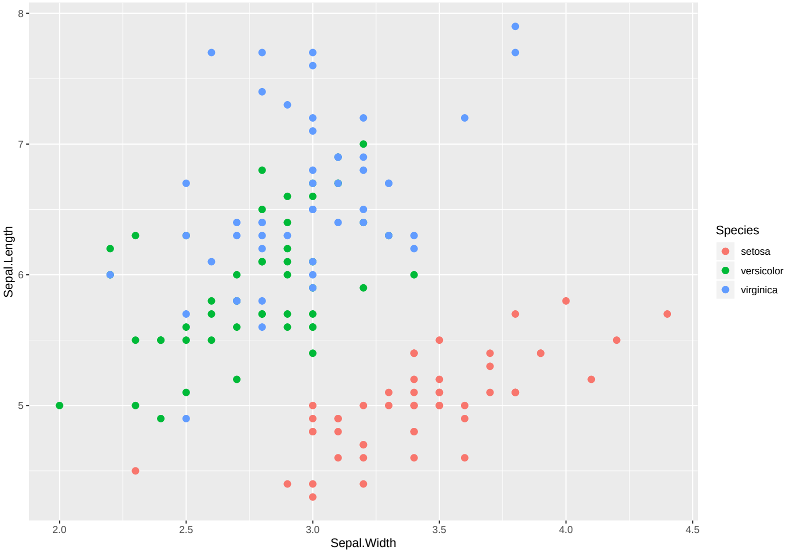
Example: Classification of iris plants

We will use `sepal width` and `sepal length` to build a classificator, here 50 observations from each class available.

```
attach(iris)
library(class)
library(MASS)
library(ggplot2)
library(dplyr)
kable(head(iris), format = whichformat)
```

Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Species
5.1
3.5
1.4
0.2
setosa
4.9

```
iris0_plot = ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length,
    geom_point(size = 2.5)
iris0_plot
```



Iris: LDA

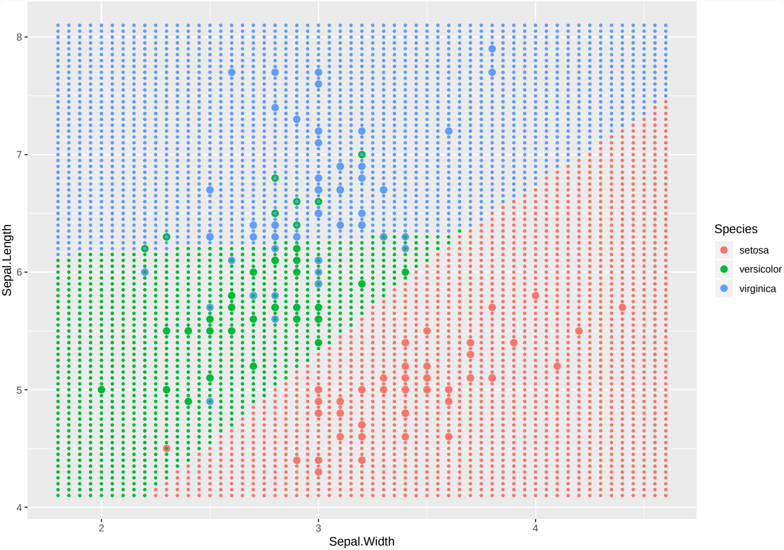
```
# reporting on parameter estimates

grS = cbind(iris[iris$Species == "setosa", 1:2])
grVe = cbind(iris[iris$Species == "versicolor", 1:2])
grVi = cbind(iris[iris$Species == "virginica", 1:2])
nk = dim(grS)[1]
apply(grS, 2, mean)
```

```
## Sepal.Length  Sepal.Width
##        5.006        3.428
```

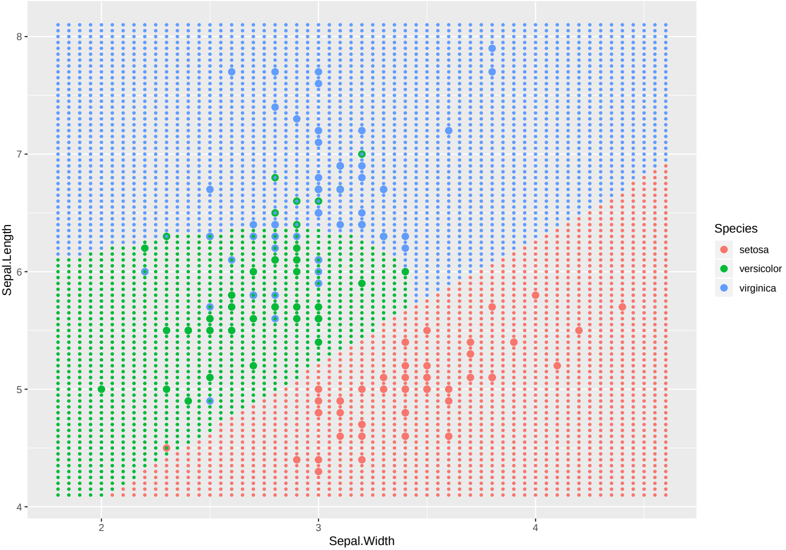
```
cS = cov(grS)
apply(grVe, 2, mean)
```

```
## Sepal.Length  Sepal.Width
##        5.936        2.770
```

```
cVe = cov(grVe)
apply(grVi, 2, mean)
```


Sepal.Length
8
7
6
5
4
2
3
4
Sepal.Width
Species
setosa
versicolor
virginica

Iris: QDA



8
7
6
5
4
Sepal.Length
2
3
4
Sepal.Width
Species
setosa
versicolor
virginica

Iris: compare LDA and QDA

We now want to compare the predictive performance of our two classifiers. We do this by dividing the original `iris` data set, randomly, into train and test samples of equal size:

```r
set.seed(1)
train = sample(1:150, 75)

iris_train = iris[train, ]
iris_test = iris[-train, ]

iris_lda2 = lda(Species ~ Sepal.Length + Sepal.Width, data = iri
    prior = c(1, 1, 1)/3)
```

```
# Training error
table(predict(iris_lda2, newdata = iris_train)$class, iris_train
```

```
##
##              setosa versicolor virginica
##   setosa         27          0         0
##   versicolor      1         15         8
##   virginica       0          5        19
```

```
# Test error
iris_lda2_predict = predict(iris_lda2, newdata = iris_test)
table(iris_lda2_predict$class, iris$Species[-train])
```

```
##
##              setosa versicolor virginica
##   setosa         22          0         0
##   versicolor      0         22        11
##   virginica       0          8        12
```

The LDA classifier has a training error rate of 15/75, that is 20 %. When tested on a new set it correctly classified 24+13+23 times and misclassified 15 times. This gives a misclassification rate of

$$\text{Test error}_{\text{LDA}} = \frac{15}{75} = 0.2.$$

Using a different division into training and test set will give (small) changes to these numbers.

Iris: training and test error for QDA

```
iris_qda2 = qda(Species ~ Sepal.Length + Sepal.Width, data = iri
    prior = c(1, 1, 1)/3)
```

Important: use the same division into training and test set for methods we want to compare.

```r
# Training error
table(predict(iris_qda2, newdata = iris_train)$class, iris_train
```

```
##
##               setosa versicolor virginica
##   setosa          28          0         0
##   versicolor       0         16         9
##   virginica        0          4        18
```

```r
# Test error
iris_qda2_predict = predict(iris_qda2, newdata = iris_test)
table(iris_qda2_predict$class, iris$Species[-train])
```

```
##
##               setosa versicolor virginica
##   setosa          22          0         0
##   versicolor       0         18        12
##   virginica        0         12        11
```

The QDA classifier has a training error rate of 16%. When tested on a new set, the misclassification error rate was

$$\text{Test error}_{\text{QDA}} = \frac{18}{75} = .24$$

The LDA classifier has given the smallest test error for classifying iris plants based on sepal width and sepal length for our test set and should be preferred in this case.

1. Would another division of the data into training and test set give the same conclusion (that LDA is better than QDA for this data set)? (A: Not necessarily, but probably.)

We will look into other choice than dividing into one training and one test set in Module 5 (crossvalidation).

2. What about the other two covariates? Would adding them to the model (4 covariates) give a better classification rule? (A: Probably. Try if you want.)

Fishers idea (Optional)

In 1936 R. A. Fisher developed LDA.

- His aim was to find a linear combination of the explanatory variables which *maximized the ratio of its between class to within class variance.*
- In this way the observations are transformed so that they are separated as much as possible.
- His approach has the advantage that it is suited for visual inspection and graphical description , it “separates” the populations.

Let the between-class variance be denoted $\boldsymbol{B} = \sum_{m=1}^{M} (\boldsymbol{\mu}_m - \bar{\boldsymbol{\mu}})(\boldsymbol{\mu}_m - \bar{\boldsymbol{\mu}})^T$, where $\boldsymbol{\mu}_m$ denotes the mean of class ω_m and $\bar{\boldsymbol{\mu}}$ the overall mean.
The within-class variance is assumed equal for all classes and is denoted $\boldsymbol{\Sigma}$ ($\boldsymbol{\Sigma}$ is assumed to have full rank).

The linear combination $\boldsymbol{l}^T\boldsymbol{X}$ that maximize $\boldsymbol{l}^T\boldsymbol{B}\boldsymbol{l}/\boldsymbol{l}^T\boldsymbol{\Sigma}\boldsymbol{l}$ under the constraint that $\boldsymbol{l}^T\boldsymbol{\Sigma}\boldsymbol{l} = 1$ is found to be the scaled eigenvectors of $\boldsymbol{\Sigma}^{-1}\boldsymbol{B}$ corresponding to the nonzero eigenvalues of $\boldsymbol{\Sigma}^{-1}\boldsymbol{B}$. The eigenvector corresponding to the largest eigenvalue defines the first discriminant $\boldsymbol{l}_1^T\boldsymbol{X}$. The second linear discriminant $\boldsymbol{l}_2^T\boldsymbol{X}$ is constructed from the eigenvector corresponding to the second largest eigenvalue and so on. (We also have $Cov(\boldsymbol{l}_j^T\boldsymbol{X}, \boldsymbol{l}_i^T\boldsymbol{X}) = 0$ for $i \neq j$ and $Var(\boldsymbol{l}_j^T\boldsymbol{X}) = 1$.)

The number of linear discriminants equals the number of nonzero eigenvalues. Observations are assigned to the class of the nearest (Euclidean distance) class mean in the discriminant space.

This equals classification to the nearest Mahalanobis distance population mean in the input space. Again, the parameters $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}$ and the between-class variance $\boldsymbol{B}$ are usually unavailable. Replacing the parameters by estimates from the training set leads to Fisher's sample linear discriminants.

Diagnostic paradigm

Remember:

- The **diagnostic paradigm**: We focus on *directly* estimating the posterior distribution for the classes $P(Y = k \mid X = x)$.
- The **sampling paradigm**: There focus is on estimating the prior probabilities for the classes and the class conditional distributions. We classify to the class with the maximal product $\pi_k f_k(x)$.

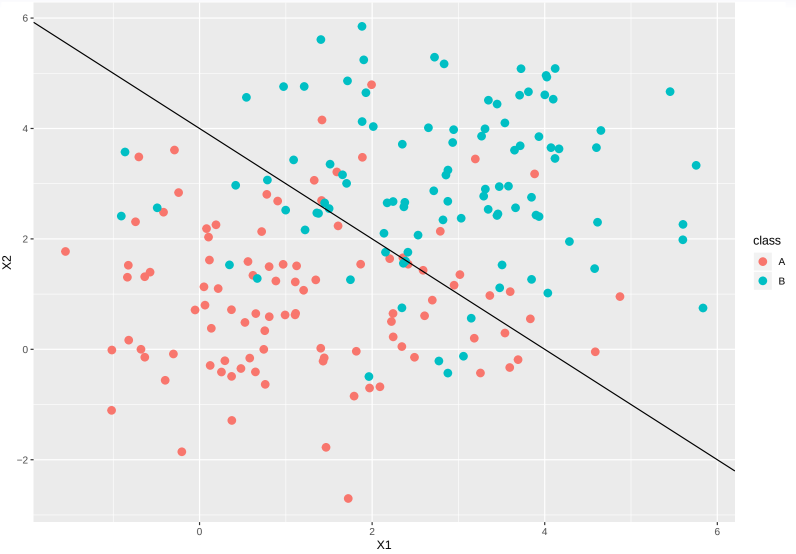
Now we move to the *diagnostic paradigm* and the K-nearest neighbor classifier.

The K-nearest neighbour classifier estimates $P(Y = k \mid X = x)$ and classifies a new observation based on this estimated probability

Synthetic example for KNN classification

Consider a two-class example, and equal class prior probabilites.
A new observation x_0 will be classified to A if
$P(Y = A|X = x_0) > 0.5$ and to class B otherwise.

- The figure below shows a plot of 100 observations from two classes A (red dots) and B (turquoise dots),
- simulated from a bivariate normal distribution with mean vectors $\mu_A = (1, 1)^T$ and $\mu_B = (3, 3)^T$ and a covariance matrix $\Sigma_A = \Sigma_B = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$.
- We want to find a rule to classify a new observation to class A or B.


6
4
2
0
−2
X2
0
2
4
6
X1
class
A
B

Remark: since the truth is known here we can calculate the Bayes boundary and the Bayes error.

Since we have bivariate normal class distributions with common covariance matrix, the optimal boundary is given by LDA. The boundary will be at $\delta_A(\mathbf{x}) = \delta_B(\mathbf{x})$, where $\delta_A(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_A - \frac{1}{2}\boldsymbol{\mu}_A^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_A + \log \pi_A$, and for $\delta_B(\mathbf{x})$ with $\boldsymbol{\mu}_B$.

$$\mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_A - \frac{1}{2}\boldsymbol{\mu}_A^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_A + \log \pi_A = \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_B - \frac{1}{2}\boldsymbol{\mu}_B^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_B + \log \pi_B$$

$$\mathbf{x}^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_A - \boldsymbol{\mu}_B) - \frac{1}{2}\boldsymbol{\mu}_A^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_A + \frac{1}{2}\boldsymbol{\mu}_B^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_B + \log \pi_A - \log \pi_B = 0$$

Inserting numerical values gives: $-x_1 - x_2 + 4 = 0$, and boundary $x_2 = 4 - x_1$.

```r
muA = matrix(c(1, 1), ncol = 1)
muB = matrix(c(3, 3), ncol = 1)
sigmainv = diag(2)/2
sigmainv %*% (muA - muB)
```

K-nearest neighbour classifier

(warning: K is not the number of classes, but neighbours...)

The K-nearest neighbour classifier (KNN) works in the following way:

- Given a new observation x_0 it searches for the K points in our training data that are closest to it.
- These points make up the neighborhood of x_0, $\mathcal{N}_0$.
- The point x_0 is classified by taking a majority vote of the neighbors.
- This means that KNN estimate the posterior class probability as:

$$\hat{P}(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j).$$

What did we cover today - and what is left?

- What is classification and discrimination?
- What is the Bayes classifier and the Bayes risk?
- What is the sampling paradigm and what is modelled then?
- Linear discriminant analysis: model, method, results.
- Quadratic discriminant analysis: model, method, results.
- Naive Bayes - when and why?
- That is the diagnostic paradigm and what is modelled then?
- KNN - majority vote or estimate posterior class probability?

Part B: more on methods that model and estimate the posterior class probabilites.

So far - and moving on

Part B: Modelling posterior probabilites, ROC/AUC and comparisons

What to remember from Part A?

Aim: Discrimination and classification

Today - data from:

- Default: will a new customer `default` or not based on his/her status (`student` or not), `balance` and `income`?
- South African heart disease data set: classify to coronary heart disease or not, based on 9 covariates.

Notation

Training set: observations (independent pairs) $\{(x_1, y_1), ..., (x_n, y_n)\}$ where the response variable Y is *qualitative* and labelled $1, 2, ..., K$.

The training set is used to construct the classification rule (by estimating parameters in class densities or posterior probabilites).

Test set: observations (independent pairs), same format as the training set.

The test set is used to evaluate the classification rule.

Loss function:: The misclassifications are given the loss 1 and the correct classifications loss 0 - this is called *0/1-loss*.

Bayes classifier

- Assume that we know or can estimate the probability that a new observation x_0 belongs to class k:

  $$p_k(x_0) = P(Y = k|X = x_0), \quad k = 1, 2, ...K.$$

  This is the probability that $Y = k$ given the observation x_0. The *Bayes classifier assigns an observation to the most likely class*, given its predictor values.

Two paradigms

- The **sampling paradigm**: There focus is on estimating the prior probabilities for the classes and the class conditional distributions. We classify to the class with the maximal product $\pi_k f_k(x)$. We have looked at LDA (multivariate normal densities with equal covariance matrices) and QDA (ditto, but each class has it's own covariance matrix).
- The **diagnostic paradigm**: We focus on *directly* estimating the posterior distribution for the classes $P(Y = k \mid X = x)$. We have looked at the KNN-classifier in Part A.

Focus now is on *diagnostic paradigm* = we estimates $P(Y = k \mid X = x)$ and classify a new observation based on this estimated probability.
But first, what about linear regression Y on $\mathbf{x}$ to make a classification?

Using linear regression on a classification problem

Example 1: This example uses the `Default` data set from the `ISLR` package. Suppose we want to predict if a new customer will `default` or not based on his/her `balance` or `income`. We try to model this using a simple linear regression and a binary response variable:

$$Y = \begin{cases} 1 & \text{if default = "Yes"} \\ 0 & \text{if default = "No"} \end{cases} .$$

It would be tempting to do the classification according to the rule: classify as *yes* if $\hat{Y} > 0.5$, else as *no*.

default

student

balance

income

No

No

729.5265

44361.625

No

Yes

817.1804

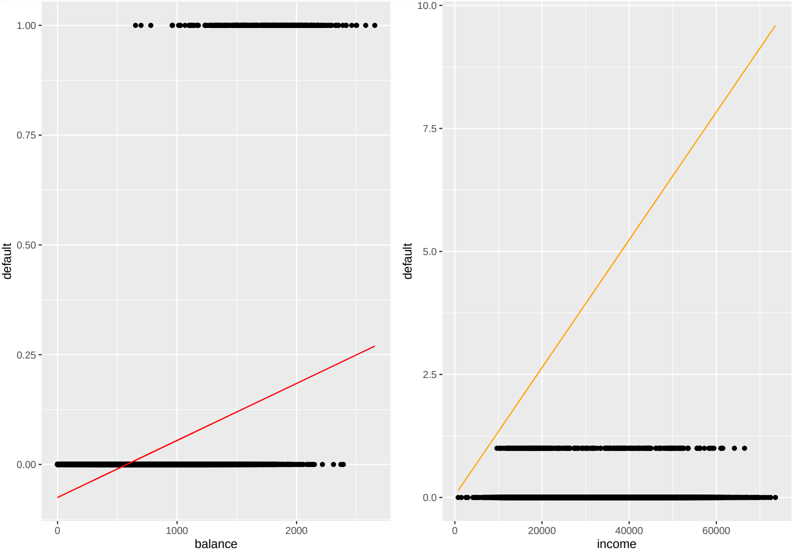
12106.135

No

No

1073.5492

31767.139


1.00
0.75
0.50
0.25
0.00
default
0
1000
2000
balance
10.0
7.5
5.0
2.5
0.0
default
0
20000
40000
60000
income

The above plots shows `default` as a function of `balance` and `default` as a function of `income` with corresponding fitted linear regression lines (red for `x=balance` and orange for `x=income`). Notice that linear regression in this case produces predictions that are smaller than zero or bigger than one, this it is hard to interpret these as probabilities.

It is still possible to use linear regression for classification problems with two classes. It turns out that - if the conditional class densities are (multivariate) normal with equal covariance matrices then this linear regression (with 0 and 1 response) will in fact give the same classification as LDA. See e.g. Ripley (1995), Section 3.2.

Using linear regression on a classification problem?

Example 2: Suppose we want to classify a film. We have defined three classes: { drama, comedy, science-fiction}. We could try to model this using linear regression and the following coding:

$$Y = \begin{cases} 1 & \text{if drama,} \\ 2 & \text{if comedy,} \\ 3 & \text{if science-fiction.} \end{cases}$$

However, this coding implies an ordering of the variables and that the difference between the classes is equal. There is in general no natural way to code a quantitative variable with more than two classes such that it can be used with linear regression.

So, using linear regression to solve a classification problem seems hard with more than two classes - as done here. But, it turns out that using a dummy variable conding for the classes, it is possible to produce the same result as LDA (also with many classes). This is the starting point for *flexible discriminant analysis*.

Linear regression to do classification is not a bad idea, but requires some extra work (multivariate Y due to the dummy variable coding). Therefore we leave linear regression for now.

For two classes *binary regression*, in particular *logistic regression*, is very popular - and is up next.

Logistic regression

In logistic regression we consider a classification problem with two classes.

The model

Assume that Y is coded ($\mathcal{C} = \{1, 0\}$ or {success, failure}), and we focus on success. We may assume that Y_i follows a Bernoulli distribution with probability of success p_i.

$$Y_i = \begin{cases} 1 \text{ with probability } p_i, \\ 0 \text{ with probability } 1 - p_i. \end{cases}$$

In logistic regression we *link* together our covariates $\mathbf{x}_i$ with this probability p_i using a *logistic function*.

In the case of one covariate, the logistic function has the form:

$$p_i = \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}}.$$

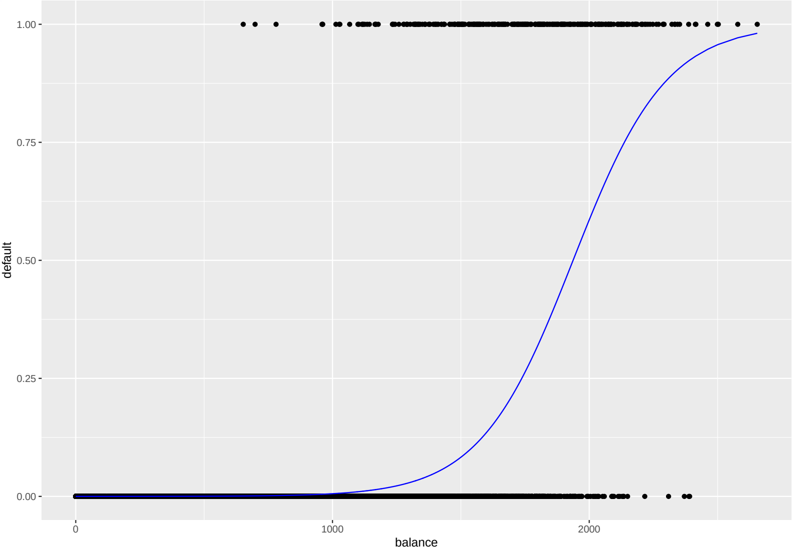
This function is S-shaped, and ranges between 0 and 1 (so the p_i is between 0 and 1). The parameter β_1 determines the rate of increase or decrease of the S-shaped curve, and the sign indicates whether the curve ascends or descends.

Q: Where did that come from? There are other transforms that takes a linear predictor and transforms into the range 0 to 1.

Logistic regression ensures that the estimated probabilities lie in the interval between 0 and 1. This is illustrated in the figure below. The blue line shows the fitted line when performing logistic regression on `default` as a function of `balance`.

The parameters are estimated using the method of maximum likelihood - we will look at that soon, but first we look at how to interpret the estimated parameters.

Example estimated β and logistic curve



Default data: here $\hat{\beta}_0$ = -10.65 and $\hat{\beta}_1 = 0.005$.

Observe effect of intercept and slope term:

$$p_i = \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}}$$

Solid lines: $\beta_0 = 0$ and β_1 is 0.8 (blue), 1 (red) and 2 (orange). Dashed lines $\beta_0 = 1$.

```
library(ggplot2)
ggplot(data.frame(x = c(-6, 5)), aes(x)) + xlab(expression(x)) +
    stat_function(fun = function(x) exp(x)/(1 + exp(x)), geom =
        colour = "red") + stat_function(fun = function(x) exp(2
    exp(2 * x)), geom = "line", colour = "orange") + stat_functi
    x)/(1 + exp(0.8 * x)), geom = "line", colour = "blue") + sta
    x)/(1 + exp(1 + x)), geom = "line", colour = "red", linetype
    stat_function(fun = function(x) exp(1 + 2 * x)/(1 + exp(1 +
        geom = "line", colour = "orange", linetype = "dashed") +
    0.8 * x)/(1 + exp(1 + 0.8 * x)), geom = "line", colour = "bl
    linetype = "dashed") + scale_colour_manual("0+k x", values =
    "orange", "blue"), labels = c("1", "2", "0.8"))
```

The odds and the odds ratio

For the logistic regression it is hard to give a simple interpretation regression coefficients β, because an increase in x_1 by one unit does not give the same increase (decrease) in the probability p_i for all values of x_1. But, looking at odds - it is simpler to explain what the βs mean.

For a probability p_i the ratio $\frac{p_i}{1-p_1}$ is called the *odds*.

If $p_i = \frac{1}{2}$ then the odds is 1, and if $p_i = \frac{1}{4}$ then the odds is $\frac{1}{3}$. We may make a table for probability vs. odds in R:

```r
library(knitr)
library(kableExtra)
p = seq(0.1, 0.9, 0.1)
odds = p/(1 - p)
kable(t(data.frame(p, odds)), digits = c(2, 2), format = whichfo
```

p
0.10
0.20
0.30
0.40
0.5

Why is the odds relevant?

(Since p is used for probability we use r for number of covariates now.)

Let us assume that we have r covariates, and we use η_i (linear predictor) to help with our notation.

$$
\begin{aligned}
\eta_i &= \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_r x_{ir} \\
p_i &= \frac{\exp(\eta_i)}{1+\exp(\eta_i)} \\
\eta_i &= \ln(\frac{p_i}{1-p_i}) \\
\ln(\frac{p_i}{1-p_i}) &= \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_r x_{ir} \\
\frac{p_i}{1-p_i} &= \frac{P(Y_i=1|\mathbf{x}_i)}{P(Y_i=0|\mathbf{x}_i)} = \exp(\beta_0)\cdot\exp(\beta_1 x_{i1})\cdots\exp(\beta_r x_{ir})
\end{aligned}
$$

We have a *multiplicative model* for the odds - which can help us to interpret our βs.

In addition we see that the *logit* of p_i, $\ln(\frac{p_i}{1-p_i})$, is linear in the βs (and in the x's).

So, what if we increase x_{1i} to $x_{1i} + 1$?

If the covariate x_{1i} increases by one unit (while all other covariates are kept fixed) then the odds is multiplied by $\exp(\beta_1)$:

$$
\begin{aligned}
\frac{P(Y_i = 1 \mid x_{i1} + 1)}{P(Y_i = 0) \mid x_{i1} + 1)} &= \exp(\beta_0) \cdot \exp(\beta_1(x_{i1} + 1)) \exp(\beta_2(x_{i2})) \cdots \exp(\beta_r x_{ir}) \\
&= \exp(\beta_0) \cdot \exp(\beta_1 x_{i1}) \exp(\beta_1) \exp(\beta_2 x_{i2}) \cdots \exp(\beta_r x_{ir}) \\
&= \frac{P(Y_i = 1 \mid x_{i1})}{P(Y_i = 0 \mid x_{i1})} \cdot \exp(\beta_1)
\end{aligned}
$$

This means that if x_{i1} increases by 1 then: if $\beta_1 < 0$ we get a decrease in the odds, if $\beta_1 = 0$ no change, and if $\beta_1 > 0$ we have an increase. Here $\exp(\beta_1)$ is easier to interpret than β_1.

Default-example

Default as response and student, balance and income as covariates
Result:

$$\frac{P(Y_i = 1 \mid x_{i1} + 1)}{P(Y_i = 0) \mid x_{i1} + 1)} = \frac{P(Y_i = 1 \mid x_{i1})}{P(Y_i = 0 \mid x_{i1})} \cdot \exp(\beta_1)$$

What is done below? Explain what the effect of `student` gives.

```r
colnames(Default)
fit = glm(default ~ student + balance + income, family = "binomial",
    data = Default)
coef(fit)
round(exp(coef(fit)), 3)
```

```
## [1] "default" "student" "balance" "income"
##   (Intercept)    studentYes       balance        income
## -1.086905e+01 -6.467758e-01  5.736505e-03  3.033450e-06
## (Intercept)  studentYes     balance      income
##       0.000       0.524       1.006       1.000
```

Maximum likelihood estimation

We assume that pairs of covariates and responses $\{x_i, y_i\}$ are measured independently of each other. Given n such observation pairs, the likelihood function of a logistic regression model can be written as:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} L_i(\boldsymbol{\beta}) = \prod_{i=1}^{n} f(y_i; \boldsymbol{\beta}) = \prod_{i=1}^{n} (p_i)^{y_i} (1 - p_i)^{1-y_i},$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \ldots, \beta_r)^T$ enters into p_i.

$$p_i = \frac{\exp(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ir})}{1 + \exp(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_r x_{ir})}$$

The maximum likelihood estimates are found by maximizing the likelihood, and since the log is a monotone transform (and maximizing the log-likelihood will give the same result as maximizing the likelihood) we usually work with the log-likelihood (because this makes the maths easier).

$$\begin{aligned}
\ln(L(\boldsymbol{\beta})) = l(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \Big(y_i \log p_i + (1 - y_i) \log(1 - p_i) \Big) \\
&= \sum_{i=1}^{n} \left(y_i \log \left(\frac{p_i}{1 - p_i} \right) + \log(1 - p_i) \right) \\
&= \sum_{i=1}^{n} \left(y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_r x_{ir}) - \log(1 + e^{\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ir}} \right).
\end{aligned}$$

- To maximize the log-likelihood function we find the $r+1$ partial derivatives, and set equal til 0.
- This gives us a set of $r+1$ non-linear equations in the βs.
- This set of equations does not have a closed form solution.
- These equations are therefore solved numerically. The *Newton-Raphson algorithm* (or Fisher Scoring) is used.

```r
fit = glm(default ~ student + balance + income, family = "binomial",
    data = Default)
summary(fit)
```

```
##
## Call:
## glm(formula = default ~ student + balance + income, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4691  -0.1418  -0.0557  -0.0203   3.7383
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
## studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 **
## balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
## income       3.033e-06  8.203e-06   0.370  0.71152
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1571.5  on 9996  degrees of freedom
## AIC: 1579.5
```

Inference

We may construct confidence intervals and test hypotheses about the βs, with the aim to understand which covariate that contributes to our posterior probabilites and classification.
This is done by assuming that each $\hat{\beta}_j$ is approximately normally distributed with mean β_j and variance $\hat{\text{Var}}(\hat{\beta}_j)$ (related to the negative of the inverse of the expected Hessian of the loglikelihood function).

The Akaike Information Criterion (AIC) for model selection

The AIC score is given by:

$$AIC = 2 \cdot (r+1) - 2 \cdot \text{loglik},$$

where $r+1$ is the number of model parameters. The loglik is the maximized log-likelihood $l(\hat{\boldsymbol{\beta}})$ and $\hat{\boldsymbol{\beta}}$ is the maximum-likelihood estimate of the parameter-vector $\boldsymbol{\beta} = (\beta_0, \beta_1, ..., \beta_r)^T$. The role of $r+1$ is to penalize models with many parameters as a high number of parameters may lead to overfitting. The AIC value can be used to choose between candidate logistic regression models, where the model with the lowest AIC value is the one expected to give the best fit. More about the AIC in Module 6.

Predictions

- We fit a (simple) logistic regression model to our data set, and
- get parameter estimates $\hat{\beta}_0$ and $\hat{\beta}_1$.
- We want to use this model to make a prediction when given a new observation x_0.

$$\hat{p}(x_0) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 x_0}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 x_0}}$$

This $\hat{p}(x_0)$ is the estimated probability that the new observation x_0 belongs to the class defined by $Y = 1$.
In the case of qualitative covariates, a dummy variable needs to be introduced. This is done in a similar fashion as for linear regression.

Want to learn more (theory) about logistic regression?

In TMA4315 Generalized linear models we spent 3 weeks with binary regression - mainly logistic regression. The focus there was on all parts of the regression (not classification) with a mathematical focus on estimation, inference, model fit.

Example: South African heart disease data set

In this example we use the `SAhert` data set from the `ElemStatLearn` package. This is a retrospective sample of males in a heart-disease high-risk region in South Africa. It consists of 462 observations on the 10 variables. All subjects are male in the age range 15-64. There are 160 cases (individuals who have suffered from a conorary heart disease) and 302 controls (individuals who have not suffered from a conorary heart disease).

The response value (`chd`) and covariates

- `chd` : conorary heart disease {yes, no} coded by the numbers {1, 0}
- `sbp` : systolic blood pressure
- `tobacco` : cumulative tobacco (kg)
- `ldl` : low density lipoprotein cholesterol
- `adiposity` : a numeric vector
- `famhist` : family history of heart disease. Categorical variable with two levels: {Absent, Present}.
- `typea` : type-A behavior
- `obesity` : a numerical value
- `alcohol` : current alcohol consumption
- `age` : age at onset

The goal is to identify important risk factors. We start by loading and looking at the data:

```r
library(ElemStatLearn)
library(knitr)
library(kableExtra)
heartds = SAheart
heartds$chd = as.factor(heartds$chd)
kable(head(heartds), format = whichformat)
```

sbp

tobacco

ldl

adiposity

famhist

typea

obesity

alcohol

age

chd

160

12.00

5.73

In order to investigate the data further, we use the `ggpairs` function from the `GGally` library, to make scatter plots of the covariates. The coloring is done according to the response variable, where green represents a case $Y = 1$ and red represents a control $Y = 0$.

```
library(ggplot2)
library(GGally)
ggpairs(heartds, ggplot2::aes(color=chd), #upper="blank",
        lower = list(continuous = wrap("points", alpha = 0.3, si
```

We now fit a (multiple) logistic regression model using the `glm` function and the full data set. In order to fit a logistic model, the `family` argument must be set equal to `="binomial"`. The `summary` function prints out the estimates of the coefficients, their standard errors and z-values. As for a linear regression model, the significant coefficients are indicated by stars where the significant codes are included in the `R` outprint.

```r
glm_heart = glm(chd ~ ., data = heartds, family = "binomial")
summary(glm_heart)
```

```
##
## Call:
## glm(formula = chd ~ ., family = "binomial", data = heartds)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7781  -0.8213  -0.4387   0.8889   2.5435
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -6.1507209  1.3082600  -4.701 2.58e-06 ***
## sbp             0.0065040  0.0057304   1.135 0.256374
## tobacco         0.0793764  0.0266028   2.984 0.002847 **
## ldl             0.1739239  0.0596617   2.915 0.003555 **
## adiposity       0.0185866  0.0292894   0.635 0.525700
## famhistPresent  0.9253704  0.2278940   4.061 4.90e-05 ***
## typea           0.0395950  0.0123202   3.214 0.001310 **
## obesity        -0.0629099  0.0442477  -1.422 0.155095
## alcohol         0.0001217  0.0044832   0.027 0.978350
## age             0.0452253  0.0121298   3.728 0.000193 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

Estimated coeffs

exp(estimated coeffs)

(Intercept)

-6.151

0.002

sbp

0.007

1.007

tobacco

0.079

1.083

ldl

0.174

1.190

adiposity

0.019

Multinomial logistic regression

The logistic regression model can be generalized for a response variable with more than two classes. Assume we have a response variable with K possible classes and r covariates. The probability that Y belongs to class k, given an observation vector $\mathbf{x} = (x_1, x_2, \dots, x_r)^T$ is (usually) modelled by:

$$\ln \frac{P(Y = k|\mathbf{x})}{P(Y = K|\mathbf{x})} = \beta_{0k} + \beta_{1k}x_1 + \cdots + \beta_{rk}x_r.$$

The multinomial logistic regression model is implemented in the `glmnet` or `VGAM` package in `R`.
We will not discuss this further since LDA is more popular (than logistic regression) in the multi-class setting. And, as we shall see soon - they are not that different.

Confusion matrix, sensitivity, specificity

In a two class problem - assume the classes are labelled “-” (non disease) and “+” (disease). In a population setting we define the following event and associated number of observations.

Predicted -

Predicted +

Total

True -

True Negative TN

False Positive FP

N

True +

False Negative FN

True Positive TP

P

Total

Sensitivity is the proportion of correctly classified positive observations: $\frac{\#\text{True Positive}}{\#\text{Condition Positive}} = \frac{\text{TP}}{\text{P}}$.

Specificity is the proportion of correctly classified negative observations: $\frac{\#\text{True Negative}}{\#\text{Condition Negative}} = \frac{\text{TN}}{\text{N}}$.

We would like that a classification rule (or a diagnostic test) have both a high sensitivity and a high specificity.

Other useful quantities:

Name

Definition

Synonyms

False positive rate

FP/N

Type I error, 1-specificity

True positive rate

TP/P

1-Type II error, power, sensitivity, recall

Positive predictive value (PPV)

TP/P*

Precision, 1-false discovery proportion

Negative predictive value (NPV)

TN/N*

(These two tables are tables 4.6 and 4.7 in our ISL textbook.)

Example Continued: South African heart disease

We want to evaluate our multiple logistic model for the `SAheart` data set. In order to investigate the training error and the test error, we divide the original data set, randomly, into two samples of equal size.

```r
set.seed(20)
train_ID = sample(1:nrow(heartds), nrow(heartds)/2)
train_SA = heartds[train_ID, ]
test_SA = heartds[-train_ID, ]
```

We now fit a logistic regression model, using the training set only:

```
glm_SA = glm(chd ~ ., data = train_SA, family = "binomial")
summary(glm_SA)
```

```
##
## Call:
## glm(formula = chd ~ ., family = "binomial", data = train_SA)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0583  -0.7660  -0.4523   0.8679   2.2682
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -5.4999981  1.9010122  -2.893  0.00381 **
## sbp            0.0050738  0.0089677   0.566  0.57154
## tobacco        0.1228036  0.0443098   2.771  0.00558 **
## ldl            0.0315840  0.0817017   0.387  0.69907
## adiposity      0.0223541  0.0398203   0.561  0.57454
## famhistPresent 0.7763880  0.3434487   2.261  0.02379 *
## typea          0.0328267  0.0181739   1.806  0.07088 .
## obesity       -0.0487256  0.0592647  -0.822  0.41098
## alcohol       -0.0007868  0.0079569  -0.099  0.92123
## age            0.0470105  0.0170190   2.762  0.00574 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

By comparing this outprint with the corresponding outprint above, we see that the estimated coefficients slightly differ. This is because a different data set has been used to fit the model. We previously used the full data set.

We want to estimate the probability of a `chd` event for the observations in the test set. To do this we can insert the estimated coefficient into the logistic equation, remembering that `famhist` is a categorical covariate, modeled by a dummy variable:

$$x_{\text{famhist}} = \begin{cases} 1, & \text{if Present,} \\ 0, & \text{if Absent.} \end{cases}$$

The estimated probability of $Y = 1$ if `famhist = "Present"`, given a vector $\mathbf{X}$ of covariate observations is:

$$\hat{p}(\mathbf{X}) = \frac{e^{-7.43+0.01x_{\text{sbp}}+0.09x_{\text{tobacco}}+0.16x_{\text{ldh}}+0.01x_{\text{adiposity}}+1.04\cdot 1+0.04x_{\text{typea}}-0.04x}}{1+e^{-7.43+0.01x_{\text{sbp}}+0.09x_{\text{tobacco}}+0.16x_{\text{ldh}}+0.01x_{\text{adiposity}}+1.04\cdot 1+0.04x_{\text{typea}}-0.0}}$$

Whereas, if `famhist = "Absent"` the estimated probability is:

$$\hat{p}(\mathbf{X}) = \frac{e^{-7.43+0.01x_{\text{sbp}}+0.09x_{\text{tobacco}}+0.16x_{\text{ldh}}+0.01x_{\text{adiposity}}+1.04\cdot 0+0.04x_{\text{typea}}-0.04x}}{1+e^{-7.43+0.01x_{\text{sbp}}+0.09x_{\text{tobacco}}+0.16x_{\text{ldh}}+0.01x_{\text{adiposity}}+1.04\cdot 0+0.04x_{\text{typea}}-0.0}}$$

The `predict` function does these calculations for us. When specifying `type="response"` the function returns the probabilities for $Y = 1$.

```
probs_SA = predict(glm_SA, newdata = test_SA, type = "response")
```

From these probabilities we can obtain classifications, by specifying a threshold value. We have here chosen a threshold value of 0.5. By using the `ifelse` function we specify that all probabilities larger than 0.5 are to be classified as 1, while the remaining probabilities are to be classified as 0.

```
pred_SA = ifelse(probs_SA > 0.5, 1, 0)

predictions_SA = data.frame(probs_SA, pred_SA, test_SA[, 10])
colnames(predictions_SA) = c("Estim. prob. of Y=1", "Predicted c
    "True class")
kable(head(predictions_SA), format = whichformat, booktabs = TRU
```

Estim. prob. of Y=1

Predicted class

True class

1

0.7316942

1

1

4

0.7181876

1

We can now use the confusion matrix to count the number of misclassifications. The below confusion matrix is calculated using the test set and comparing the predicted classes with the true classes.

```r
table(pred_SA, SAheart[-train_ID, 10])
```

```
##
## pred_SA   0   1
##       0 117  44
##       1  28  42
```

The logistic model has correctly classified 130+40 times, and misclassified 24+37 times. The misclassification test error rate is thus:

$$\text{Test error} = \frac{24+37}{231} \approx 0.264$$

The training error can be calculated in a similar fashion, but now we use the fitted model to make prediction for the training set.

```r
SA_train_prob = glm_SA$fitted.values
SA_train_pred = ifelse(SA_train_prob > 0.5, 1, 0)
conf_train = table(SA_train_pred, SAheart[train_ID, 10])
misclas_train = (231 - sum(diag(conf_train)))/231
misclas_train
```

```
## [1] 0.2380952
```

The train misclassification error rate is $\approx 25.1\%$.

ROC curves and AUC

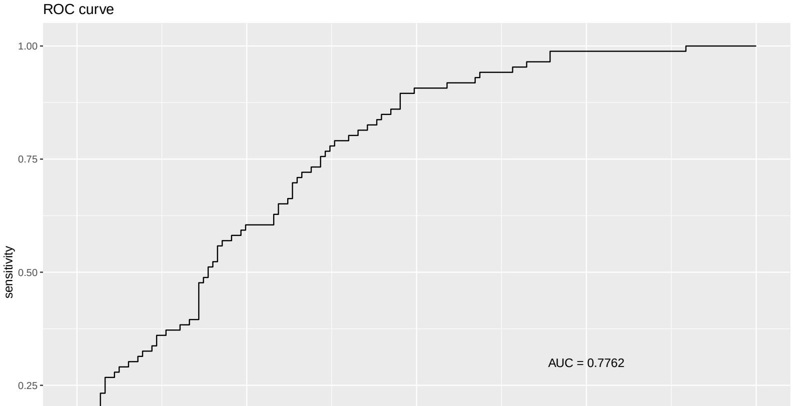
The receiver operating characteristics (ROC) curve gives a graphical display of the sensitivity against specificity, as the threshold value (cut-off on probability of success or disease) is moved over the range of all possible values. An ideal classifier will give a ROC curve which hugs the top left corner, while a straight line represents a classifier with a random guess of the outcome.

The **AUC** score is the area under the AUC curve. It ranges between the values 0 and 1, where a higher value indicates a better classifier. The AUC score is useful for comparing the performance of different classifiers, as all possible threshold values are taken into account.

Example Continued: South African heart disease

In order to see how our model performs for different threshold values, we can plot a ROC curve:

```
library(pROC)
SA_roc = roc(SAheart[-train_ID, 10], probs_SA, legacy.axes = TRU
ggroc(SA_roc) + ggtitle("ROC curve") + annotate("text", x = 0.25
    label = "AUC = 0.7762")
```



To check where in the plot we find the default cut-off on 0.5, we need to calculate sensitivity and specificity for this cut-off:

```
res = table(pred_SA, SAheart[-train_ID, 10])
sens = res[2, 2]/sum(res[2, ])
spec = res[1, 1]/sum(res[1, ])
sens
```

```
## [1] 0.6
```

```
spec
```

```
## [1] 0.7267081
```

Observe that the value 0.6 (on y-axis) and 0.7267081 (on x-axis) is on our ROC curve.

The ROC-curve is made up of all possible cut-offs and their associated sensitivity and specificity.

Which classification method is the best?

Advantages of discriminant analysis

- Discriminant analysis is more stable than logistic regression when the classes are well-separated.
- Discriminant analysis is more stable than logistic regression if the number of observations n is small and the distribution of the predictors $\mathbf{X}$ is approximately (multivariate) normal.

Linearity

Assume a binary classification problem with one covariate. Recall that logistic regression can be written:

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x.$$

For LDA we have that $p_0(x)$ is the probability that the observation x belongs to class 0, while $p_1(x) = 1 - p_0(x)$ is the probability that it belongs to class 1.

Observe that this show that our class boundary is linear.

Compulsory Exercise 1, Problem 3a.

$$\begin{aligned}
\log \frac{P(Y=0|X=x)}{P(Y=1|X=x)} &= \log \frac{\pi_0}{\pi_1} + \log \frac{f_0(x)}{f_1(x)} \\
&= \log \frac{\pi_0}{\pi_1} - \frac{1}{2\sigma^2}(x-\mu_0)^2 + \frac{1}{2\sigma^2}(x-\mu_1)^2 \\
&= \log \frac{\pi_0}{\pi_1} - \frac{1}{2\sigma^2}(x^2 - 2x\mu_0 + \mu_0^2 - x^2 + 2x\mu_1 - \mu_1^2) \\
&= \log \frac{\pi_0}{\pi_1} - \frac{1}{2\sigma^2}(\mu_0^2 - \mu_1^2) + \frac{1}{\sigma^2}(\mu_0 - \mu_1)x \\
&= \alpha_0 + \alpha_1 x
\end{aligned}$$

The two methods can thus be written in the same form (linear in parameters and in x). The difference is in *how* the parameters $(\alpha_0, \alpha_1, \beta_0, \beta_1)$ are estimated.

LDA vs logistic regression

- Logistic regression uses the diagnostic paradigm, and models the posterior distribution $P(Y = 1|\mathbf{x})$.
- Linear discriminant analysis models the class conditional densities $f_k(\mathbf{x})$.
- The results are usually quite similar, but
    - LDA is "more available" in the multi-class setting
    - if the class conditional distributions are multivariate normal then LDA (or QDA) is preferred
    - if the class conditional distributions are far from multivariate normal then logistic regression is preferred
    - in medicine for two-class problems logistic regression is often preferred (for interpretability) and (always) together with ROC and AUC (for model comparison).

and KNN?

- KNN is used when the class boundaries are non-linear.

Extensions for classifications

- Module 5: how to use cross-validation in model evaluation and model selection
- (Module 6: model selection - but mainly regression)
- Module 7: maybe a taste of nonlinear methods
- Module 8: classification trees (binary splits for the covariates)
- Module 9: support vector machines
- Module 11: neural nets

Further reading

- More on logistic regression from TMA4315 Generalized linear models H2018: TMA4315M3: Binary regression
- Videoes on YouTube by the authors of ISL, Chapter 4

R packages

```r
# packages to install before knitting this R Markdown file to kn
# the Rmd
install.packages("knitr")
install.packages("rmarkdown")
# nice tables in Rmd
install.packages("kableExtra")
# cool layout for the Rmd plotting
install.packages("ggplot2")  # cool plotting
install.packages("ggpubr")  # for many ggplots
install.packages("GGally")  # for ggpairs
# datasets
install.packages("ElemStatLearn")
install.packages("ISLR")
# data manipulations
install.packages("dplyr")
install.packages("reshape")
# classificaton
install.packages("class")
install.packages("pROC")
# div statistics
```

Acknowledgements

Thanks to Julia Debik for contributing to this module page.