Multivariate Statistics and Machine Learning
MATH38161

Korbinian Strimmer

16 September 2020

Contents

Preface

About these notes

This is the course text for MATH38161, an introductory course in **Multivariate Statistics and Machine Learning** for third year mathematics students.

These notes will be updated from time to time. To view the current version in your browser visit the online MATH38161 lecture notes. You may also downlad the MATH38161 lecture notes as PDF.

About the author

My name is Korbinian Strimmer and I am a Professor in Statistics in the Statistics group of the Department of Mathematics at the University of Manchester. You can find more information about me on my home page.

I have first taught this module in the Winter term 2018 at the University of Manchester.

I hope you enjoy the course. If you have any questions, comments, or corrections then please email me at korbinian.strimmer@manchester.ac.uk

About the module

Topics covered

The MATH38161 module is designed to run over the course of 11 weeks. It has six parts, each covering a particular aspect of multivariate statistics and machine learning:

1. Multivariate random variables and estimation in large and small sample settings (W1 and W2)
2. Transformations and dimension reduction (W3 and W4)
3. Unsupervised learning/clustering (W5 and W6)
4. Supervised learning/classification (W7 and W8)
5. Measuring and modelling multivariate dependencies (W9)
6. Nonlinear and nonparametric models (W10, W11)

This module focuses on:

- *Concepts and methods* (not on theory)
- *Implementation and application in R*
- *Practical data analysis and interpretation* (incl. report writing)
- *Modern tools in data science and statistics* (R markdown, R studio)

Additional support material

Accompanying these notes are

- a weekly learning plan for an 11 week study period,
- corresponding worksheets, one for each week, with examples (theory and application in R) and solutions in R Markdown,
- lecture videos (visualiser style).

Organisatorical information is available from the course home page and on Blackboard.

If you are a University of Manchester student and enrolled for this module you can find the exam questions of previous years (without solution) as well as the coursework instructions on Blackboard.

Furthermore, there is also an MATH38161 online reading list hosted by the University of Manchester library.

Acknowledgements

Many thanks to Beatriz Costa Gomes for her help to compile the first draft of these course notes in the winter term 2018 while she was a graduate teaching assistant for this course. I also thank the many students who suggested corrections.

Chapter 1

Multivariate random variables

1.1 Why multivariate statistics?

Science uses experiments to verify hypotheses about the world. Statistics provides tools to quantify this procedure and offers methods to link data (experiments) with probabilistic models (hyptheses). Since the world is complex we need complex models and complex data, hence the need for multivariate statistics and machine learning.

Specifically, multivariate statistics (as opposed to univariate statistics) is concerned with methods and models for **random vectors** and **random matrices**, rather than just random univariate (scalar) variables. Therefore, in multivariate statistics we will frequently make use of matrix notation.

Closely related to multivariate statistics (traditionally a subfield of statistics) is machine learning (ML) which is traditionally a subfield of computer science. ML used to focus more on on algorithms rather on probabilistic modeling but nowadays most machine learning methods are fully based on statistical multivariate approaches, so the two fields are converging.

Learning multivariate models allows us to learn dependencies and interactions among the components of the random variables which in turns allows to draw conclusion about the world.

Two main tasks:

- unsupervised learning (finding structure, clustering)
- supervised learning (training from labeled data, followed by prediction)

Challenges:

- complexity of model needs to be appropriate for problem and available data,
- high dimensions make estimation and inference difficult
- computational issues.

1.2 Basics

1.2.1 Univariate vs. multivariate random variables

Univariate random variable (dimension $d = 1$):

$$x \sim F$$

where x is a **scalar** and F is the distribution. $\text{E}(x) = \mu$ denotes the mean and $\text{Var}(x) = \sigma^2$ the variance of x.

Multivariate random **vector** of dimension d:

$$\boldsymbol{x} = (x_1, x_2, ..., x_d)^T \sim F$$

$\boldsymbol{x}$ is **vector** valued random variable.

The vector $\boldsymbol{x}$ is column vector (=matrix of size $d \times 1$). Its components $x_1, x_2, ..., x_d$ are univariate random variables. The dimension d is also often denoted by p or q.

1.2.2 Mean of a random vector

The mean / expectation of a random vector with dimensions d is also a vector with dimensions d:

$$\text{E}(\boldsymbol{x}) = \boldsymbol{\mu} = \begin{pmatrix} \text{E}(x_1) \\ \text{E}(x_2) \\ \vdots \\ \text{E}(x_d) \end{pmatrix} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_d \end{pmatrix}$$

1.2.3 Variance of a random vector

Recall the efinition of variance for a univariate random variable:

$$\text{Var}(x) = \text{E}\left((x - \text{E}(x))^2\right) = \text{E}\left((x-\mu)^2\right) = \text{E}\left((x-\mu)(x-\mu)\right) = \text{E}(x^2) - \mu^2$$

Definition of **variance of a random vector:**

$$\text{Var}(\boldsymbol{x}) = \text{E}\left(\underbrace{(\boldsymbol{x}-\boldsymbol{\mu})}_{d\times 1} \underbrace{(\boldsymbol{x}-\boldsymbol{\mu})^T}_{1\times d} \right) = \underbrace{\boldsymbol{\Sigma}}_{d\times d} = \text{E}(\boldsymbol{x}\boldsymbol{x}^T) - \boldsymbol{\mu}\boldsymbol{\mu}^T$$

The variance of a random vector is, therefore, **not** a vector but a **matrix**!

$$\boldsymbol{\Sigma} = (\sigma_{ij}) = \begin{pmatrix} \sigma_{11} & \dots & \sigma_{1d} \\ \vdots & \ddots & \vdots \\ \sigma_{d1} & \dots & \sigma_{dd} \end{pmatrix}$$

This matrix is called the **Covariance Matrix**, with off-diagonal elements $\sigma_{ij} = \text{Cov}(x_i, x_j)$ and the diagonal $\sigma_{ii} = \text{Var}(X_i) = \sigma_i^2$.

1.2.4 Properties of the covariance matrix

1. $\boldsymbol{\Sigma}$ is real valued: $\sigma_{ij} \in \mathbb{R}$
2. $\boldsymbol{\Sigma}$ is symmetric: $\sigma_{ij} = \sigma_{ji}$
3. The diagonal of $\boldsymbol{\Sigma}$ contains $\sigma_{ii} = \text{Var}(x_i) = \sigma_i^2$, i.e. the variances of the components of $\boldsymbol{x}$.
4. Off-diagonal elements $\sigma_{ij} = \text{Cov}(x_i, x_j)$ represent linear dependencies among the x_i. $\Longrightarrow$ linear regression, correlation

How many separate entries does $\boldsymbol{\Sigma}$ have?

$$\boldsymbol{\Sigma} = (\sigma_{ij}) = \underbrace{\begin{pmatrix} \sigma_{11} & \dots & \sigma_{1d} \\ \vdots & \ddots & \vdots \\ \sigma_{d1} & \dots & \sigma_{dd} \end{pmatrix}}_{d \times d}$$

with $\sigma_{ij} = \sigma_{ji}$.

Number of separate entries: $\frac{d(d+1)}{2}$.

This numbers grows with the square of the dimension d, i.e. is of order O(d^2):

d	# entries
1	1
10	55
100	5050
1000	500500
10000	50005000

For large dimension d the covariance matrix has many components!

–> computationally expensive (both for storage and in handling) –> very challenging to estimate in high dimensions d.

Note: matrix inversion requires O(d^3) operations! So, computing $\boldsymbol{\Sigma}^{-1}$ is difficult for large d!

1.2.5 Eigenvalue decomposition of $\boldsymbol{\Sigma}$

Theorem from matrix theory / linear algebra: A symmetric matrix with real entries has real eigenvalues.

Thus, the eigenvalues of $\boldsymbol{\Sigma}$ must be real. However, for covariance matrices this can be clarified further:

Assume that $\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ is the eigenvalue decomposition of the covariance matrix $\boldsymbol{\Sigma}$ with $\boldsymbol{U}$ an orthogonal matrix containing the eigenvectors (eigensystem) and $\boldsymbol{\Lambda}$ is the diagonal matrix containing eigenvalues:

$$\boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_d \end{pmatrix}$$

Assume that $\boldsymbol{x}$ is a random vector with covariance $\text{Var}(\boldsymbol{x}) = \boldsymbol{\Sigma}$. Then $\text{Var}(\boldsymbol{U}^T\boldsymbol{x}) = \boldsymbol{U}^T\boldsymbol{\Sigma}\boldsymbol{U} = \boldsymbol{\Lambda}$. Since the variance is always positive or zero the eigenvalues λ_i of a the covariance matrix $\boldsymbol{\Sigma}$ cannot be negative. Hence, **$\boldsymbol{\Sigma}$ is positive semidefinite**.

In fact, **unless there is collinearity** (i.e. a variable is a linear function the other variables) all eigenvalues will be positive and **$\boldsymbol{\Sigma}$ is positive definite**.

1.2.6 Quantities related to the covariance matrix

1.2.6.1 Correlation matrix $\boldsymbol{P}$

The correlation matrix $\boldsymbol{P}$ (= upper case greek "rho") is the standardised covariance matrix

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}} = \text{Cor}(x_i, x_j)$$

$$\rho_{ii} = 1 = \text{Cor}(x_i, x_i)$$

$$\boldsymbol{P} = (\rho_{ij}) = \begin{pmatrix} 1 & \dots & \rho_{1d} \\ \vdots & \ddots & \vdots \\ \rho_{d1} & \dots & 1 \end{pmatrix}$$

where $\boldsymbol{P}$ ("capital rho") is a symmetric matrix ($\rho_{ij} = \rho_{ji}$).

Note the **variance-correlation decomposition**

$$\boldsymbol{\Sigma} = \boldsymbol{V}^{\frac{1}{2}}\boldsymbol{P}\boldsymbol{V}^{\frac{1}{2}}$$

where $\boldsymbol{V}$ is a diagonal matrix containing the variances:

$$\boldsymbol{V} = \begin{pmatrix} \sigma_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_{dd} \end{pmatrix}$$

$$\boldsymbol{P} = \boldsymbol{V}^{-\frac{1}{2}}\boldsymbol{\Sigma}\boldsymbol{V}^{-\frac{1}{2}}$$

This is the definition of correlation written in matrix notation.

1.2.6.2 Precision matrix or concentration matrix

$$\boldsymbol{\Omega} = (\omega_{ij}) = \boldsymbol{\Sigma}^{-1}$$

$\boldsymbol{\Omega}$ ("Omega") is the inverse of the covariance matrix.

The inverse of the covariance matrix can be obtained via the spectral decomposition, followed by inverting the eigenvalues λ_i:

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{U}\boldsymbol{\Lambda}^{-1}\boldsymbol{U}^T = \boldsymbol{U} \begin{pmatrix} \lambda_1^{-1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_d^{-1} \end{pmatrix} \boldsymbol{U}^T$$

Note that **all eigenvalues λ_i need to be positive so that $\boldsymbol{\Sigma}$ can be inverted.** (i.e., $\boldsymbol{\Sigma}$ needs to be positive definite).
If any $\lambda_i = 0$ then $\boldsymbol{\Sigma}$ is singular and not invertible.

Importance of $\boldsymbol{\Sigma}^{-1}$: - Natural parameter in exponential family. - Many expressions in multivariate statistics contain $\boldsymbol{\Sigma}^{-1}$ and not $\boldsymbol{\Sigma}$ - $\boldsymbol{\Sigma}^{-1}$ has close connection with graphical models (e.g. conditional independence graph, partial correlations, see later chapter)

1.2.6.3 Partial correlation matrix

This is a standardised version of the precision matrix, see later chapter on graphical models.

1.2.6.4 Total variation and generalised variance

To summarise the covariance matrix $\boldsymbol{\Sigma}$ in a single scalar value there are two commonly used measures:

- **total variation**: $\text{Tr}(\boldsymbol{\Sigma}) = \sum_{i=1}^d \lambda_i$
- **generalised variance**: $\det(\boldsymbol{\Sigma}) = \prod_{i=1}^d \lambda_i$

If all eigenvalues are positive then $\log\det(\boldsymbol{\Sigma}) = \sum_{i=1}^d \log\lambda_i = \text{Tr}(\log\boldsymbol{\Sigma})$.

$\log\boldsymbol{\Sigma}$ is the matrix logarithm of $\boldsymbol{\Sigma}$ and is given by $\log\boldsymbol{\Sigma} = \boldsymbol{U} \begin{pmatrix} \log\lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \log\lambda_d \end{pmatrix} \boldsymbol{U}^T$

1.3 Multivariate normal distribution

The multivariate normal model is a generalisation of the univariate normal distribution from dimension 1 to dimension d.

1.3.1 Univariate normal distribution:

$$\text{Dimension } d = 1$$

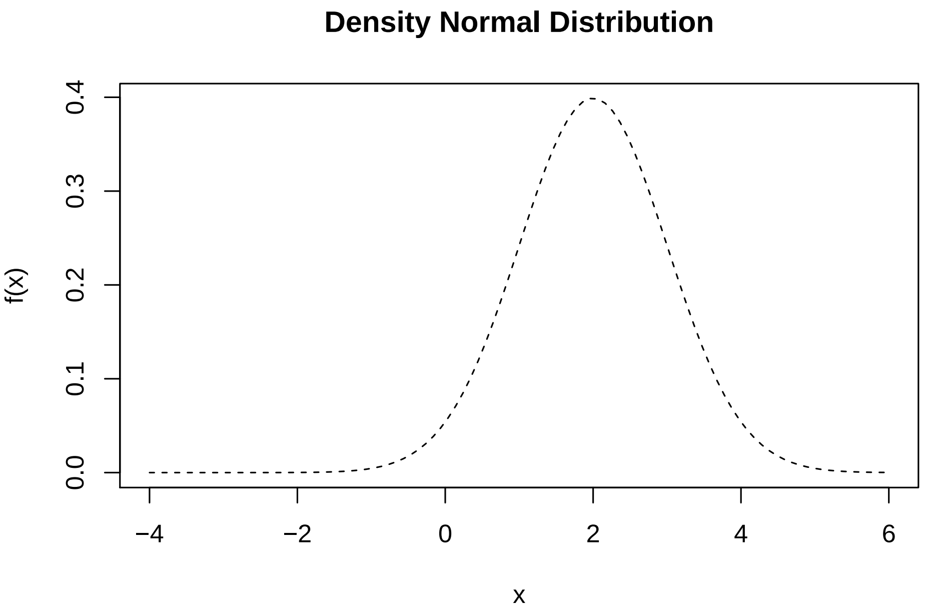
$$x \sim N(\mu, \sigma^2)$$

$$\mathrm{E}(x) = \mu, \mathrm{Var}(x) = \sigma^2$$

Density:

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

Plot of univariate normal density :
- Unimodal with peak at μ, the width is determined by σ (in this plot: $\mu = 2, \sigma = 1$)



Special case: **standard normal** with $\mu = 0$ and $\sigma^2 = 1$:

$$f(x|\mu = 0, \sigma^2 = 1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

Maximum entropy characterisation: the normal distribution is the unique distribution that has the highest (differential) entropy over all continuous distributions with support from minus infinity to plus infinity with a given mean and variance.

This is in fact one of the reasons why the normal distribution is so important (und useful) – if we only know that a random variable has a mean and variance, and not much else, then using the normal distribution will be a reasonable and well justified working assumption!

1.3.2 Multivariate normal model

$$\text{Dimension } d$$

$$\boldsymbol{x} \sim N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\boldsymbol{x} \sim \mathrm{MVN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$
$$\mathrm{E}(\boldsymbol{x}) = \boldsymbol{\mu}, \mathrm{Var}(\boldsymbol{x}) = \boldsymbol{\Sigma}$$

Density:

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{d}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \underbrace{\underbrace{(\boldsymbol{x}-\boldsymbol{\mu})^T}_{1\times d} \underbrace{\boldsymbol{\Sigma}^{-1}}_{d\times d} \underbrace{(\boldsymbol{x}-\boldsymbol{\mu})}_{d\times 1}}_{1\times 1=\text{scalar!}} \right)$$

- note that density contains precision matrix $\boldsymbol{\Sigma}^{-1}$
- inverting $\boldsymbol{\Sigma}$ implies inverting the eigenvalues λ_i of $\boldsymbol{\Sigma}$ (thus we need $\lambda_i > 0$)
- density also contains $\det(\boldsymbol{\Sigma}) = \prod_{i=1}^{d} \lambda_i \equiv$ product of eigenvalues of $\boldsymbol{\Sigma}$

Special case: **standard multivariate normal** with

$$\boldsymbol{\mu} = \boldsymbol{0}, \boldsymbol{\Sigma} = \boldsymbol{I} = \begin{pmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{pmatrix}$$

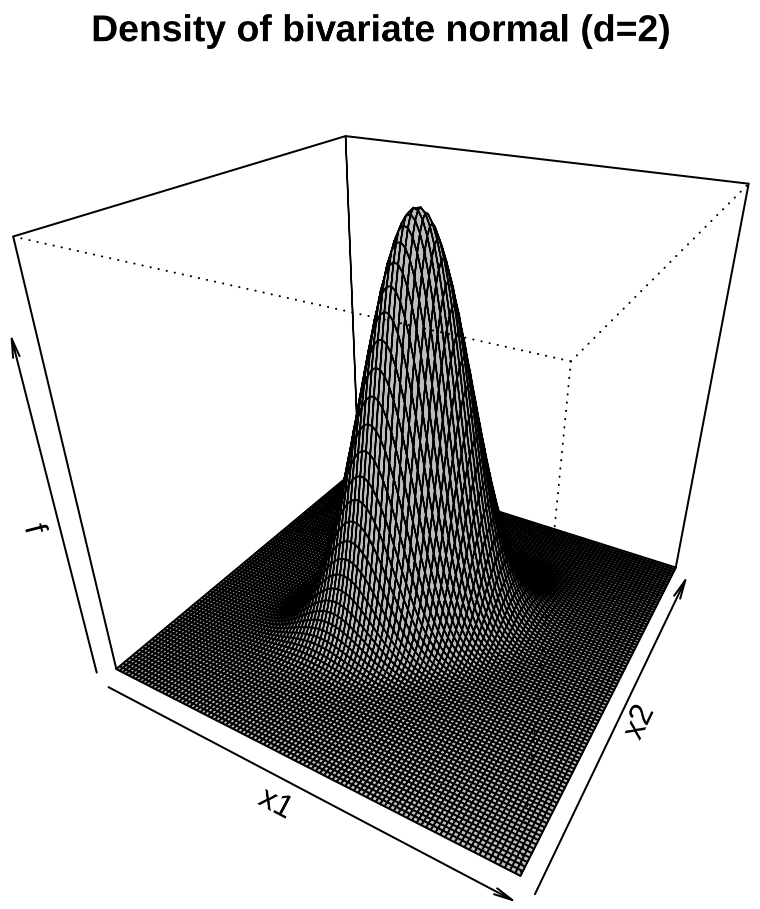
$$f(\boldsymbol{x}|\boldsymbol{\mu} = \boldsymbol{0}, \boldsymbol{\Sigma} = \boldsymbol{I}) = (2\pi)^{-d/2} \exp\left(-\frac{1}{2} \boldsymbol{x}^T \boldsymbol{x} \right) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x_i^2}{2} \right)$$

which is equivalent to the product of d univariate standard normals!

Misc:

- for $d = 1$, multivariate normal reduces to normal.
- for $\boldsymbol{\Sigma}$ diagonal (i.e. $\boldsymbol{P} = \boldsymbol{I}$, no correlation), MVN is the product of univariate normals (see Worksheet 2).

Plot of MVN density:



- Location: $\boldsymbol{\mu}$
- Shape: $\boldsymbol{\Sigma}$
- Unimodal: **one** peak
- Support from $-\infty$ to $+\infty$ in each dimension

1.3.3 Shape of the contour lines of the multivariate normal density

Now show that the contour lines of the multivariate normal density always take on the form of an ellipse, and that the radii of the ellipse is determined by the eigenvalues of $\boldsymbol{\Sigma}$.

We start by observing that a circle with radius r around the origin can be described as the set of points (x_1, x_2) satisfying $x_1^2 + x_2^2 = r^2$, or equivalently, $\frac{x_1^2}{r^2} + \frac{x_2^2}{r^2} = 1$. This is generalised to the shape of an ellipse by allowing (in two dimensions) for two radii r_1 and r_2 with $\frac{x_1^2}{r_1^2} + \frac{x_2^2}{r_2^2} = 1$, or in vector notation

$\boldsymbol{x}^T \text{Diag}(r_1^2, r_2^2)^{-1}\boldsymbol{x} = 1$. In d dimensions and allowing for rotation of the axes and a shift of the origin from 0 to $\boldsymbol{\mu}$ the condition for an ellipse is

$$(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{Q} \text{Diag}(r_1^2, \ldots, r_d^2)^{-1} \boldsymbol{Q}^T (\boldsymbol{x}-\boldsymbol{\mu}) = 1$$

where $\boldsymbol{Q}$ is an orthogonal rotation matrix.

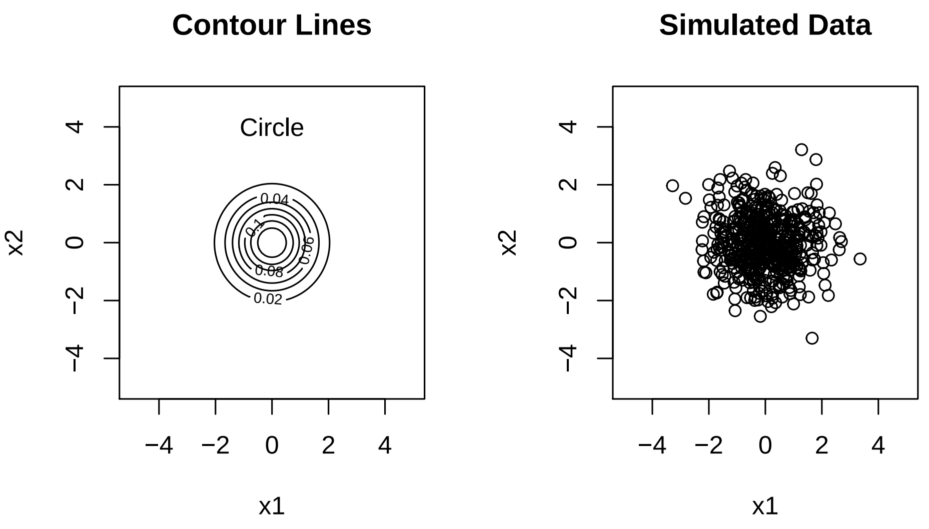
A contour line of a probability density function is a set of connected points where the density assumes the same constant value. In the case of the multivariate normal distribution keeping the density at some fixed value implies that $(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) = c$ where c is a constant. Using the eigenvalue decomposition of $\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ we can rewrite this condition as

$$(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{U}\boldsymbol{\Lambda}^{-1}\boldsymbol{U}^T (\boldsymbol{x}-\boldsymbol{\mu}) = c\,.$$

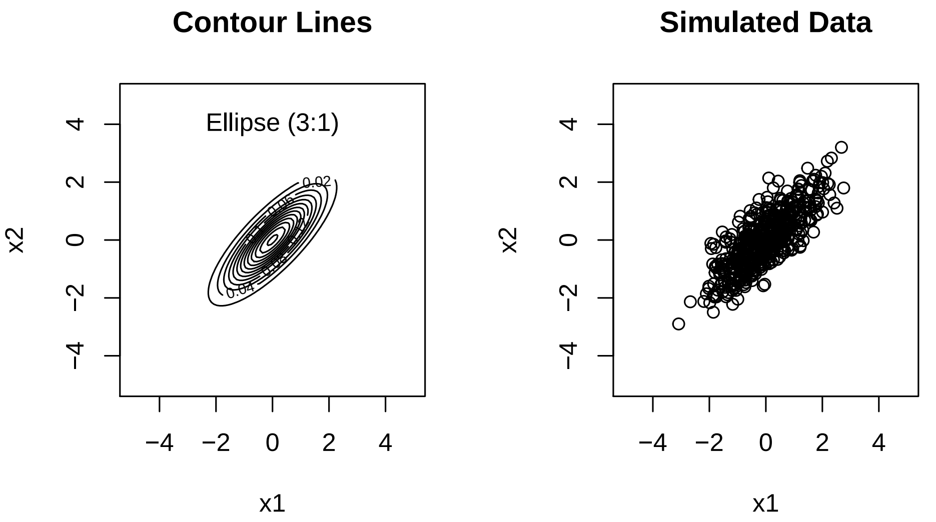
This implies that contour lines of the multivariate normal density are indeed ellipses and that the squared radii are proportional to the eigenvalues of $\boldsymbol{\Sigma}$. Equivalently, the positive square roots of the eigenvalues are proportional to the radii of the ellipse. Hence, for singular covariance matrix with one or more $\lambda_i = 0$ the corresponding radii are zero.

Two examples:

Case 1: No Correlation / Diagonal / Isotropic / Spherical Covariance $\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ with $\sqrt{\lambda_1/\lambda_2} = 1$:



Case 2: with correlation $\Sigma = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}$ with $\sqrt{\lambda_1/\lambda_2} = 3$:



Small λ_i indicates collinearity!

1.4 Estimation in large sample and small sample settings

In practical application of multivariate normal model we need to learn its parameters from data. We first consider the case when there are many measurements available, and then second the case when the number of data points is small compared to the number of parameters.

1.4.1 Data matrix

Observations from a multivariate normal are vectors:

$$\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n \overset{\text{iid}}{\sim} N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Data matrix (statistics convention):

Each *line* of the matrix is a transposed vector $\boldsymbol{x}_k^T$.

Thus:

$$\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)^T = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ x_{21} & x_{22} & \dots & x_{2d} \\ \vdots & & & \\ x_{n1} & x_{n2} & \dots & x_{nd} \end{pmatrix}$$

with

$$\boldsymbol{x}_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1d} \end{pmatrix}, \boldsymbol{x}_2 = \begin{pmatrix} x_{21} \\ \vdots \\ x_{2d} \end{pmatrix}, \dots, \boldsymbol{x}_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{nd} \end{pmatrix}$$

Thus, in statistics the first index runs over $(1, ..., n)$ and denotes the samples while the second index runs over $(1, ..., d)$ and refers to the variables.

The convention on data matrices that variables are in columns while samples are in rows. is *not* universal! In fact it's the **other way around in machine learning** (where samples are stored in columns and variables in rows). However, some machine learning books also follow the statistics convention.

1.4.2 Strategies for large sample estimation

1.4.2.1 Empirical estimators (outline)

For large n:

$$\underbrace{F}_{\text{true}} \approx \underbrace{\widehat{F}}_{\text{empirical}}$$

Replacing F by $\hat{F}$ leads to *empirical estimators*.

For example, the expectation can be approximated/estimated as follows:

$$\mathrm{E}_F(\boldsymbol{x}) \approx \mathrm{E}_{\widehat{F}}(\boldsymbol{x}) = \frac{1}{n} \sum_{k=1}^{n} \boldsymbol{x}_k$$

$$\mathrm{E}_F(g(\boldsymbol{x})) \approx \mathrm{E}_{\widehat{F}}(g(\boldsymbol{x})) = \frac{1}{n} \sum_{k=1}^{n} g(\boldsymbol{x}_k)$$

Recipe: to obtain an empirical plug-in estimator simply replace the expectation by the sample average in the population expression of the quantity of interest.

What does this work: the empirical distribution $\widehat{F}$ is actually the nonparametric maximum likelihood estimate of F (see below for likelihood estimation).

Note: the approximation of F by $\widehat{F}$ also the basis other approaches such as Efron's bootstrap method.

1.4.2.2 Maximum likelihood estimation (outline)

R.A. Fisher (1922): model-based estimators using the density or probability mass function

log-likelihood function:

$$\log L(\boldsymbol{\theta}) = \sum_{k=1}^{n} \underbrace{f}_{\text{density}} (\underbrace{x_i}_{\text{data}} \mid \underbrace{\boldsymbol{\theta}}_{\text{parameters}})$$

= conditional probability of the observed data given the model parameters

Maximum likelihood estimate:

$$\hat{\boldsymbol{\theta}}^{\text{ML}} = \arg\max_{\boldsymbol{\theta}} \log L(\boldsymbol{\theta})$$

ML finds the parameters that make the observed data most likely (it does *not* find the most probable model!)

The great appeal of **MLEs** is that they **are optimal for large n**, i.e. they use all the information available in the data optimally to estimate parameters, and **for large sample size no estimator can be constructed that outperforms the MLE!**

A further advantage of the method of maximum likelihood is that it does not only provide a point estimate but also the asymptotic error (via the Fisher information which is related to the curvature of the log-likelihood function).

1.4.3 Large sample estimates of mean μ and covariance Σ

1.4.3.1 Empirical estimates:

These can be written in three different ways:

Vector notation

$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{k=1}^{n} \boldsymbol{x}_k$$

$$\underbrace{\widehat{\boldsymbol{\Sigma}}}_{d\times d} = \frac{1}{n} \sum_{k=1}^{n} \underbrace{(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}})}_{d\times 1} \underbrace{(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}})^T}_{1\times d}$$

Component notation

$$\hat{\mu}_i = \frac{1}{n} \sum_{k=1}^{n} x_{ki}$$

$$\hat{\sigma}_{ij} = \frac{1}{n} \sum_{k=1}^{n} (x_{ki} - \hat{\mu}_i) \left(x_{kj} - \hat{\mu}_j \right)$$

$$\hat{\boldsymbol{\mu}} = \begin{pmatrix} \hat{\mu}_1 \\ \vdots \\ \hat{\mu}_d \end{pmatrix}, \widehat{\boldsymbol{\Sigma}} = (\hat{\sigma}_{ij})$$

Variance estimate:

$$\hat{\sigma}_{ii} = \frac{1}{n} \sum_{k=1}^{n} (x_{ki} - \hat{\mu}_i)^2$$

Note the factor $\frac{1}{n}$ (not $\frac{1}{n-1}$)

Data matrix notation

The empirical mean and covariance can also be written in terms of the data matrix $\boldsymbol{X}$. See Worksheet 2 for details.

1.4.3.2 Maximum likelihood estimates

It turns out that the empirical estimates are identical to the MLE assuming multivariate normal model:

$\Longrightarrow$ empirical estimates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$

$$\hat{\boldsymbol{\mu}}_{ML} = \hat{\boldsymbol{\mu}}_{\text{emp}}$$

$$\widehat{\boldsymbol{\Sigma}}_{ML} = \widehat{\boldsymbol{\Sigma}}_{\text{emp}}$$

For a direct derivation of the multivariate normal MLEs by optimising the multivariate normal log-likelihood function see the Worksheet 2 (easy for the mean, more difficult for the covariance matrix).

Note the factor $\frac{1}{n}$ in the MLE of the covariance matrix.

1.4.3.3 Distribution of the empirical / maximum likelihood estimates

With $\boldsymbol{x}_1, ..., \boldsymbol{x}_n \sim N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ one can find the exact distributions of the estimators.

1. Distribution of the estimate of the mean:

$$\hat{\boldsymbol{\mu}}_{ML} \sim N_d\left(\boldsymbol{\mu}, \frac{\boldsymbol{\Sigma}}{n}\right)$$

Since $\text{E}(\hat{\boldsymbol{\mu}}_{ML}) = \boldsymbol{\mu} \Longrightarrow \hat{\boldsymbol{\mu}}_{ML}$ is unbiased

2. Distribution of the covariance estimate:

$$\widehat{\boldsymbol{\Sigma}} \sim \text{Wishart}(\frac{\boldsymbol{\Sigma}}{n}, n-1)$$

Since $\text{E}(\widehat{\boldsymbol{\Sigma}}_{ML}) = \frac{n-1}{n}\boldsymbol{\Sigma} \Longrightarrow \widehat{\boldsymbol{\Sigma}}_{ML}$ is biased!

Easy to make unbiased: $\widehat{\boldsymbol{\Sigma}}_{UB} = \frac{n}{n-1}\widehat{\boldsymbol{\Sigma}} = \frac{1}{n-1}\sum_{k=1}^{n}(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}})^T$ is unbiased.

But unbiasedness is **not** a very relevant criteria in multivariate statistics!

1.4.4 Problems with maximum likelihood in small sample settings and high dimensions

Modern data is high dimensional!

Data sets with $n < d$, i.e. high dimension d and small sample size n are now common in many fields, e.g., medicine, biology but also finance and business analytics.

$$n = 100 \,\text{(e.g, patients/samples)}$$

$$d = 20000 \,\text{(e.g., genes/SNPs/proteins/variables)}$$

Reasons:

- the number of measured variables is increasing quickly with technological advances (e.g. genomics)
- but the number of samples cannot be similary increased (for cost and ethical reasons)

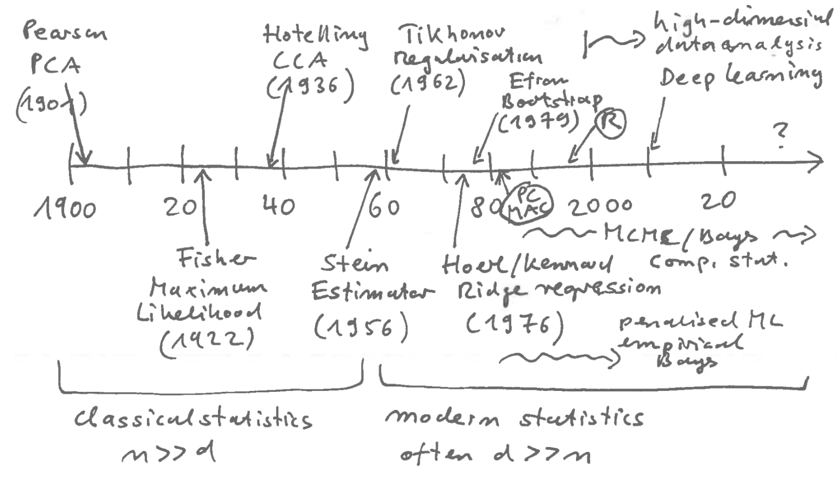
General problems of MLEs:

1. ML estimators are optimal only if **sample size is large** compared to the number of parameters. However, this optimality is not any more valid if sample size is moderate or smaller than the number of parameters.
2. If there is not enough data the **ML estimate overfits**. This means ML fits the current data perfectly but the resulting model does not generalise well (i.e. model will perform poorly in prediction)
3. If there is a choice between different models with different complexity **ML will always select the model with the largest number of parameters**.

-> for high-dimensional data with small sample size maximum likelihood estimation does not work!!!

History of Statistics:

Much of modern statistics (from 1960 onwards) is devoted to the development of inference and estimation techniques that work with complex, high-dimensional data.



- Maximum likelihood is a method from classical statistics (time up to about 1960).
- From 1960 modern (computational) statistics emerges, starting with **"Stein Paradox" (1956):** Charles Stein showed that in a **multivariate setting** ML estimators are **dominated by** (= are always worse than) shrinkage estimators!
- For example, there is a shrinkage estimator for the mean that is better (in terms of MSE) than the average (which is the MLE)!

Modern statistics has developed many different (but related) methods for use in high-dimensional, small sample settings:

- regularised estimators

- shrinkage estimators
- penalised maximum likelihood estimators
- Bayesian estimators
- Empirical Bayes estimators
- KL / entropy-based estimators

Most of this is out of scope for our class, but will be covered in advanced statistical courses.

Next, we describe a **simple regularised estimator for the estimation of the covariance** that we will use later (i.e. in classification).

1.4.5 Estimation of covariance matrix in small sample settings

Problems with ML estimate of Σ

1. Σ has O(d^2) number of parameters! $\Longrightarrow$ $\hat{\boldsymbol{\Sigma}}^{\text{MLE}}$ requires *a lot* of data! $n \gg d$ or d^2
2. if $n < d$ then $\hat{\boldsymbol{\Sigma}}$ is positive **semi**-definite (even if Σ is p.d.f.!) $\Longrightarrow$ $\hat{\boldsymbol{\Sigma}}$ will have **vanishing eigenvalues** (some $\lambda_i = 0$) and thus **cannot be inverted** and is singular!

Simple regularised estimate of Σ

Regularised estimator $\boldsymbol{S}^*$ = convex combination of $\boldsymbol{S} = \hat{\boldsymbol{\Sigma}}^{\text{MLE}}$ and $\boldsymbol{I}_d$ (identity matrix) to get

Regularisation:

$$\underbrace{\boldsymbol{S}^*}_{\text{regularised estimate}} = \underbrace{\lambda}_{\text{shrinkage intensity}} \underbrace{\boldsymbol{I}_d}_{\text{target}} + (1-\lambda) \underbrace{\boldsymbol{S}}_{\text{ML estimate}}$$

Next, choose $\lambda \in [0,1]$ such that $\boldsymbol{S}^*$ is better (in terms of MSE) than both $\boldsymbol{S}$ and $\boldsymbol{I}_d$.

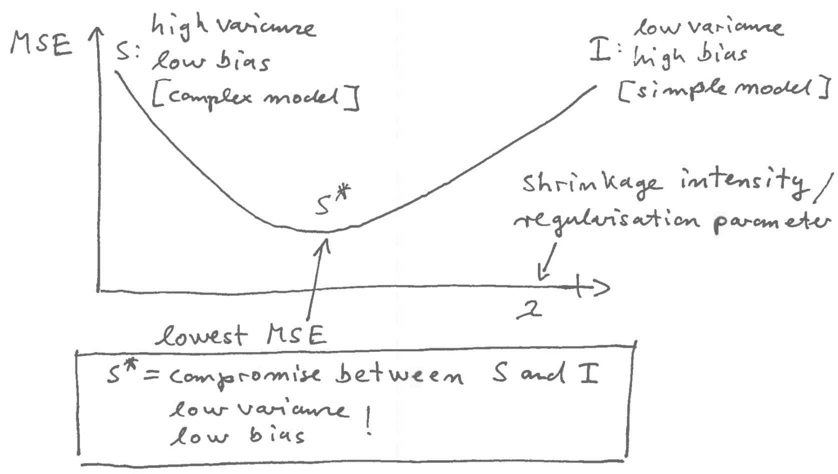
Bias-variance trade-off

MSE is the Mean Squared Error, composed of squared bias and variance.

$$\text{MSE}(\theta) = \text{E}((\hat{\theta} - \theta)^2) = \text{Bias}(\hat{\theta})^2 + \text{Var}(\hat{\theta})$$

with $\text{Bias}(\hat{\theta}) = \text{E}(\hat{\theta}) - \theta$

$\boldsymbol{S}$: ML estimate, many parameters, low bias, high variance
$\boldsymbol{I}_d$: "target", no parameters, high bias, low variance
$\Longrightarrow$ **reduce high variance of $\boldsymbol{S}$ by *introducing* a bit of bias through $\boldsymbol{I}_d$!**
$\Longrightarrow$ overall, MSE is decreased



How to find optimal shrinkage / regularisation parameter λ? Minimise MSE!

Challenge: since we don't know the true $\boldsymbol{\Sigma}$ we cannot actually compute the MSE directly but have to estimate it! How is this done in practise?

- by cross-validation (=resampling procedure)
- by using some analytic approximation (e.g. Stein's formula)

Why does regularisation of $\hat{\boldsymbol{\Sigma}}$ work?

1. $\boldsymbol{S}^*$ is **positive definite**:
   Matrix Theory:

$$\underbrace{\boldsymbol{M}_1}_{\text{positive definite, } \lambda \boldsymbol{I}} + \underbrace{\boldsymbol{M}_2}_{\text{positive semi-definite, } (1-\lambda)\boldsymbol{S}} = \underbrace{\boldsymbol{M}_3}_{\text{positive definite, } \boldsymbol{S}^*}$$

   $\Longrightarrow \boldsymbol{S}^*$ can be inverted even if $n < d$

2. It's **Bayesian** in disguise!

$$\underbrace{\boldsymbol{S}^*}_{\text{posterior mean}} = \underbrace{\lambda \boldsymbol{I}_d}_{\text{prior information}} + (1-\lambda) \underbrace{\boldsymbol{S}}_{\text{data summarised by maximum likelihood}}$$

   - Prior information helps to infer $\boldsymbol{\Sigma}$ even in small samples
   - Since λ is chosen from data, it is actually an empirical Bayes.
   - also called shrinkage estimator since the off-diagonal entries are shrunk towards zero
   - this type of linear shrinkage/regularisation is natural for models in the exponential family (Diaconis and Ylvisaker, 1979)

In Worksheet 2 the empirical estimator of covariance is compared with the covariance estimator implemented in the R package "corpcor". This uses a regularisation similar as above (but for the correlation rather than covariance matrix) and it employs an analytic data-adaptive estimate of the shrinkage intensity λ. This estimator is a variant of an empirical Bayes / James-Stein estimator (see the year 2 Statistical Methods 20802 module).

Summary

- In multivariate statistics, it is useful (and often necessary) to utilise prior information!
- Regularisation introduces bias and reduces variance, minimising overall MSE
- Unbiased estimation (a highly valued property in classical statistics!) is not a good idea in multivariate settings and often leads to poor estimators.

1.5 Discrete multivariate distributions

Most univariate distributions have multivariate counterparts. Here are some of the most important ones. First we discuss discrete distributions, then later continuous distributions.

1.5.1 Categorical distribution

1.5.1.1 Univariate case

Bernoulli distribution (=coin tossing model)

$$x \sim \text{Ber}(\pi)$$
$$x \in \{0,1\}$$
$$\pi \in [0,1]$$
$$\Pr(x=1) = \pi$$
$$\Pr(x=0) = 1-\pi$$
$$\text{E}(x) = \pi$$
$$\text{Var}(x) = \pi(1-\pi)$$

1.5.1.2 Multivariate case

$$\boldsymbol{x} \sim \text{Categ}(\boldsymbol{\pi})$$
$$\boldsymbol{x} = (x_1, ..., x_d)^T;\ x_i \in \{0,1\};\ \sum_{i=1}^{d} x_i = 1$$
$$\boldsymbol{\pi} = (\pi_1, ..., \pi_d)^T;\ \sum_{i=1}^{d} \pi_i = 1$$
$$\Pr(x_i=1) = \pi_i$$
$$\Pr(x_i=0) = 1-\pi_i$$

$$\text{E}(\boldsymbol{x}) = \boldsymbol{\pi}$$
$$\text{Var}(x_i) = \pi_i(1-\pi_i)$$
$$\text{Cov}(x_i, x_j) = -\pi_i\pi_j$$

1.5.2 Multinomial distribution

1.5.2.1 Univariate case

Binomial Distribution

Repeat Bernoulli experiment r times

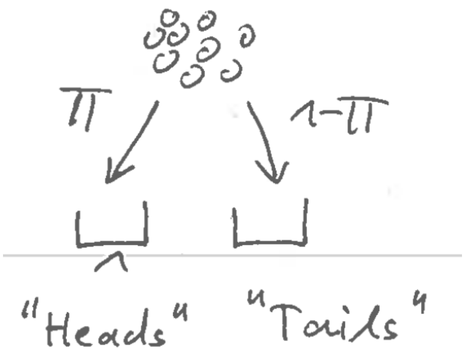
$$x \sim \text{Binom}(\pi, r)$$
$$x \in \{0, ..., r\}$$
$$\text{E}(x) = r\,\pi$$
$$\text{Var}(x) = r\,\pi(1-\pi)$$

Standardised to unit interval:

$$\frac{x}{r} \in \left\{0, \frac{1}{r}, ..., 1\right\}$$
$$\text{E}\left(\frac{x}{r}\right) = \pi$$
$$\text{Var}\left(\frac{x}{r}\right) = \frac{\pi(1-\pi)}{r}$$

Urn model:

distribute r balls into two bins



1.5.2.2 Multivariate case

Multinomial distribution

Draw r times from categorical distribution

$$\boldsymbol{x} \sim Multinom(\boldsymbol{\pi}, r)$$
$$x_i \in \{0, 1, ..., r\};\ \sum_{i=1}^{d} x_i = r$$

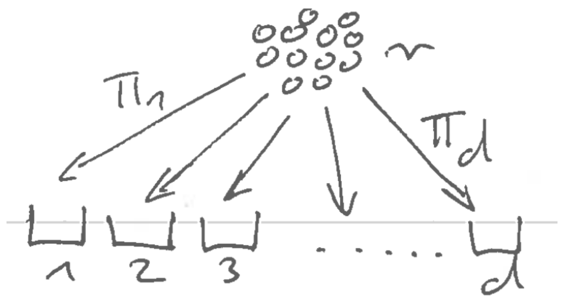
$$
\begin{aligned}
\mathrm{E}(\boldsymbol{x}) &= r\,\boldsymbol{\pi} \\
\mathrm{Var}(x_i) &= r\,\pi_i(1-\pi_i) \\
\mathrm{Cov}(x_i, x_j) &= -r\,\pi_i\pi_j
\end{aligned}
$$

Standardised to unit interval:

$$
\begin{aligned}
\frac{x_i}{r} &\in \left\{0, \frac{1}{r}, \frac{2}{r}, ..., 1\right\} \\
\mathrm{E}\left(\frac{\boldsymbol{x}}{r}\right) &= \boldsymbol{\pi} \\
\mathrm{Var}\left(\frac{x_i}{r}\right) &= \frac{\pi_i(1-\pi_i)}{r} \\
\mathrm{Cov}\left(\frac{x_i}{r}, \frac{x_j}{r}\right) &= -\frac{\pi_i\pi_j}{r}
\end{aligned}
$$

Urn model:

distribute r balls into d bins



1.6 Continuous multivariate distributions

1.6.1 Dirichlet distribution

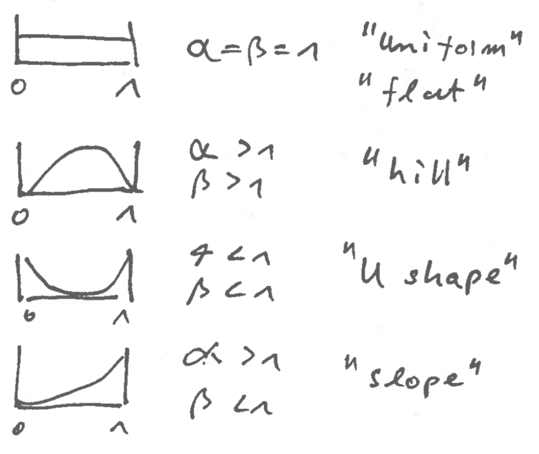
1.6.1.1 Univariate case

Beta distribution

$$
\begin{aligned}
x &\sim \mathrm{Beta}(\alpha, \beta) \\
x &\in [0, 1] \\
\alpha > 0&; \beta > 0 \\
m &= \alpha + \beta \\
\mu &= \frac{\alpha}{m} \in [0, 1] \\
\mathrm{E}(x) &= \mu \\
\mathrm{Var}(x) &= \frac{\mu(1-\mu)}{m+1}
\end{aligned}
$$

compare with unit standardised binomial!

Different shapes



Useful as distribution for a proportion π

Bayesian Model:

Beta prior: $\pi \sim Beta(\alpha, \beta)$
Binomial likelihood: $x|\pi \sim Binom$

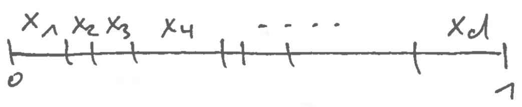
1.6.1.2 Multivariate case

Dirichlet distribution

$$\boldsymbol{x} \sim \text{Dirichlet}(\boldsymbol{\alpha})$$
$$x_i \in [0,1]; \sum_{i=1}^{d} x_i = 1$$
$$\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_d)^T > 0$$
$$m = \sum_{i=1}^{d} \alpha_i$$
$$\mu_i = \frac{\alpha_i}{m} \in [0,1]$$
$$\text{E}(x_i) = \mu_i$$
$$\text{Var}(x_i) = \frac{\mu_i(1-\mu_i)}{m+1}$$
$$\text{Cov}(x_i, x_j) = -\frac{\mu_i \mu_j}{m+1}$$

compare with unit standardised multinomial!

Stick breaking" model



Useful as distribution for a proportion $\boldsymbol{\pi}$

Bayesian Model:

Dirichlet prior: $\boldsymbol{\pi} \sim Dirichlet(\boldsymbol{\alpha})$

Multinomial likelihood: $\boldsymbol{x}|\boldsymbol{\pi} \sim Multinom$

1.6.2 Wishart distribution

This multivariate distribution generalises the univariate scaled χ^2 distribution (also known as Gamma distribution).

1.6.2.1 Univariate case

Scaled χ^2 distribution (=Gamma distribution, see below)

$$z_1, z_2, \ldots, z_m \overset{\text{iid}}{\sim} N(0, \sigma^2)$$
$$x = \sum_{i=1}^{m} z_i^2$$

$$x \sim \sigma^2 \chi^2_m = \text{W}_1(\sigma^2, m)$$
$$\text{E}(x) = m\,\sigma^2$$
$$\text{Var}(x) = m\,2\sigma^4$$

Useful as the distribution of sample variance:

$$y_1, \ldots, y_n \sim N(\mu, \sigma^2)$$

Known mean μ:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu)^2 \sim \text{W}_1\left(\frac{\sigma^2}{n}, n\right)$$

Unknown mean μ (estimated by $\bar{y}$):

$$\widehat{\sigma^2}_{ML} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 \sim \text{W}_1\left(\frac{\sigma^2}{n}, n-1\right)$$

$$\widehat{\sigma^2}_{UB} = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2 \sim \text{W}_1\left(\frac{\sigma^2}{n-1}, n-1\right)$$

1.6.2.2 Multivariate case

Wishart distribution

$$z_1, z_2, \ldots, z_m \overset{\text{iid}}{\sim} N_d(0, \boldsymbol{\Sigma})$$
$$\underbrace{\boldsymbol{X}}_{d\times d} = \sum_{i=1}^{m} \underbrace{z_i z_i^T}_{d\times d}$$

Note that $\boldsymbol{X}$ is a *matrix*!

$$\boldsymbol{X} \sim \mathrm{W}_d\left(\boldsymbol{\Sigma}, m\right)$$
$$\mathrm{E}(\boldsymbol{X}) = m\boldsymbol{\Sigma}$$
$$\mathrm{Var}(x_{ij}) = m\left(\sigma_{ij}^2 + \sigma_{ii}\sigma_{jj}\right)$$

Useful as distribution of sample covariance:

$$\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n \sim N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$
$$\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{y}_i - \boldsymbol{\mu})(\boldsymbol{y}_i - \boldsymbol{\mu})^T \sim \mathrm{W}_d\left(\boldsymbol{\Sigma}/n, n\right)$$
$$\widehat{\boldsymbol{\Sigma}}_{ML} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{y}_i - \bar{\boldsymbol{y}})(\boldsymbol{y}_i - \bar{\boldsymbol{y}})^T \sim \mathrm{W}_d\left(\boldsymbol{\Sigma}/n, n-1\right)$$
$$\widehat{\boldsymbol{\Sigma}}_{UB} = \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{y}_i - \bar{\boldsymbol{y}})(\boldsymbol{y}_i - \bar{\boldsymbol{y}})^T \sim \mathrm{W}_d\left(\boldsymbol{\Sigma}/(n-1), n-1\right)$$

1.6.2.3 Relationship to Gamma distribution

The scaled χ^2 distribution (=one-dimensional Wishart distribution) with parameters σ^2 and m is in fact a reparameterised **Gamma distribution** with shape parameter α and scale parameter β:

$$\text{Gamma}\left(\underbrace{\frac{m}{2}}_{\text{shape}}, \underbrace{2\sigma^2}_{\text{scale}}\right) = \sigma^2\chi_m^2 = \mathrm{W}_1(\sigma^2, m)$$

or, equivalently ($m = 2\alpha$, $\sigma^2 = \beta/2$)

$$\text{Gamma}\left(\underbrace{\alpha}_{\text{shape}}, \underbrace{\beta}_{\text{scale}}\right) = \frac{\beta}{2}\chi_{2\alpha}^2 = \mathrm{W}_1(\frac{\beta}{2}, 2\alpha)$$

The mean of the Gamma distribution is $\mathrm{E}(x) = \alpha\beta = \mu$ and the variance is $\mathrm{Var}(x) = \alpha\beta^2 = \mu^2/\alpha$.

The **exponential distribution** with rate parameter λ is a special case of the Gamma distribution with $\alpha = 1$:

$$\text{Exp}(\lambda) = \text{Gamma}(1, \frac{1}{\lambda}) = \frac{1}{2\lambda}\chi^2_2 = \text{W}_1(\frac{1}{2\lambda}, 2)$$

The corresponding mean is $1/\lambda = \mu$ and variance $1/\lambda^2 = \mu^2$.

1.6.3 Inverse Wishart distribution

1.6.3.1 Univariate case

Inverse χ^2 Distribution

$$x \sim \text{W}_1^{-1}(\psi, k+2) = \psi \ \text{Inv-}\chi^2_{k+2}$$
$$\text{E}(x) = \frac{\psi}{k}$$
$$\text{Var}(x) = \frac{2\psi^2}{k^2(k-2)}$$

Relationship to scaled χ^2 :

$$\frac{1}{x} \sim W_1(\psi^{-1}, k+2) = \psi^{-1}\, \chi^2_{k+2}$$

1.6.3.2 Multivariate case

Inverse Wishart distribution

$$\underbrace{\boldsymbol{X}}_{d\times d} \sim \text{W}_d^{-1}\left(\underbrace{\boldsymbol{\Psi}}_{d\times d}, k+d+1\right)$$
$$\text{E}(\boldsymbol{X}) = \boldsymbol{\Psi}/k$$
$$\text{Var}(x_{ij}) = \frac{2}{k^2(k-2)}\frac{(k+2)\psi_{ij} + k\,\psi_{ii}\psi_{jj}}{2k+2}$$

Relationship to Wishart:

$$\boldsymbol{X}^{-1} \sim \text{W}_d\left(\boldsymbol{\Psi}^{-1}, k+d+1\right)$$

1.6.3.3 Relationship to inverse Gamma distribution

Another way to express the univariate inverse Wishart distribution is via the **inverse Gamma distribution**:

$$IG(\underbrace{1+\frac{k}{2}}_{\text{shape } \alpha}, \underbrace{\frac{\psi}{2}}_{\text{scale } \beta}) = \psi \ \text{Inv-}\chi^2_{k+2} = \text{W}_1^{-1}(\psi, k+2)$$

or equivalently ($k = 2(\alpha - 1)$ and $\psi = 2\beta$)

$$IG(\alpha, \beta) = 2\beta \text{ Inv-}\chi^2_{2\alpha} = \mathrm{W}_1^{-1}(2\beta, 2\alpha)$$

The mean of the inverse Gamma distribution is $\mathrm{E}(x) = \frac{\beta}{\alpha-1} = \mu$ the variance $\mathrm{Var}(x) = \frac{\beta^2}{(\alpha-1)^2(\alpha-2)} = \frac{2\mu^2}{k-2}$.

The inverse of x is Gamma distributed:

$$\frac{1}{x} \sim \text{Gamma}(1 + \frac{k}{2}, 2\psi^{-1}) = \text{Gamma}(\alpha, \beta^{-1})$$

The inverse Wishart distribution is useful as conjugate distribution for Bayesian modelling of the variance, with k the sample size parameter and $\Psi = k\Sigma$ (or $\psi = k\sigma^2$).

1.6.4 Further distributions

https://en.wikipedia.org/wiki/List_of_probability_distributions

Wikipedia is a quite good source for information on distributions!

Chapter 2

Transformations and dimension reduction

Motivation: In the following we study transformations of random vectors and their distributions. These transformation are very important since they either transform simple distributions into more complex distributions or allow to simplify complex models. In machine learning invertible mappings of transformations for probability distributions are known as "normalising flows" (these play a key role e.g. in neural networks).

2.1 Linear Transformations

2.1.1 Location-scale transformation

Also known as affine transformation.

$$\boldsymbol{y} = \underbrace{\boldsymbol{a}}_{\text{location parameter}} + \underbrace{\boldsymbol{B}}_{\text{scale parameter}} \boldsymbol{x}$$

$$\boldsymbol{y} : m \times 1 \text{ random vector}$$

$$\boldsymbol{a} : m \times 1 \text{ vector, location parameter}$$

$$\boldsymbol{B} : m \times d \text{ matrix, scale parameter }, m \geq 1$$

$$\boldsymbol{x} : d \times 1 \text{ random vector}$$

$$\begin{array}{l} \text{E}(\boldsymbol{x}) = \boldsymbol{\mu} \\ \text{Var}(\boldsymbol{x}) = \boldsymbol{\Sigma} \end{array} \Longrightarrow \begin{array}{l} \text{E}(\boldsymbol{y}) = \boldsymbol{a} + \boldsymbol{B}\boldsymbol{\mu} \\ \text{Var}(\boldsymbol{y}) = \boldsymbol{B}\boldsymbol{\Sigma}\boldsymbol{B}^T \end{array}$$

Special cases/examples:

1. Univariate case ($d = 1, m = 1$)

   - $\text{E}(y) = a + b\mu$
   - $\text{Var}(y) = b^2\sigma^2$
2. Sum of two random univariate variables
   $y = x_1 + x_2$, i.e. $a = 0$ and $\boldsymbol{B} = (1, 1)$
   - $\text{E}(x_1 + x_2) = \mu_1 + \mu_2$
   - $\text{Var}(x_1 + x_2) = (1, 1) \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \sigma_1^2 + \sigma_2^2 + 2\sigma_{12} = \text{Var}(x_1) + \text{Var}(x_2) + 2\text{Cov}(x_1, x_2)$
3. $y_1 = a_1 + b_1x_1$ and $y_2 = a_2 + b_2x_2$, i.e. $\boldsymbol{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$ and $\boldsymbol{B} = \begin{pmatrix} b_1 & 0 \\ 0 & b_2 \end{pmatrix}$
   - $\text{E}(\boldsymbol{y}) = \begin{pmatrix} a_1 + b_1\mu_1 \\ a_2 + b_2\mu_2 \end{pmatrix}$
   - $\text{Var}(\boldsymbol{y}) = \begin{pmatrix} b_1 & 0 \\ 0 & b_2 \end{pmatrix} \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{pmatrix} \begin{pmatrix} b_1 & 0 \\ 0 & b_2 \end{pmatrix} = \begin{pmatrix} b_1^2\sigma_1^2 & b_1b_2\sigma_{12} \\ b_1b_2\sigma_{21} & b_2^2\sigma_2^2 \end{pmatrix}$
     i.e. $\text{Cov}(a_1 + b_1x_1, a_2 + b_2x_2) = b_1b_2\text{Cov}(x_1, x_2)$

2.1.2 Invertible location-scale transformation

If $m = d$ and $\det(\boldsymbol{B}) \neq 0$ then we get an **invertible** transformation:

$$\boldsymbol{y} = \boldsymbol{a} + \boldsymbol{B}\boldsymbol{x}$$

$$\boldsymbol{x} = \boldsymbol{B}^{-1}(\boldsymbol{y} - \boldsymbol{a})$$

Transformation of density: $\boldsymbol{x} \sim F_{\boldsymbol{x}}$ with density $f_{\boldsymbol{x}}(\boldsymbol{x})$

$\Longrightarrow \boldsymbol{y} \sim F_{\boldsymbol{y}}$ with density

$$f_{\boldsymbol{y}}(\boldsymbol{y}) = |\det(\boldsymbol{B})|^{-1} f_{\boldsymbol{x}}\left(\boldsymbol{B}^{-1}(\boldsymbol{y} - \boldsymbol{a})\right)$$

Example 2.1. Mahalanobis transform $\boldsymbol{y} = \boldsymbol{\Sigma}^{-1/2}(\boldsymbol{x} - \boldsymbol{\mu})$

We assume a positive definite and thus invertible $\boldsymbol{\Sigma}$, so that the inverse principal matrix square root $\boldsymbol{\Sigma}^{-1/2}$ can be computed, and the transformation itself is invertible.

$$\boldsymbol{a} = -\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\mu}$$

$$\boldsymbol{B} = \boldsymbol{\Sigma}^{-1/2}$$

$$\text{E}(\boldsymbol{x}) = \boldsymbol{\mu} \text{ and } \text{Var}(\boldsymbol{x}) = \boldsymbol{\Sigma}$$

$$\Longrightarrow \text{E}(\boldsymbol{y}) = \boldsymbol{0} \text{ and } \text{Var}(\boldsymbol{y}) = \boldsymbol{I}_d$$

Mahalanobis transformation performs three functions:

1. Centering $(-\boldsymbol{\mu})$
2. Standardisation $\text{Var}(y_i) = 1$
3. Decorrelation $\text{Cor}(y_i, y_j) = 0$ for $i \neq j$

Univariate case ($d = 1$)

$$y = \frac{x-\mu}{\sigma}$$

= centering + standardisation

The **Mahalanobis transformation** appears implicitly in many places in multivariate statistics, e.g. in the multivariate normal density.

It is a particular example of a whitening transformation (of which there are infinitely many, see later chapters).

Example 2.2. Inverse Mahalanobis transformation $\boldsymbol{y} = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\boldsymbol{x}$

This transformation is the **inverse** of the Mahalanobis transformation. As the Mahalanobis transform is a particular whitening transform the inverse transform is sometimes called the Mahalanobis colouring transformation.

$$\boldsymbol{a} = \boldsymbol{\mu}$$

$$\boldsymbol{B} = \boldsymbol{\Sigma}^{1/2}$$

$$\text{E}(\boldsymbol{x}) = \boldsymbol{0} \text{ and } \text{Var}(\boldsymbol{x}) = \boldsymbol{I}_d$$

$$\Longrightarrow \text{E}(\boldsymbol{y}) = \boldsymbol{\mu} \text{ and } \text{Var}(\boldsymbol{y}) = \boldsymbol{\Sigma}$$

Assume $\boldsymbol{x}$ is multivariate standard normal $\boldsymbol{x} \sim N_d(\boldsymbol{0}, \boldsymbol{I}_d)$ with density

$$f_{\boldsymbol{x}}(\boldsymbol{x}) = (2\pi)^{-d/2} \exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{x}\right)$$

Then the density after applying this inverse Mahalanobis transform is

$$f_{\boldsymbol{y}}(\boldsymbol{y}) = |\det(\boldsymbol{\Sigma}^{1/2})|^{-1}(2\pi)^{-d/2} \exp\left(-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\Sigma}^{-1/2}(\boldsymbol{y}-\boldsymbol{\mu})\right)$$

$$= (2\pi)^{-d/2} \det(\boldsymbol{\Sigma})^{-1/2} \exp\left(-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\boldsymbol{\mu})\right)$$

$\Longrightarrow \boldsymbol{y}$ has multivariate normal density!!

Application: e.g. random number generation: draw from $N_d(\boldsymbol{0}, \boldsymbol{I}_d)$ (easy!) then convert to multivariate normal by tranformation.

2.2 Nonlinear transformations

2.2.1 General transformation

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x})$$

with $\boldsymbol{h}$ an arbitrary vector-valued function

- linear case: $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{a} + \boldsymbol{B}\boldsymbol{x}$

$$\begin{aligned}
&\text{E}(\boldsymbol{x}) = \boldsymbol{\mu}\\
&\text{Var}(\boldsymbol{x}) = \boldsymbol{\Sigma}\\
&\text{E}(\boldsymbol{y}) = \text{E}(\boldsymbol{h}(\boldsymbol{x})) = ?\\
&\text{Var}(\boldsymbol{y}) = \text{Var}(\boldsymbol{h}(\boldsymbol{x})) = ?
\end{aligned}$$

For a general transformation $\boldsymbol{h}(\boldsymbol{x})$ the mean and variance of the transformed variable cannot be easily or analytically calculated.

However, we can find a **linear approximation** and then compute the mean and variance. This is called the "Delta Method".

2.2.2 Linearisation of $\boldsymbol{h}(\boldsymbol{x})$

Taylor series approximation (first order) of $\boldsymbol{h}(\boldsymbol{x})$ around $\boldsymbol{x}_0$:

$$\boldsymbol{h}(\boldsymbol{x}) \approx \boldsymbol{h}(\boldsymbol{x}_0) + \underbrace{\boldsymbol{J}_{\boldsymbol{h}}(\boldsymbol{x}_0)}_{\text{Jacobi matrix}} (\boldsymbol{x} - \boldsymbol{x}_0) = \underbrace{\boldsymbol{h}(\boldsymbol{x}_0) - \boldsymbol{J}_{\boldsymbol{h}}(\boldsymbol{x}_0)\,\boldsymbol{x}_0}_{\boldsymbol{a}} + \underbrace{\boldsymbol{J}_{\boldsymbol{h}}(\boldsymbol{x}_0)}_{\boldsymbol{B}}\,\boldsymbol{x}$$

∇, the nabla operator, is the row vector $(\frac{\partial}{\partial x_1}, ..., \frac{\partial}{\partial x_d})$, which when applied to univariate h gives the gradient:

$$\nabla h(\boldsymbol{x}) = \left(\frac{\partial h}{\partial x_1}, ..., \frac{\partial h}{\partial x_d}\right)$$

The **Jacobi matrix** is the **generalisation of gradient** if $\boldsymbol{h}$ is vector-valued:

$$\boldsymbol{J}_{\boldsymbol{h}}(\boldsymbol{x}) = \begin{pmatrix} \nabla h_1(\boldsymbol{x}) \\ \nabla h_2(\boldsymbol{x}) \\ \vdots \\ \nabla h_m(\boldsymbol{x}) \end{pmatrix} = \begin{pmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial x_1} & \cdots & \frac{\partial h_m}{\partial x_d} \end{pmatrix}$$

2.2.3 Delta method

First order approximation of $\boldsymbol{h}(\boldsymbol{x})$ around $\boldsymbol{x}_0 = \boldsymbol{\mu}$ yields $\boldsymbol{a} = \boldsymbol{h}(\boldsymbol{\mu}) - \boldsymbol{J}_{\boldsymbol{h}}(\boldsymbol{\mu})\,\boldsymbol{\mu}$ $\boldsymbol{B} = \boldsymbol{J}_{\boldsymbol{h}}(\boldsymbol{\mu})$ and

$$\text{E}(\boldsymbol{y}) \approx \boldsymbol{h}(\boldsymbol{\mu})$$

$$\text{Var}(\boldsymbol{y}) \approx \boldsymbol{J}_h(\boldsymbol{\mu})\, \boldsymbol{\Sigma}\, \boldsymbol{J}_h(\boldsymbol{\mu})^T$$

Special case: univariate Delta method

$$\text{E}(y) \approx h(\mu)$$

$$\text{Var}(y) \approx \sigma^2 h'(\mu)^2$$

2.2.4 Transformation of densities under general invertible transformation

Assume $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{y}(\boldsymbol{x})$ is invertible: $\boldsymbol{h}^{-1}(\boldsymbol{y}) = \boldsymbol{x}(\boldsymbol{y})$

$\boldsymbol{x} \sim F_{\boldsymbol{x}}$ with probability density function $f_{\boldsymbol{x}}(\boldsymbol{x})$

The density $f_{\boldsymbol{y}}(\boldsymbol{y})$ of the transformed random vector $\boldsymbol{y}$ is then given by

$$f_{\boldsymbol{y}}(\boldsymbol{y}) = |\det(\boldsymbol{J}_{\boldsymbol{x}}(\boldsymbol{y}))|\ f_{\boldsymbol{x}}(\boldsymbol{x}(\boldsymbol{y}))$$

where $\boldsymbol{J}_{\boldsymbol{x}}(\boldsymbol{y})$ is the Jacobi matrix of the inverse transformation

Special cases:

- Univariate version: $f_y(y) = |\frac{dx}{dy}|\, f_x(x(y))$
- Linear transformation $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{a} + \boldsymbol{B}\boldsymbol{x}$, with $\boldsymbol{x}(\boldsymbol{y}) = \boldsymbol{B}^{-1}(\boldsymbol{y} - \boldsymbol{a})$ and $\boldsymbol{J}_{\boldsymbol{x}}(\boldsymbol{y}) = \boldsymbol{B}^{-1}$:
  $f_{\boldsymbol{y}}(\boldsymbol{y}) = |\det(\boldsymbol{B})|^{-1} f_{\boldsymbol{x}}\left(\boldsymbol{B}^{-1}(\boldsymbol{y} - \boldsymbol{a})\right)$

2.2.5 Normalising flows

In machine learning (sequences of) invertible nonlinear transformations are known as "normalising flows". They are used both in a generative way (building complex models from simple models) and also in a simplification and dimension reduction context.

In this module we will focus mostly on linear transformations as these underpin much of classical multivariate statistics, but it is important to keep in mind for later study the importance of nonlinear transformations —see, e.g, the review paper by Kobyzev et al. "Normalizing Flows: Introduction and Ideas", available from https://arxiv.org/abs/1908.09257 .

2.3 Whitening transformations

2.3.1 Overview

The *Mahalanobis* transform (also know as "zero-phase component analysis" or short ZCA transform in machine learning) is a specific example of a **whitening transformation**. These constitute an important and widely used class of invertible location-scale transformations.

Terminology: whitening refers to the fact that after the transformation the covariance matrix is spherical, isotrop, white ($\boldsymbol{I}_d$)

Whitening is **useful in preprocessing**, to **turn multivariate problems into simple univariate models** and some **reduce the dimension in an optimal way**.

In so-called latent variable models whitening procedures link observed and latent variables (which usually are uncorrelated and standardised random variables):

$$\begin{array}{ccl} \text{Whitening} & \boldsymbol{x} & \text{Observed external variable (can be measured), typically correlated} \\ \downarrow & \uparrow & \\ & \boldsymbol{z} & \text{Unobserved "latent" variable internal, typically not correlated} \end{array}$$

2.3.2 General whitening transformation

$\boldsymbol{x}$ random vector $\sim F_{\boldsymbol{x}}$ (not necessarily from multivariate normal)

$$\underbrace{\boldsymbol{z}}_{d\times 1 \text{ vector}} = \underbrace{\boldsymbol{W}}_{d\times d \text{ whitening matrix}} \underbrace{\boldsymbol{x}}_{d\times 1 \text{ vector}}$$

Objective: choose $\boldsymbol{W}$ so that $\text{Var}(\boldsymbol{z}) = \boldsymbol{I}_d$

Note: we do not care about $\text{E}(\boldsymbol{z})$ since we can always centre!

For Mahalanobis/ZCA whitening we already know that $\boldsymbol{W}^{\text{ZCA}} = \boldsymbol{\Sigma}^{-1/2}$.

In general, $\boldsymbol{W}$ needs to satisfy a constraint:

$$\begin{array}{lll} & \text{Var}(\boldsymbol{z}) & = \boldsymbol{I}_d \\ \Longrightarrow & \text{Var}(\boldsymbol{W}\boldsymbol{x}) & = \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{W}^T = \boldsymbol{I}_d \\ \Longrightarrow & \boldsymbol{W}\,\boldsymbol{\Sigma}\,\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{W} & \end{array}$$

$$\Longrightarrow \text{constraint on whitening matrix: } \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{\Sigma}^{-1}$$

Clearly, the ZCA whitening matrix satisfies this constraint: $(\boldsymbol{W}^{ZCA})^T\boldsymbol{W}^{ZCA} = \boldsymbol{\Sigma}^{-1/2}\boldsymbol{\Sigma}^{-1/2} = \boldsymbol{\Sigma}^{-1}$

2.3.3 General solution of whitening constraint (covariance-based)

$$\boldsymbol{W} = \boldsymbol{Q}_1\boldsymbol{\Sigma}^{-1/2}$$

where $\boldsymbol{Q}_1$ is a rotation (orthogonal) matrix (with properties $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I}$ and $\boldsymbol{Q}^T = \boldsymbol{Q}^{-1}$).

Easy to see that this $\boldsymbol{W}$ satisfies the whitening constraint:

$$\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{\Sigma}^{-1/2} \underbrace{\boldsymbol{Q}_1^T\boldsymbol{Q}_1}_{\boldsymbol{I}} \boldsymbol{\Sigma}^{-1/2} = \boldsymbol{\Sigma}^{-1}$$

Note the converse is also true: any whitening whitening matrix, i.e. any $\boldsymbol{W}$ satisfying the whitening constraint, can be written in the above form as $\boldsymbol{Q}_1 = \boldsymbol{W}\boldsymbol{\Sigma}^{1/2}$ is orthogonal by construction.

$\Longrightarrow$ instead of choosing $\boldsymbol{W}$, **we choose the rotation matrix $\boldsymbol{Q}_1$**!

- it is now clear that there are infinitely many whitening procedures, because there are infinitely many rotations! This also means we need to find ways to choose/select among whitening procedures.
- for the Mahalanobis/ZCA transformation $\boldsymbol{Q}_1^{\text{ZCA}} = \boldsymbol{I}$
- **whitening** can be interpreted as **Mahalanobis transform** followed by **rotation**

2.3.4 Another solution (correlation-based)

Instead of working with $\boldsymbol{\Sigma}$, we can express $\boldsymbol{W}$ also in terms of correlation matrix $\boldsymbol{P} = (\rho_{ij})$.

$$\boldsymbol{W} = \boldsymbol{Q}_2 \boldsymbol{P}^{-1/2} \boldsymbol{V}^{-1/2}$$

where $\boldsymbol{V}^{1/2}$ is the diagonal matrix containing the variances.

It is easy to verify that this $\boldsymbol{W}$ also satisfies the whitening constraint:

$$\begin{aligned} \boldsymbol{W}^T\boldsymbol{W} &= \boldsymbol{V}^{-1/2}\boldsymbol{P}^{-1/2} \underbrace{\boldsymbol{Q}_2^T\boldsymbol{Q}_2}_{\boldsymbol{I}} \boldsymbol{P}^{-1/2}\boldsymbol{V}^{-1/2} \\ &= \boldsymbol{V}^{-1/2}\boldsymbol{P}^{-1}\boldsymbol{V}^{-1/2} = \boldsymbol{\Sigma}^{-1} \end{aligned}$$

Conversely, any whitening matrix $\boldsymbol{W}$ can also be written in this form as $\boldsymbol{Q}_2 = \boldsymbol{W}\boldsymbol{V}^{1/2}\boldsymbol{P}^{1/2}$ is orthogonal by construction.

- for Mahalanobis/ZCA transformation $\boldsymbol{Q}_2^{\text{ZCA}} = \boldsymbol{\Sigma}^{-1/2}\boldsymbol{V}^{1/2}\boldsymbol{P}^{1/2}$
- **Another interpretation of whitening**: first **standardising** ($\boldsymbol{V}^{-1/2}$), then **decorrelation** ($\boldsymbol{P}^{-1/2}$), followed by **rotation** ($\boldsymbol{Q}_2$)

Both forms to write $\boldsymbol{W}$ are equally valid (and interchangeable).

Note that for the same $\boldsymbol{W}$

$$\boldsymbol{Q}_1 \neq \boldsymbol{Q}_2 \text{ Two different rotation matrices!}$$

and also

$$\underbrace{\boldsymbol{\Sigma}^{-1/2}}_{\text{Symmetric}} \neq \underbrace{\boldsymbol{P}^{-1/2}\boldsymbol{V}^{-1/2}}_{\text{Not Symmetric}}$$

even though

$$\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\Sigma}^{-1/2} = \boldsymbol{\Sigma}^{-1} = \boldsymbol{V}^{-1/2}\boldsymbol{P}^{-1/2}\boldsymbol{P}^{-1/2}\boldsymbol{V}^{-1/2}$$

2.3.5 Objective criteria for choosing among W using Q_1 or Q_2

1. **Cross-covariance $\boldsymbol{\Phi} = \Sigma_{zx}$** between $\boldsymbol{z}$ and $\boldsymbol{x}$

$$\begin{aligned} \boldsymbol{\Phi} = \text{Cov}(\boldsymbol{z}, \boldsymbol{x}) &= \text{Cov}(\boldsymbol{W}\boldsymbol{x}, \boldsymbol{x}) = \boldsymbol{W}\boldsymbol{\Sigma} \\ &= \boldsymbol{Q}_1\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\Sigma} = \boldsymbol{Q}_1\boldsymbol{\Sigma}^{1/2} \end{aligned}$$

- **Cross-covariance linked with $\boldsymbol{Q}_1$!**
- Choosing suitable cross-covariance allows the selection of a "good" $\boldsymbol{Q}_1$ (and hence $\boldsymbol{W}$)

2. **Cross-correlation $\boldsymbol{\Psi} = \boldsymbol{P}_{\boldsymbol{zx}}$** between $\boldsymbol{z}$ and $\boldsymbol{x}$

$$\begin{aligned}\boldsymbol{\Psi} = \text{Cor}(\boldsymbol{z}, \boldsymbol{x}) &= \boldsymbol{\Phi}\boldsymbol{V}^{-1/2} = \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{V}^{-1/2} \\ &= \boldsymbol{Q}_2\boldsymbol{P}^{-1/2}\boldsymbol{V}^{-1/2}\boldsymbol{\Sigma}\boldsymbol{V}^{-1/2} = \boldsymbol{Q}_2\boldsymbol{P}^{1/2}\end{aligned}$$

- **Cross-correlation linked with $\boldsymbol{Q}_2$!**
- Choosing suitable cross-correlation allows the selection of a "good" $\boldsymbol{Q}_2$ (and hence $\boldsymbol{W}$)

Properties:

$$\boldsymbol{\Psi} = (\psi_{ij})$$

$$\boldsymbol{\Psi}^T\boldsymbol{\Psi} = \boldsymbol{P}$$

$\Longrightarrow \text{Diag}(\boldsymbol{\Psi}^T\boldsymbol{\Psi}) = (1, 1, \ldots, 1)$ and $\text{Tr}(\boldsymbol{\Psi}^T\boldsymbol{\Psi}) = d$

$$\Longrightarrow \sum_{i=1}^{d} \psi_{ij}^2 = 1$$

i.e. the *column* sums of ψ_{ij}^2 are equal to 1.

Interpretation: this is the quared multiple correlation coefficient $R^2 = 1$ between $z_1, ..., z_d$ and x_j!
(as the z_i are all uncorrelated you can simply sum the squared correlations to get R^2; it is equal to 1 because whitening is an invertible linear transformation)

The R^2-s going from $x_1, \ldots, x_d$ to the z_i also equal 1, but because the x_i are correlated they need to be computed as $\text{Diag}(\boldsymbol{\Psi}\boldsymbol{P}^{-1}\boldsymbol{\Psi}^T) = \text{Diag}(\boldsymbol{P}_{\boldsymbol{zx}}\boldsymbol{P}_{\boldsymbol{x}}^{-1}\boldsymbol{P}_{\boldsymbol{xz}}) = (1, 1, \ldots, 1)$.

2.4 Natural whitening procedures

Now we discuss several strategies (maximise correlation between individual components, maximise compression, etc.) to arrive at optimal whitening transformation.

This leads to the following "natural" whitening transformations:

- **Mahalanobis** whitening, also known as **ZCA** (zero-phase component analysis) whitening in machine learning
- **ZCA-cor** whitening
- **Cholesky** whitening
- **PCA** whitening
- **PCA-cor** whitening

In the following $\boldsymbol{x}_c = \boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{x}}$ and $\boldsymbol{z}_c = \boldsymbol{z} - \boldsymbol{\mu}_{\boldsymbol{z}}$ denote the mean-centered variables.

2.4.1 ZCA Whitening

Aim: remove correlations but otherwise keep $\boldsymbol{z}$ as similar as possible to $\boldsymbol{x}$ (componentwise!)

$$\begin{array}{c} z_1 \leftrightarrow x_1 \\ z_2 \leftrightarrow x_2 \\ z_3 \leftrightarrow x_3 \\ \vdots \end{array}$$

Objective function: minimise $\text{E}\left((\boldsymbol{z}_c - \boldsymbol{x}_c)^T(\boldsymbol{z}_c - \boldsymbol{x}_c)\right) = d - 2\text{Tr}(\boldsymbol{\Phi}) + \text{Tr}(\boldsymbol{V})$

Equivalent objective: maximise $\text{Tr}(\boldsymbol{\Phi}) = \text{Tr}(\boldsymbol{Q}_1\boldsymbol{\Sigma}^{1/2})$ to find optimal $\boldsymbol{Q}_1$
Optimal solution: Using the eigendecomposition $\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ we have $\text{Tr}(\boldsymbol{Q}_1\boldsymbol{\Sigma}^{1/2}) = \text{Tr}(\boldsymbol{\Lambda}^{1/2}\boldsymbol{U}^T\boldsymbol{Q}_1\boldsymbol{U})$ which is maximised for $\boldsymbol{U}^T\boldsymbol{Q}_1\boldsymbol{U} = \boldsymbol{I}$. Thus, the optimal rotation matrix is

$$\boldsymbol{Q}_1^{\text{ZCA}} = \boldsymbol{I}$$

$$\Longrightarrow \boldsymbol{W}^{\text{ZCA}} = \boldsymbol{\Sigma}^{-1/2}$$

- ZCA/Mahalanobis transform is the unique transformation that minimises the expected squared component-wise difference between $\boldsymbol{x}$ and $\boldsymbol{z}$.
- Use ZCA and Mahalanobis whitening if you want to "just" remove correlations.

2.4.2 ZCA-Cor Whitening

Aim: same as above but remove scale in $\boldsymbol{x}$ first before comparing to $\boldsymbol{z}$
Objective function: minimise $\text{E}\left((\boldsymbol{z}_c - \boldsymbol{V}^{-1/2}\boldsymbol{x}_c)^T(\boldsymbol{z}_c - \boldsymbol{V}^{-1/2}\boldsymbol{x}_c)\right) = 2d - 2\text{Tr}(\boldsymbol{\Psi})$
Equivalent objective: maximise $\text{Tr}(\boldsymbol{\Psi}) = \text{Tr}(\boldsymbol{Q}_2\boldsymbol{P}^{1/2})$ to find optimal $\boldsymbol{Q}_2$
Optimal solution: same as above for ZCA but using correlation matrix instead of covariance: using the eigendecomposition of $\boldsymbol{P} = \boldsymbol{G}\boldsymbol{\Theta}\boldsymbol{G}^T$ we get $\text{Tr}(\boldsymbol{Q}_2\boldsymbol{P}^{1/2})) = \text{Tr}(\boldsymbol{\Theta}^{1/2}\boldsymbol{G}^T\boldsymbol{Q}_2\boldsymbol{G})$ which is maximised for

$$\boldsymbol{Q}_2^{\text{ZCA-Cor}} = \boldsymbol{I}$$

$$\Longrightarrow \boldsymbol{W}^{\text{ZCA-Cor}} = \boldsymbol{P}^{-1/2}\boldsymbol{V}^{-1/2}$$

- ZCA-cor whitening is the unique whitening transformation if you aim to maximise correlation between corresponding components in $\boldsymbol{x}$ and $\boldsymbol{z}$.
- Only if $\boldsymbol{x}$ is standardised to $\text{Var}(x_i) = 1$ then ZCA and ZCA-cor are identical
- ZCA and ZCA-cor: lead to interpretable $\boldsymbol{z}$

2.4.3 PCA Whitening

Aim: remove correlations and compress information in $\boldsymbol{x}$ (each component z_i maximally linked with all variables in $\boldsymbol{x}$):

$$\begin{array}{llllll} z_1 & \leftarrow x_1 & z_2 & \leftarrow x_1 & \dots \\ z_1 & \leftarrow x_2 & z_2 & \leftarrow x_2 & \\ \vdots & & & & \\ z_1 & \leftarrow x_d & z_2 & \leftarrow x_d & \end{array}$$

Objective: maximise $\sum_{j=1}^d \text{Cov}(z_i, x_j)^2 = \sum_{j=1}^d \phi_{ij}^2$ for all i
Equivalent objective: maximise $(\phi_1, ..., \phi_d)^T = \text{Diag}(\boldsymbol{\Phi}\boldsymbol{\Phi}^T) = \text{Diag}(\boldsymbol{Q}_1 \boldsymbol{\Sigma} \boldsymbol{Q}_1^T) = \text{Diag}((\boldsymbol{Q}_1\boldsymbol{U})\boldsymbol{\Lambda}(\boldsymbol{Q}_1\boldsymbol{U})^T)$, with $\phi_1 > \phi_2 > \dots > \phi_d$

Optimal solution: this is optimised for $\boldsymbol{Q}_1\boldsymbol{U} = \boldsymbol{I}$ hence

$$\boldsymbol{Q}_1^{\text{PCA}} = \boldsymbol{U}^T$$

$$\Longrightarrow \boldsymbol{W}^{\text{PCA}} = \boldsymbol{U}^T \boldsymbol{\Sigma}^{-1/2} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{U}^T$$

- Optimum value: $\text{Diag}(\boldsymbol{Q}_1^{\text{PCA}} \boldsymbol{\Sigma} (\boldsymbol{Q}_1^{\text{PCA}})^T) = \text{Diag}(\boldsymbol{\Lambda}) = (\lambda_1, \dots, \lambda_d)^T$, i.e. the eigenvalues of $\boldsymbol{\Sigma}$ are equal to the sum of the squared covariances from each latent component to all the original variables.
- PCA whitening is a whitening transformation that maximises compression with the sum of squared cross-covariances as underlying optimality criteron

The fraction $\frac{\lambda_i}{\sum_{j=1}^d \lambda_j}$ can be used as a measure of relative importance of each component in $\boldsymbol{z}$ to explain the original variables. Thus, low ranking components in $\boldsymbol{z}$ with low total squared covariance with $\boldsymbol{x}$ may be discarded, thus leading to a reduction in dimension.

Note that the column signs in $\boldsymbol{U}$ are not identified in an eigenvalue decomposition. Therefore, it is useful to impose a further constraint on the rotation matrix. A useful condition is to assume a positive diagonal, i.e. $\text{Diag}(\boldsymbol{Q}_1^{\text{PCA}}) > 0$ and thus $\text{Diag}(\boldsymbol{U}) > 0$ and $\text{Diag}(\boldsymbol{\Phi}) > 0$.

2.4.4 PCA-cor Whitening

Aim: same as for PCA whitening but remove scale in $\boldsymbol{x}$ first (i.e. use squared correlations rather squared covariances to measure compression)
Objective: maximise $\sum_{j=1}^d \text{Cor}(z_i, x_j)^2 = \sum_{i=1}^d \psi_{ij}^2$ for all i
Equivalent objective: maximise $(\psi_1, ..., \psi_d)^T = \text{Diag}(\boldsymbol{\Psi}\boldsymbol{\Psi}^T) = \text{Diag}(\boldsymbol{Q}_2 \boldsymbol{P} \boldsymbol{Q}_2^T) = \text{Diag}((\boldsymbol{Q}_2\boldsymbol{G})\boldsymbol{\Theta}(\boldsymbol{Q}_2\boldsymbol{G})^T$ with $\psi_1 > \psi_2 > \dots > \psi_d$.

Optimal solution: this is optimised for $\boldsymbol{Q}_2\boldsymbol{G} = \boldsymbol{I}$ hence

$$\boldsymbol{Q}_2^{\text{PCA-Cor}} = \boldsymbol{G}^T$$

$$\Longrightarrow \boldsymbol{W}^{\text{PCA-Cor}} = \boldsymbol{\Theta}^{-1/2} \boldsymbol{G}^T \boldsymbol{V}^{-1/2}$$

- Optimum value: $\text{Diag}(\boldsymbol{Q}_2^{\text{PCA-Cor}} \boldsymbol{P} (\boldsymbol{Q}_2^{\text{PCA-Cor}})^T) = \text{Diag}(\boldsymbol{\Theta}) = (\theta_1, \ldots, \theta_d)^T$, i.e. the eigenvalues of $\boldsymbol{P}$ are equal to the sum of the squared correlations from each latent component to all the original variables.
- PCA-cor whitening is a whitening procedure that maximises compression with the sum of squared cross-correlation as optimality criterion
- If $\boldsymbol{x}$ is standardised to $\text{Var}(x_i) = 1$, then PCA and PCA-cor are identical.

Note that $\sum_{j=1}^d \theta_j = \text{Tr}(\boldsymbol{P}) = d$. Therefore the fraction $\frac{\theta_i}{\sum_{j=1}^d \theta_j} = \frac{\theta_j}{d}$.

This can be used as a measure of relative importance of each component in $\boldsymbol{z}$ to explain the original variables. Thus, low ranking components in $\boldsymbol{z}$ with low total squared correlation with $\boldsymbol{x}$ may be discarded, thus leading to a reduction in dimension.

Note that the individual correlations $cor(z_i, x_j)$ between each component in $\boldsymbol{z}$ and the original variables in $\boldsymbol{x}$ are the correlation loadings (see section for PCA below).

As with PCA whitening for identifiability we need to impose a further constraint on the rotation matrix $\boldsymbol{Q}_2$. A useful condition is to assume a positive diagonal, i.e. $\text{Diag}(\boldsymbol{Q}_2^{\text{PCA-Cor}}) > 0$ and thus $\text{Diag}(\boldsymbol{G}) > 0$ and $\text{Diag}(\boldsymbol{\Psi}) > 0$.

2.4.5 Cholesky Whitening

Aim: find a whitening transformation such that the cross-covariance and cross-correlation have triangular structure. This is useful in some models (such as time course data) to ensure that the future cannot influence the past.

Solution: Cholesky decomposition of $\boldsymbol{\Sigma}^{-1} = \boldsymbol{L}\boldsymbol{L}^T$

$\boldsymbol{L}$ is a lower triangular matrix with positive diagonal elements
$\Longrightarrow \boldsymbol{W}^{\text{Chol}} = \boldsymbol{L}^T$

By construction, $\boldsymbol{W}^{\text{Chol}}$ satisfies the whitening constraint since $(\boldsymbol{W}^{\text{Chol}})^T \boldsymbol{W}^{\text{Chol}} = \boldsymbol{\Sigma}^{-1}$.

The corresponding rotation matrices are:

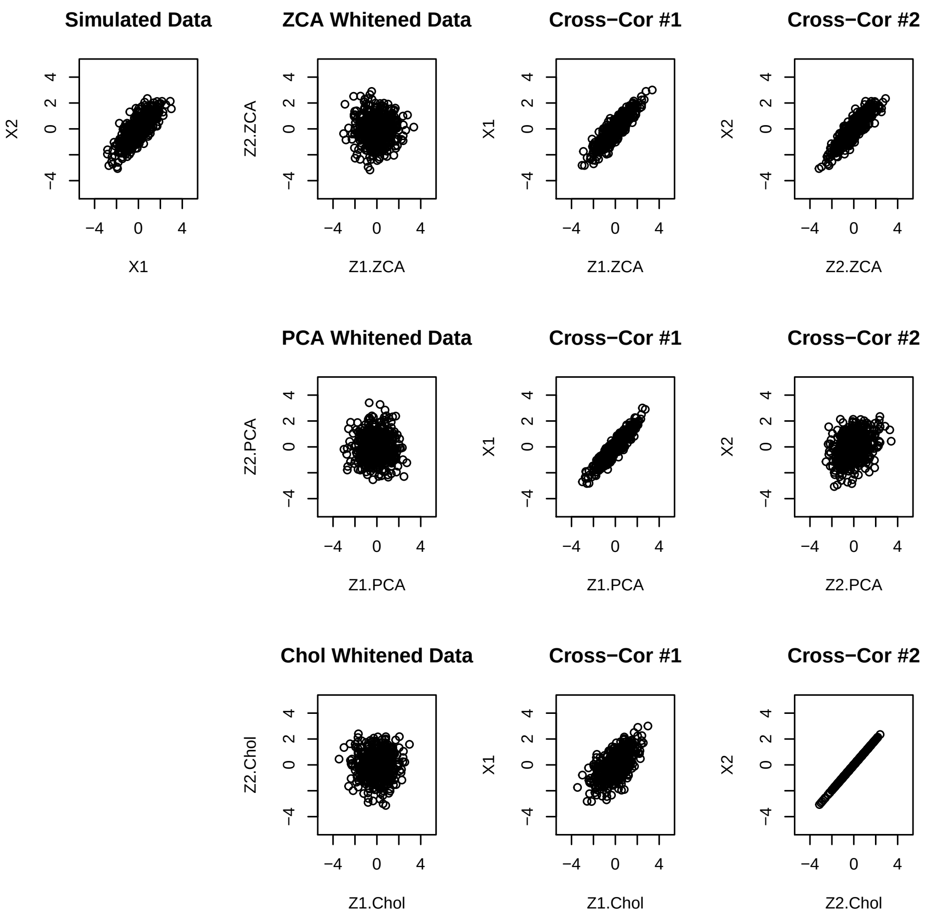
- $\boldsymbol{Q}_1^{\text{Chol}} = \boldsymbol{L}^T \boldsymbol{\Sigma}^{1/2}$ and
- $\boldsymbol{Q}_2^{\text{Chol}} = \boldsymbol{L}^T \boldsymbol{V}^{1/2} \boldsymbol{P}^{1/2}$

This results in:

- $\boldsymbol{\Phi}^{\text{Chol}} = \boldsymbol{L}^T \boldsymbol{\Sigma}$ and
- $\boldsymbol{\Psi}^{\text{Chol}} = \boldsymbol{L}^T \boldsymbol{\Sigma} \boldsymbol{V}^{-1/2}$

Note that the cross-covariance matrix $\boldsymbol{\Phi}^{\text{Chol}}$ and the cross-correlation matrix $\boldsymbol{\Psi}^{\text{Chol}}$ by construction are also lower triangular matrices with positive diagonal elements!

2.4.6 Comparison of ZCA, PCA and Chol whitening



In the above comparison you see ZCA, PCA and Cholesky whitening applied to a simulated bivariate normal data set with correlation $\rho = 0.8$ (column 1). All approaches equally succeed in whitening (column 2) but they differ in the cross-correlations. Columns 3 and 4 shows the cross-correlations between the first two pairs corresponding components for ZCA, PCA and Cholesky whitening. As expected, in ZCA both components show strong correlation, but this is not the case for PCA and Cholesky whitening.

2.4.7 Recap

Method	Type of usage
ZCA, ZCA-cor:	pure decorrelate, maintain similarity to original data set, interpretability
PCA, PCA-cor:	compression, find effective dimension, reduce dimensionality, feature identification
Chol:	time course data

Related models not discussed in this course:

- Factor models: essentially whitening plus an additional error term, factors have rotational freedom just like in whitening
- PLS: similar to PCA but in regression setting (with the choice of latent variables depending on the response)

2.5 Principal Component Analysis (PCA)

- Traditional PCA (invented 1901 by Pearson) is very closely related to **PCA whitening**.
- But PCA itself is **not** a whitening procedure!

PCA transformation:

$$\underbrace{\boldsymbol{t}}_{\text{Principal Components}} = \underbrace{\boldsymbol{U}^T\boldsymbol{x}}_{\text{Orthogonal projection}}$$

$$\Longrightarrow \text{Var}(\boldsymbol{t}) = \boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_d \end{pmatrix}$$

- Principal components are **orthogonal** but do *not* have unit variance!
- The variances of the principal components are equal to the eigenvalues of $\boldsymbol{\Sigma}$: $\text{Var}(\boldsymbol{t}^{\text{PCA}}) = \boldsymbol{\Lambda}$.
- You arrive at PCA whitening by standardising the PCA components: $\boldsymbol{z}^{\text{PCA}} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{t}^{\text{PCA}}$
- Same compression / optimality properties as PCA whitening.

With principle components the fraction $\frac{\lambda_i}{\sum_{j=1}^d \lambda_j}$ can be interpreted as the proportion of variation contributed by each component in $\boldsymbol{t}^{\text{PCA}}$ to the total variation. Thus, low ranking components in $\boldsymbol{t}^{\text{PCA}}$ with low variation may be discarded, thus leading to a reduction in dimension.

2.5.1 Application to data

Written in terms of a data matrix $\boldsymbol{X}$ instead of a random vector $\boldsymbol{x}$ PCA becomes:

$$\underbrace{\boldsymbol{T}}_{\text{Sample version of principal components}} = \underbrace{\boldsymbol{X}}_{\text{Data matrix}} \boldsymbol{U}$$

There are now two ways to obtain $\boldsymbol{U}$:

1. Estimate the covariance matrix, e.g. by $\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\boldsymbol{X}_c^T\boldsymbol{X}_c$ where $\boldsymbol{X}_c$ is the column-centred data matrix; then apply the eigenvalue decomposition on $\hat{\boldsymbol{\Sigma}}$ to get $\boldsymbol{U}$.
2. Compute the singular value decomposition of $\boldsymbol{X}_c = \boldsymbol{V}\boldsymbol{D}\boldsymbol{U}^T$. As $\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\boldsymbol{X}_c^T\boldsymbol{X} = \boldsymbol{U}(\frac{1}{n}\boldsymbol{D}^2)\boldsymbol{U}^T$ you can just use $\boldsymbol{U}$ from the SVD of $\boldsymbol{X}_c$ and there is no need to compute the covariance.

2.6 PCA correlation loadings and plot

A useful quantity to evaluate the PCA whitened components $\boldsymbol{z}^{\text{PCA}}$ and the related principle components $\boldsymbol{t}^{\text{PCA}}$ is the cross-correlation matrix $\boldsymbol{\Psi}$. Specifically, $\boldsymbol{\Psi} = \text{Cor}(\boldsymbol{z}^{\text{PCA}}, \boldsymbol{x}) = \text{Cor}(\boldsymbol{t}^{\text{PCA}}, \boldsymbol{x}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{U}^T\boldsymbol{V}^{-1/2}$.

We now consider the back-transformation of the (standardised) PCA components to the standardised original components:

- The inverse PCA transformation is $\boldsymbol{x} = \boldsymbol{U}\boldsymbol{t}^{\text{PCA}}$ (note that $\boldsymbol{U}^{-1} = \boldsymbol{U}^T$).
- If one standardises $\boldsymbol{x}$ this equation becomes $\boldsymbol{V}^{-1/2}\boldsymbol{x} = \boldsymbol{V}^{-1/2}\boldsymbol{U}\boldsymbol{t}^{\text{PCA}}$.
- expressed in terms of the standardised $\boldsymbol{z}^{\text{PCA}}$ it becomes $\boldsymbol{V}^{-1/2}\boldsymbol{x} = (\boldsymbol{V}^{-1/2}\boldsymbol{U}\boldsymbol{\Lambda}^{1/2})\boldsymbol{\Lambda}^{-1/2}\boldsymbol{t}^{\text{PCA}}$, or equivalently, $\boldsymbol{V}^{-1/2}\boldsymbol{x} = \boldsymbol{\Psi}^T\boldsymbol{z}^{\text{PCA}}$.

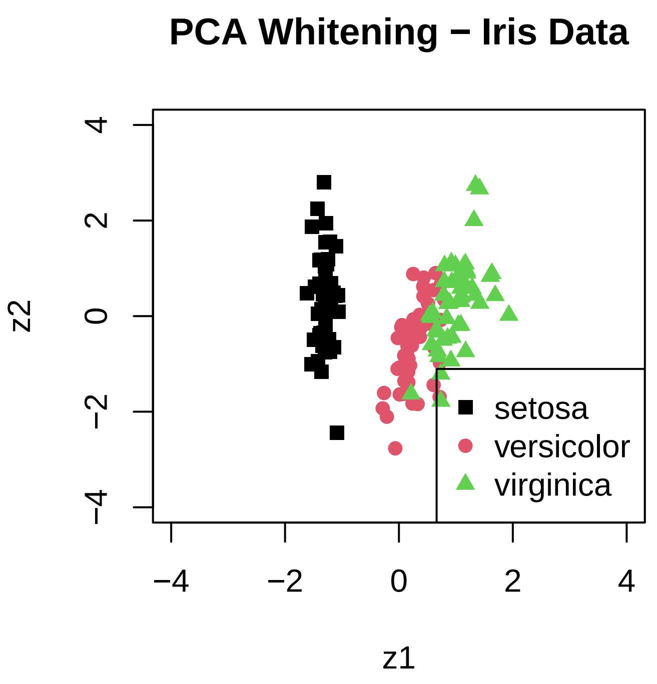
Thus the cross-correlation matrix $\boldsymbol{\Psi}$ plays the role of *correlation loadings*, i.e. they are the coefficients linking the (standardised) PCA components with the standardised original components.

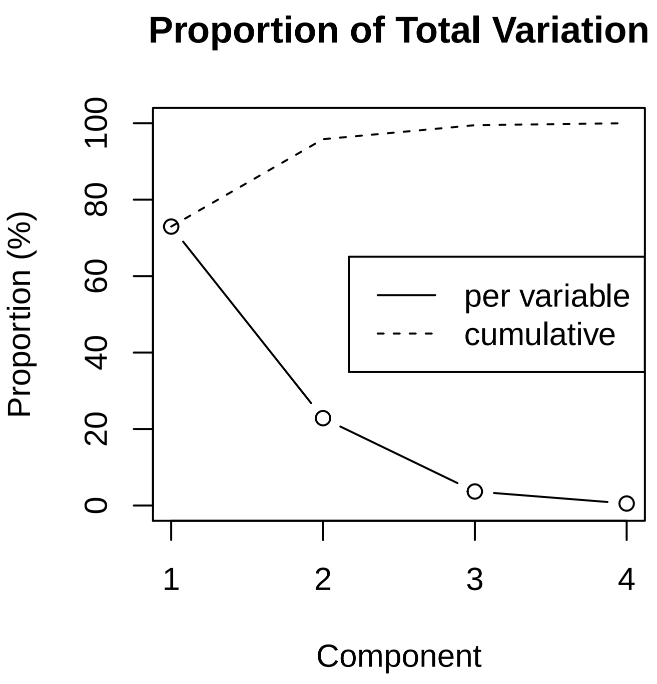
Recall that in a linear regression model with *uncorrelated predictors* the (squared) correlations between each predictor and the response can be used as measure of variable importance. Since PCA (whitened) components are uncorrelated by construction, the correlation loadings $\boldsymbol{\Psi}$ measure the capability of each principal component to predict the original variables. Note that this is explicitly maximised in PCA-cor whitening — but not in PCA and in PCA whitening because those methods use squared covariance (rather than correlation) in the objective function.

For visualisation, a correlation loadings plot is often constructed as follows. The loadings $\boldsymbol{\Psi}$ for the first two whitened PCA components z_1, z_2 (or equivalently for t_1, t_2) to all original variables are computed. Then a point for each orginal variable is drawn in a plane with the two correlation loadings acting as its coordinates. By construction, all points have to lie within a unit circle around the origin . The orginal variables most strongly influenced by the two latent variables will have strong correlation and thus lie near the outer circle, whereas variables that are not influenced by the two latent variables will lie near the origin. (see next section for an example).

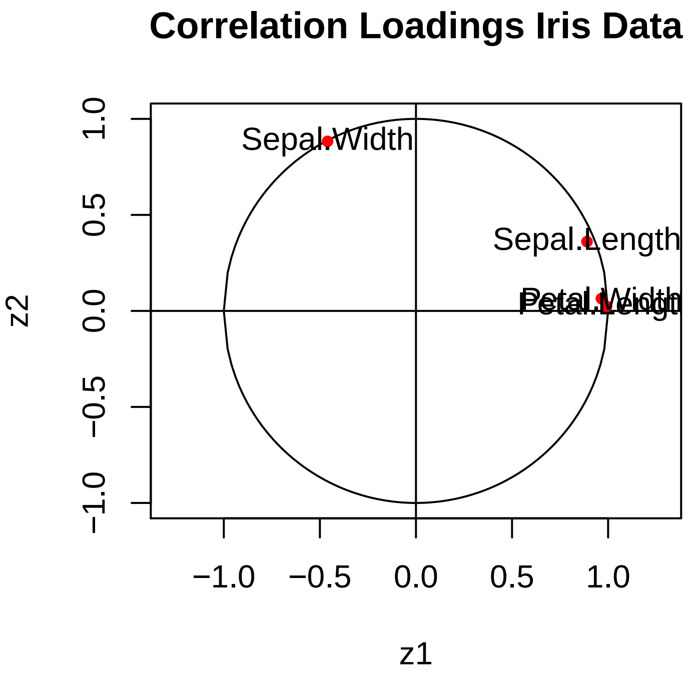
2.6.1 Iris data example

A plot of the the first two components after PCA-whitening is applied reveals the group structure among iris flowers:





Here is the corresponding loadings plot:



2.7 CCA whitening (Canonical Correlation Analysis)

So far, we have looked only into whitening as a **single** vector $\boldsymbol{x}$. In CCA whitening we consider **two vectors** $\boldsymbol{x}$ and $\boldsymbol{y}$ simultaneously:

$$\underset{\text{Dimension } p}{\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix}} \qquad \underset{\text{Dimension } q}{\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_q \end{pmatrix}} \qquad \begin{array}{l} \text{Var}(\boldsymbol{x}) = \boldsymbol{\Sigma}_{\boldsymbol{x}} = \boldsymbol{V}_x^{1/2} \boldsymbol{P}_x \boldsymbol{V}_x^{1/2} \\ \text{Var}(\boldsymbol{y}) = \boldsymbol{\Sigma}_{\boldsymbol{y}} = \boldsymbol{V}_y^{1/2} \boldsymbol{P}_y \boldsymbol{V}_y^{1/2} \end{array}$$

$$\begin{array}{ll} \text{Whitening of } \boldsymbol{x}\text{:} & \boldsymbol{z}_x = \boldsymbol{W}_x \boldsymbol{x} = \boldsymbol{Q}_x \boldsymbol{P}_x^{-1/2} \boldsymbol{V}_x^{-1/2} \boldsymbol{x} \\ \text{Whitening of } \boldsymbol{y}\text{:} & \boldsymbol{z}_y = \boldsymbol{W}_y \boldsymbol{y} = \boldsymbol{Q}_y \boldsymbol{P}_y^{-1/2} \boldsymbol{V}_y^{-1/2} \boldsymbol{y} \end{array}$$

(note we use the correlation-based form of $\boldsymbol{W}$)

Cross-correlation between $\boldsymbol{z}_y$ and $\boldsymbol{z}_y$:

$$\text{Cor}(\boldsymbol{z}_x, \boldsymbol{z}_y) = \boldsymbol{Q}_x \boldsymbol{K} \boldsymbol{Q}_y^T$$

with $\boldsymbol{K} = \boldsymbol{P}_x^{-1/2} \boldsymbol{P}_{xy} \boldsymbol{P}_y^{-1/2}$.

Idea: we can choose a suitable $\boldsymbol{Q}_x$ and $\boldsymbol{Q}_y$ rotation matrix by putting constraints on the cross-correlation.

CCA: we aim for a *diagonal* $\text{Cor}(\boldsymbol{z}_x, \boldsymbol{z}_y)$ so that each component in $\boldsymbol{z}_x$ only influences one (the corresponding) component in $\boldsymbol{z}_y$.

Motivation: pairs of "modules" represented by components of $\boldsymbol{z}_x$ and $\boldsymbol{z}_y$ influencing each other (and not anyone other module).

$$\boldsymbol{z}_x = \begin{pmatrix} z_1^x \\ z_2^x \\ \vdots \\ z_p^x \end{pmatrix} \quad \boldsymbol{z}_y = \begin{pmatrix} z_1^y \\ z_2^y \\ \vdots \\ z_q^y \end{pmatrix}$$

\end{align}

$$\text{Cor}(\boldsymbol{z}_x, \boldsymbol{z}_y) = \begin{pmatrix} d_1 & \dots & 0 \\ \vdots & \vdots & \\ 0 & \dots & d_m \end{pmatrix}$$

where d_i are the *canonical correlations* and $m = \min(p, q)$.

2.7.1 How to make cross-correlation matrix $\text{Cor}(\boldsymbol{z}_x, \boldsymbol{z}_y)$ diagonal?

- Use Singular Value Decomposition (SVD) of matrix $\boldsymbol{K}$:

$$\boldsymbol{K} = (\boldsymbol{Q}_x^{\text{CCA}})^T \boldsymbol{D} \boldsymbol{Q}_y^{\text{CCA}}$$

  where $\boldsymbol{D}$ is the diagonal matrix containing the singular values of $\boldsymbol{K}$
- This yields rotation matrices $\boldsymbol{Q}_x^{\text{CCA}}$ and $\boldsymbol{Q}_y^{\text{CCA}}$ and thus the desired whitened matrices $\boldsymbol{W}_x^{\text{CCA}}$ and $\boldsymbol{W}_y^{\text{CCA}}$
- As a result $\text{Cor}(\boldsymbol{z}_x, \boldsymbol{z}_y) = \boldsymbol{D}$ i.e. singular values of $\boldsymbol{K}$ are identical to canonical correlations d_i!

$\longrightarrow$ $\boldsymbol{Q}_x^{\text{CCA}}$ and $\boldsymbol{Q}_y^{\text{CCA}}$ are determined by the diagonality constraint (and are different to the other previously discussed whitening methods).

Note that the signs of corresponding in columns in $\boldsymbol{Q}_x^{\text{CCA}}$ and $\boldsymbol{Q}_y^{\text{CCA}}$ are not identified. Traditionally, in an SVD the signs are chosen such that the singular values are positive. However, if we impose positive-diagonality on $\boldsymbol{Q}_x^{\text{CCA}}$ and $\boldsymbol{Q}_y^{\text{CCA}}$, and thus positive-diagonality on the cross-correlations $\boldsymbol{\Psi}_x$ and $\boldsymbol{\Psi}_y$, then the canonical correlations may take on both positive and negative values.

2.7.2 Related methods

- O2PLS: similar to CCA but using orthogonal projections (thus in O2PLS the latent variables underlying $\boldsymbol{x}$ and $\boldsymbol{y}$ are not orthogonal)
- Vector correlation: aggregates the squared canonical correlations into a single overall measure of association between two random vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ (see Chapter 5 on multivariate dependencies).

Chapter 3

Clustering / unsupervised Learning

3.1 Overview of clustering

3.1.1 General aim

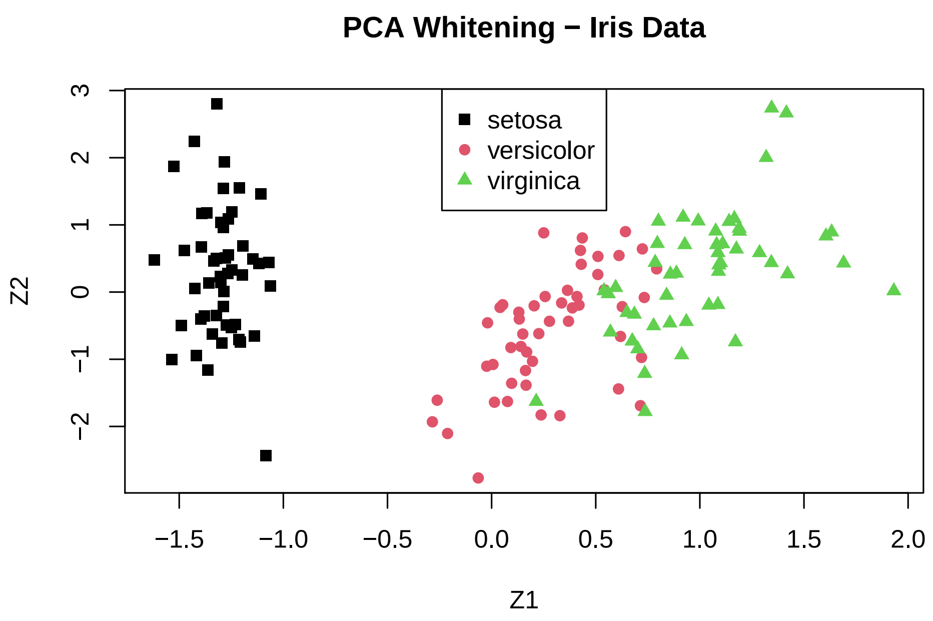
Find structure and gain insights in data $x_1, \ldots, x_n$ collected on n objects by **categorising the objects into groups** based on the d features (= the d components in x_i) obtained for each object.

In machine learning / statistical learning this process is called *unsupervised learning* or *clustering*, and there are many algorithms and procedures to automatise and quantify this process.

Unsupervised learning is a very hard problem!

Note that **unsupervised learning** the class labels are a priori unknown, and are learned in the process of clustering. In contrast, in **supervised learning** the class labels are known, at least for the training data set.

Recalle the Isis flower data (from Worksheet 4): PC1 vs. PC2 shows clear visual grouping into 2-3 clusters:



Thus, by clustering this data we aim to identify this structure that is visualised here by the *given* class labels.

Two extremes in clustering:

1) put all objects into a single cluster (low complexity model)
2) put each object into its own cluster (high complexity model)

In both instances nothing new has been learned! In practise, the aim is to find a compromise, i.e. a model that captures the structure in the data with appropriate complexity (not too low and not too complex).

Questions / Problems:

- how do we define clusters?
- how do we learn / infer clusters?
- how many clusters? (this is surprisingly difficult!)
- which features define / separate each cluster?
- uncertainty of clusters?

$\Longrightarrow$ Clustering / partitioning / structure discovery is not easy!

$\Longrightarrow$ We cannot expect perfect answers or a single "true" clustering (this is related to the problem of model selections, many differnt clusterings may fit the data equally well)

$\Longrightarrow$ Ideally we would wish to get information about the uncertainty of the clustering solution (e.g. by considering sets of possible clusters)

3.1.2 Why is clustering difficult?

Partioning problem (combinatorics): How many partitions of n objects (say flowers) into K groups (say species) exists?

Answer:

$$S(n, K) = \left\{ \begin{matrix} n \\ K \end{matrix} \right\}$$

this is the “Sterling number of the second type”.

For large n:

$$S(n, K) \approx \frac{K^n}{K!}$$

Example:

n	K	Number of possible partitions
15	3	≈ 2.4 million (10^6)
20	4	≈ 2.4 billion (10^9)
⋮		
100	5	$\approx 6.6 \times 10^{76}$

These are enormously big numbers even for relatively small problems!

3.1.3 Common types of clustering methods

There are very many different clustering algorithms!

We consider the following two broad types of methods:

1) **Algorithmic clustering methods** (these are not explicitly based on a probabilistic model)

- K-means
- PAM
- hierarchical clustering (distance or similarity-based, divise and agglomerative)

  pros: fast, effective algorithms to find at least some grouping

  cons: no probabilistic interpretation, blackbox methods

2) **Model-based clustering** (based on a probabilistic model)

- mixture models (e.g. Gaussian mixture models, GMMs, non-hierarchical)
- graphical models (e.g. Bayesian networks, Gaussian graphical models GGM, trees and networks)

  pros: full probabilistic model with all corresponding advantages

  cons: computationally very expensive, sometimes impossible to compute exactly.

3.2 Hierarchical clustering

3.2.1 Tree-like structures

Often, categorisations of objects are naturally nested, i.e. there sub-categories of categories etc. Thes can be represented by **tree-like hierarchical structures**.

In many branches of science hierarchical clusterings are widely employed, for example in evolutionary biology: see e.g.

- Tree of Life with linking the three natural kingdoms
- phylogenetic trees among species (e.g. vertebrata)
- population genetic trees to describe human evolution
- taxonomic trees for plant species
- etc.

Note that when visualising hierarchical structures typically the corresponding tree is depicted facing downwards, i.e. the root of the tree is shown on the top, and the tips/leaves of the tree are shown at the bottom!

In order to obtain such a hierarchical clustering from data two opposing strategies are commonly used:

1) **divisive or recursive partitioning algorithms**
    - grow the tree from the root downwards
    - first determine the main two clusters, then recursively refine the clusters further
2) **agglomerative algorithms**
    - grow the tree from the leafs upwards
    - successively form partitions by first joining individual object together, then recursively join groups of items together, until all is merged.

For example, K-means can be turned into a divisive hierarchical clustering algorithm by recursively applying the algorithm with $K = 2$.

In the following we discuss a number of popular hierarchical agglomerative clustering algorithms that are based on the pairwise distances / similarities (a $n \times n$ matrix) among all data points.

3.2.2 Agglomerative hierarchical clustering algorithms

A general algorithm for agglomerative construction of a hierarchical clustering works as follows:

Initialisation:

Compute a dissimilarity / distance matrix between all pairs of objects where "objects" are single data points at this stage but later are also be sets of data points.

Iterative procedure:

1) identify the pair of objects with the smallest distance. These two objects are then merged together into a common set. Create an internal node in the tree to describe this coalescent event.

2) update the distance matrix by computing the distances between the new set and all other objects. If the new set contains all data points the procedure terminates.

For actual implementation of this algorithm two key ingredients are needed:

1) a distance measure $d(\boldsymbol{a}, \boldsymbol{b})$ between two data points $\boldsymbol{a}$ and $\boldsymbol{b}$.

   This is typically on of the following.

   - Euclidean distance $d(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{(\boldsymbol{a}-\boldsymbol{b})^T(\boldsymbol{a}-\boldsymbol{b})}$
   - Manhattan distance $d(\boldsymbol{a}, \boldsymbol{b}) = \sum_{i=1}^{d} |a_i - b_i|$
   - maximum norm $d(\boldsymbol{a}, \boldsymbol{b}) = \max\limits_{i \in \{1,\ldots,d\}} |a_i - b_i|$
   - etc

   In the end, making the correct choice of distance will require subject knowledge about the data!

2) a distance measure between two sets of objects $A = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_{n_A}\}$ and $B = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{n_B}\}$ of size n_A and n_B, respectively. The centroids of the two sets is given by $\boldsymbol{\mu}_A = \frac{1}{n_A}\sum_{\boldsymbol{a}_i \in A} \boldsymbol{a}_i$ and $\boldsymbol{\mu}_B = \frac{1}{n_B}\sum_{\boldsymbol{b}_i \in B} \boldsymbol{b}_i$.

   To determine the distance $d(A, B)$ between these two sets the following measures are often employed:

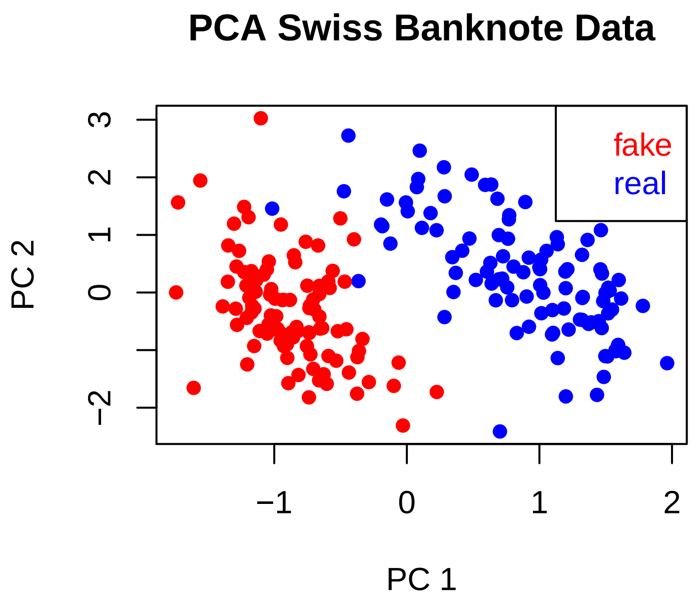
   - **complete linkage** (max. distance): $d(A, B) = \max\limits_{\boldsymbol{a}_i \in A, \boldsymbol{b}_i \in B} d(\boldsymbol{a}_i, \boldsymbol{b}_i)$
   - **single linkage** (min. distance): $d(A, B) = \min\limits_{\boldsymbol{a}_i \in A, \boldsymbol{b}_i \in B} d(\boldsymbol{a}_i, \boldsymbol{b}_i)$
   - **average linkage** (avg. distance): $d(A, B) = \frac{1}{n_A n_B}\sum_{\boldsymbol{a}_i \in A}\sum_{\boldsymbol{b}_i \in B} d(\boldsymbol{a}_i, \boldsymbol{b}_i)$

Another agglomerative hierarchical procedure is **Ward's minimum variance approach**. In this approach in each iteration the two sets A and B are merged that lead to the *smallest increase in total within-group sum of squares* (cf. K-means). Normally, the within-group A sum of squares $w_A = \sum_{\boldsymbol{a}_i \in A}(\boldsymbol{a}_i - \boldsymbol{\mu}_A)^T(\boldsymbol{a}_i - \boldsymbol{\mu}_A)$ is computed on the basis of actual observations $\boldsymbol{a}_i$ relative to their mean $\boldsymbol{\mu}_A$. However, it is equally possible to compute it in terms of the squared pairwise differences between the observations using $w_A = \frac{1}{n_A}\sum_{\boldsymbol{a}_i, \boldsymbol{a}_j \in A, i<j}(\boldsymbol{a}_i - \boldsymbol{a}_j)^T(\boldsymbol{a}_i - \boldsymbol{a}_j)$. This is exploited in Ward's clustering method where the distance measure between to sets A and B is $d(A, B) = w_{A \cap B} - w_A - w_B$. Correspondingly, between two data points $\boldsymbol{a}$ and $\boldsymbol{b}$ it is the squared Euclidean distance $d(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}-\boldsymbol{b})^T(\boldsymbol{a}-\boldsymbol{b})$.

3.2.3 Application to Swiss banknote data set

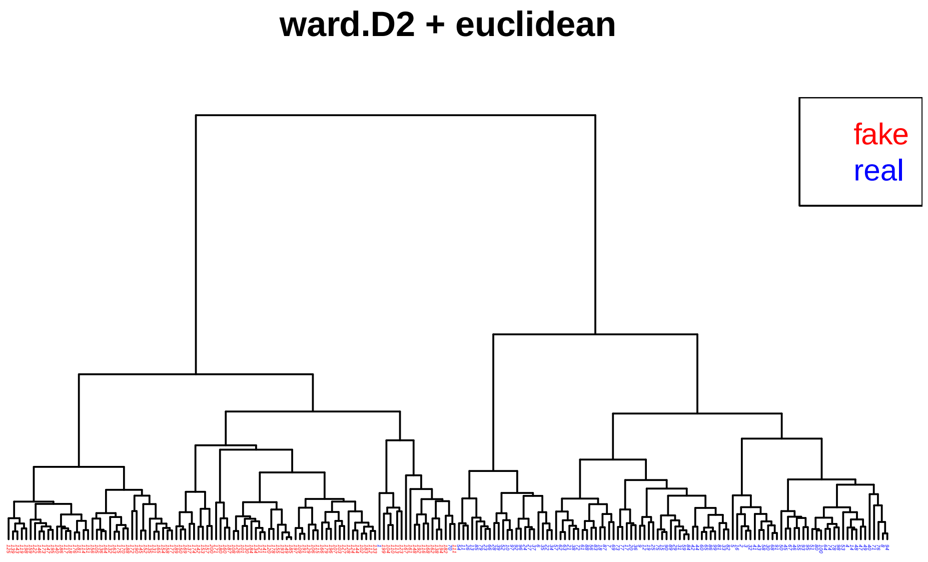
This data set is reports 6 pysical measurements on 200 Swiss bank notes. Of the 200 notes 100 are fake and 100 are real. The measurements are: length, left width, right width, bottom margin, top margin, diagonal length of the bank notes.

PCA of this data shows that there are indeed two well defined groups, and that these groups correspond precisely to the real and fake banknotes:

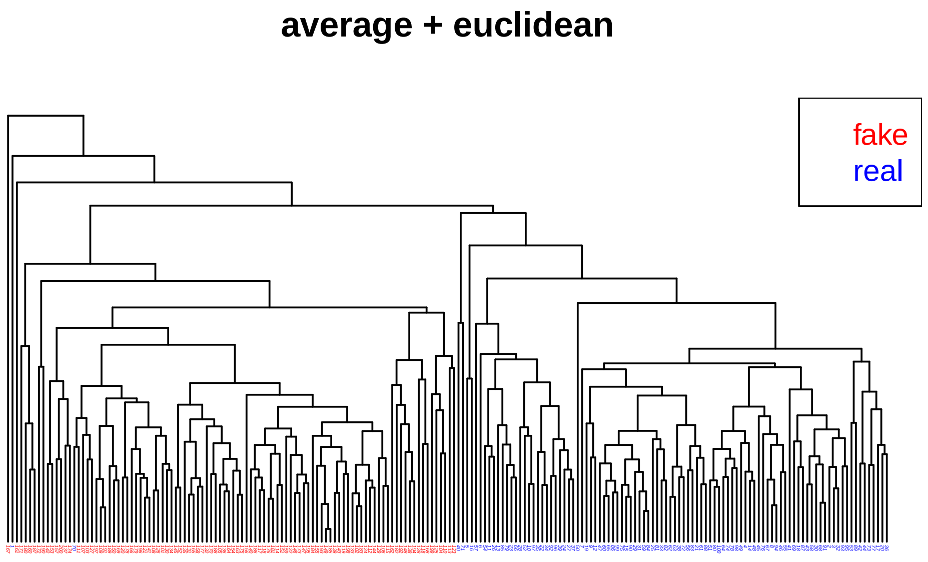


We now compare the clusterings of the above four different hierarchical methods using Euclidean distance:

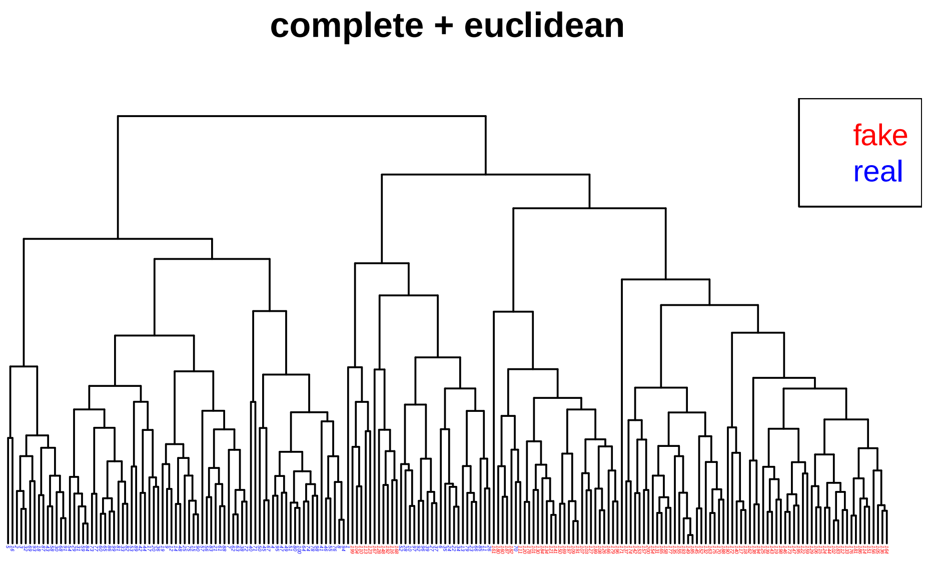
Ward.D2 (=Ward's method):



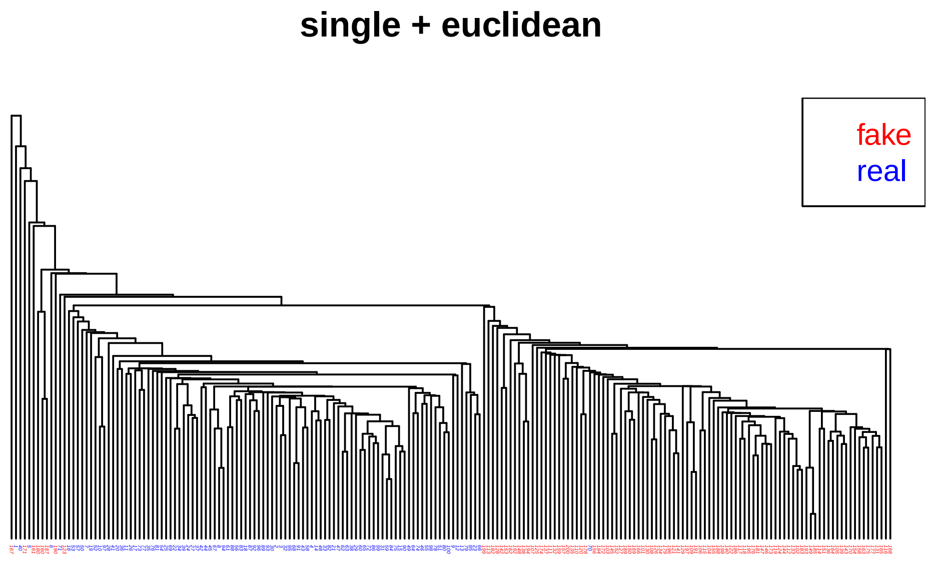
Average linkage:



Complete linkage:



Single linkage:



Result:

- All four trees / hierarchical clusterings are quite different!
- The Ward.D2 method is the only one that finds the correct grouping (except for a single error).

In practical application of hierarchical clustering methods is is essential to evaluate the stability and uncertainty of the obtained groupings. This is often done as follows:

- "bootstrap" (i.e. resampling the original data) is used to generate new data sets (say 200) similar to the original one, and to construct a hierarchical clustering for each of these data sets.
- a consensus tree is computed from the 200 bootstrap trees. This reduces the variability of te estimated tree and also provides bootstrap values measuring the stability of each implied cluster in the tree.

Disadvantage: bootstrapping trees is computationally very expensive!

3.3 *K*-means clustering

3.3.1 General aims

- Partition the data into K groups, with K given in advance
- The groups are non-overlapping, so each of the n data points / objects $\boldsymbol{x}_i$ is assigned to exactly one of the K groups
- maximise the homogeneity with each group (i.e. each group should contain similar objects)
- maximise the heterogeneity among the different groups (i.e each group should differ from the other groups)

3.3.2 Algorithm

For each group $k \in \{1, \ldots, K\}$ we assume a group mean $\boldsymbol{\mu}_k$.
After running K-means we will get estimates of $\hat{\boldsymbol{\mu}}_k$ of the group means, as well as an allocation of each data point to one of the classes.

Initialisation:

At the start of the algorithm the n observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are randomly allocated to one of the K groups. The resulting assignment is given by the function $C(\boldsymbol{x}_i) \in \{1, \ldots, K\}$. With $G_k = \{i | C(\boldsymbol{x}_i) = k\}$ we denote the set of indices of the data points in cluster k, and with $n_k = |G_k|$ the number of samples in cluster k.

Iterative refinement:

1) estimate the group means by

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i \in G_k} \boldsymbol{x}_i$$

2) update group assignment: each data point $\boldsymbol{x}_i$ is (re)assigned to the group k with the nearest $\hat{\boldsymbol{\mu}}_k$ (in terms of the Euclidean norm). Specifically, the assignment $C(\boldsymbol{x}_i)$ is updated to

$$\begin{aligned} C(\boldsymbol{x}_i) &= \arg\min_k |\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k|_2 \\ &= \arg\min_k (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^T (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k) \end{aligned}$$

Steps 1 and 2 are repeated until the algorithm converges (or an upper limit of repeats is reached).

3.3.3 Properties

Despite its simplicity K-means is a surprisingly effective clustering algorithms. The final clustering depends on the initialisation so it is often useful to run K-means several times with different starting allocations of the data points.

As a result of the way the clusters are assigned in K-means this leads to cluster boundaries that form a Voronoi tesselation (cf. https://en.wikipedia.org/wiki/Voronoi_diagram) around the cluster means.

Below we will also discuss the connection of K-means with probabilistic clustering using Gaussian mixture models.

3.3.4 Choosing the number of clusters

Once the K-means clustering has been obtained it is insightful to compute:

a) the total within-group sum of squares SSW (tot.withinss), or total unexplained sum of squares:

$$SSW = \sum_{k=1}^{K} \sum_{i \in G_k} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^T (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)$$

This quantity decreases with K and is zero for $K = n$. The K-means algorithm tries to minimise this quantity but it will typically only find a local minimum rather than the global one.

b) the between-group sum of squares SSB (betweenss), or explained sum of squares:

$$SSB = \sum_{k=1}^{K} n_k (\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_0)^T (\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_0)$$

where $\hat{\boldsymbol{\mu}}_0 = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i = \frac{1}{n} \sum_{k=1}^{K} n_k \hat{\boldsymbol{\mu}}_k$ is the global mean of the samples. SSB increases with the number of clusters K until for $K = n$ it becomes equal to

c) the total sum of squares

$$SST = \sum_{i=1}^{n} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_0)^T (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_0) \, .$$

By construction $SST = SSB + SSW$ for any K (i.e. SST is a constant independent of K).

If divide the sum of squares by the sample size n we get $T = \frac{SST}{n}$ as the *total variation*, $B = \frac{SSW}{n}$ as the *explained variation* and $W = \frac{SSW}{n}$ as the total *unexplained variation* , with $T = B + W$.

For deciding on the optimal number of clusters we can run K-means for various settings of K and then choose the smallest K for which the explained variation B is not significantly worse compared to a model with substantially larger number of clusters (see example below).

3.3.5 K-medoids aka PAM

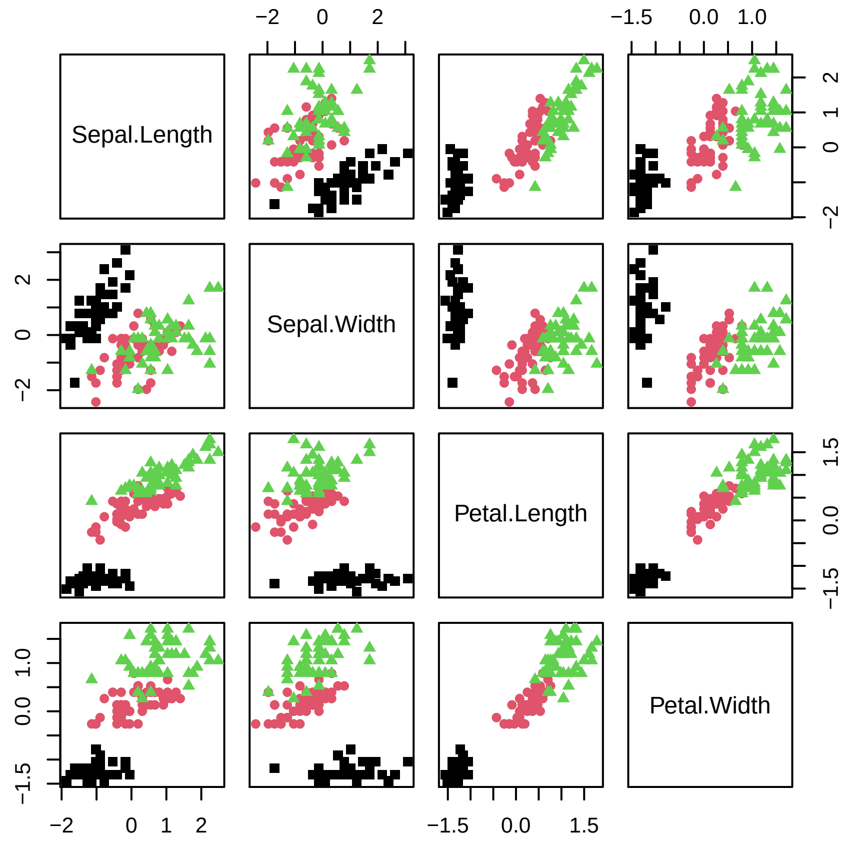
A closely related clustering method is K-medoids or PAM ("Partitioning Around Medoids").

This works exactly like K-means, only that

- instead of the estimated group means $\hat{\boldsymbol{\mu}}_k$ one member of each group is selected as its representative (the socalled "medoid")
- instead of squared euclidean distance other dissimilarity measures are also allowed.

3.3.6 Application of K-means to Iris data

Scatter plots of Iris data:



The R output from a K-means analysis with true number of clusters specified ($K = 3$) is:

```
## K-means clustering with 3 clusters of sizes 47, 50, 53
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1   1.13217737  0.08812645    0.9928284   1.0141287
## 2  -1.01119138  0.85041372   -1.3006301  -1.2507035
## 3  -0.05005221 -0.88042696    0.3465767   0.2805873
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 3 3 3 1 3 3 3 3 3 3 3 3 1 3 3 3 3 1 3 3 3
##  [75] 3 1 1 1 3 3 3 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 1 1 1 3 1 1 1 1
## [112] 1 1 3 3 1 1 1 1 3 1 3 1 3 1 1 3 1 1 1 1 1 1 3 3 1 1 1 3 1 1 1 3 1 1 1 3 1
## [149] 1 3
##
## Within cluster sum of squares by cluster:
## [1] 47.45019 47.35062 44.08754
##  (between_SS / total_SS =  76.7 %)
##
## Available components:
```

```
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

The corresponding total within-group sum of squares (SSW, tot.withinss) and the between-group sum of squares (SSB, betweenss) are:

```
kmeans.out$tot.withinss
```

```
## [1] 138.8884
```

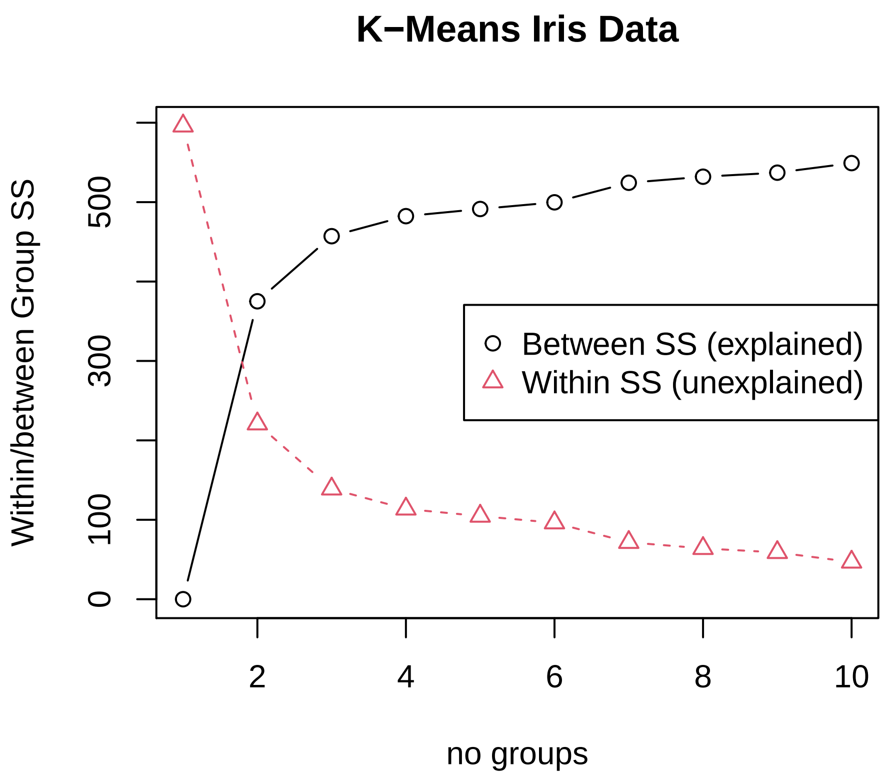
```
kmeans.out$betweenss
```

```
## [1] 457.1116
```

By comparing with the known class assignments we can find out about the accuracy of K-means in this example:

```
##
## L.iris        1  2  3
##   setosa      0 50  0
##   versicolor 11  0 39
##   virginica  36  0 14
```

For choosing K we run K-means several times and compute within and between cluster variation in dependence of K:



Thus, $K = 3$ clusters seem appropriate since the the explained variation does not significantly improve (and the unexplained variation does not significantly decrease) with a further increase of the number of clusters.

3.4 Mixture models

3.4.1 Finite mixture model

- K groups / classes / categories, with the number K specified and finite
- each class $k \in \{1, \dots, K\}$ is modeled by its own distribution F_k with own parameters $\boldsymbol{\theta}_k$.
- density in each class: $f_k(\boldsymbol{x}) = f(\boldsymbol{x}|k)$ with $k \in 1, \dots, K$
- mixing weight of each class: $\Pr(k) = \pi_k$ with $\sum_{k=1}^K \pi_k = 1$
- joint density $f(\boldsymbol{x}, k) = f(\boldsymbol{x}|k)\Pr(k) = f_k(\boldsymbol{x})\pi_k$

This results in the mixture density / marginal density

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x})$$

Very often one uses **multivariate normal components** $f_k(\boldsymbol{x}) = N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ $\Longrightarrow$ **Gaussian mixture model** (GMM)

Mixture models are fundamental not just in clustering but for many other applications (e.g. classification).

Note: don't confuse *mixture model* with *mixed model* (=random effects regression model)

3.4.2 Decomposition of covariance and total variation

Just like in regression you can decompose the variance into an explained and unexplained part.

The conditional means and variances for each class are $\text{E}(\boldsymbol{x}|k) = \boldsymbol{\mu}_k$ and $\text{Var}(\boldsymbol{x}|k) = \boldsymbol{\Sigma}_k$, and the probability of class k is given by $\Pr(k) = \pi_k$. Using the law of total expectation we can therefore obtain the mean of the mixture density as follows:

$$\begin{aligned}\text{E}(\boldsymbol{x}) &= \text{E}(\text{E}(\boldsymbol{x}|k)) \\ &= \sum_{k=1}^{K} \pi_k \boldsymbol{\mu}_k \\ &= \boldsymbol{\mu}_0\end{aligned}$$

Similarly, using the law of total variance we compute the marginal variance:

$$\underbrace{\text{Var}(\boldsymbol{x})}_{\text{total}} = \underbrace{\text{Var}(\text{E}(\boldsymbol{x}|k))}_{\text{explained / between-group}} + \underbrace{\text{E}(\text{Var}(\boldsymbol{x}|k))}_{\text{unexplained / within-group}}$$

$$\boldsymbol{\Sigma}_0 = \sum_{k=1}^{K} \pi_k (\boldsymbol{\mu}_k - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_k - \boldsymbol{\mu}_0)^T + \sum_{k=1}^{K} \pi_k \boldsymbol{\Sigma}_k$$

The total variation is given by the trace of the covariance matrix, yielding empirical estimates

$$T = \mathrm{Tr}\left(\hat{\boldsymbol{\Sigma}}_0\right) = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_0)^T(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_0)$$

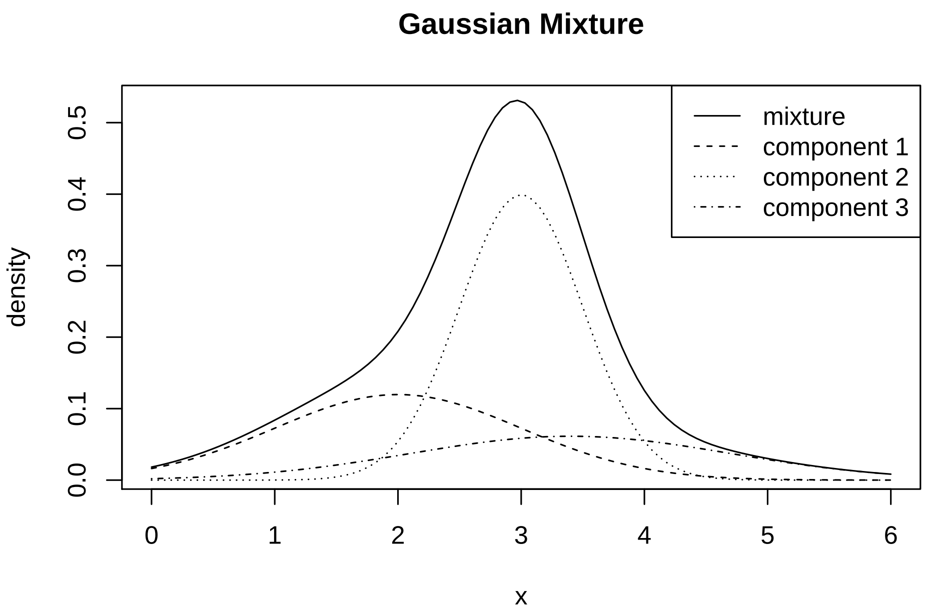
$$B = \mathrm{Tr}\left(\sum_{k=1}^{K}\hat{\pi}_k(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_0)(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_0)^T\right) = \frac{1}{n}\sum_{k=1}^{K} n_k(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_0)^T(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_0)$$

$$W = \mathrm{Tr}\left(\sum_{k=1}^{K}\hat{\pi}_k\hat{\boldsymbol{\Sigma}}_k\right) = \frac{1}{n}\sum_{k=1}^{K}\sum_{i\in G_k}(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^T(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)$$

Compare the above with the $T = B + W$ decomposition of total variation in K-means!

3.4.3 Example of mixture of three univariate normal densities:

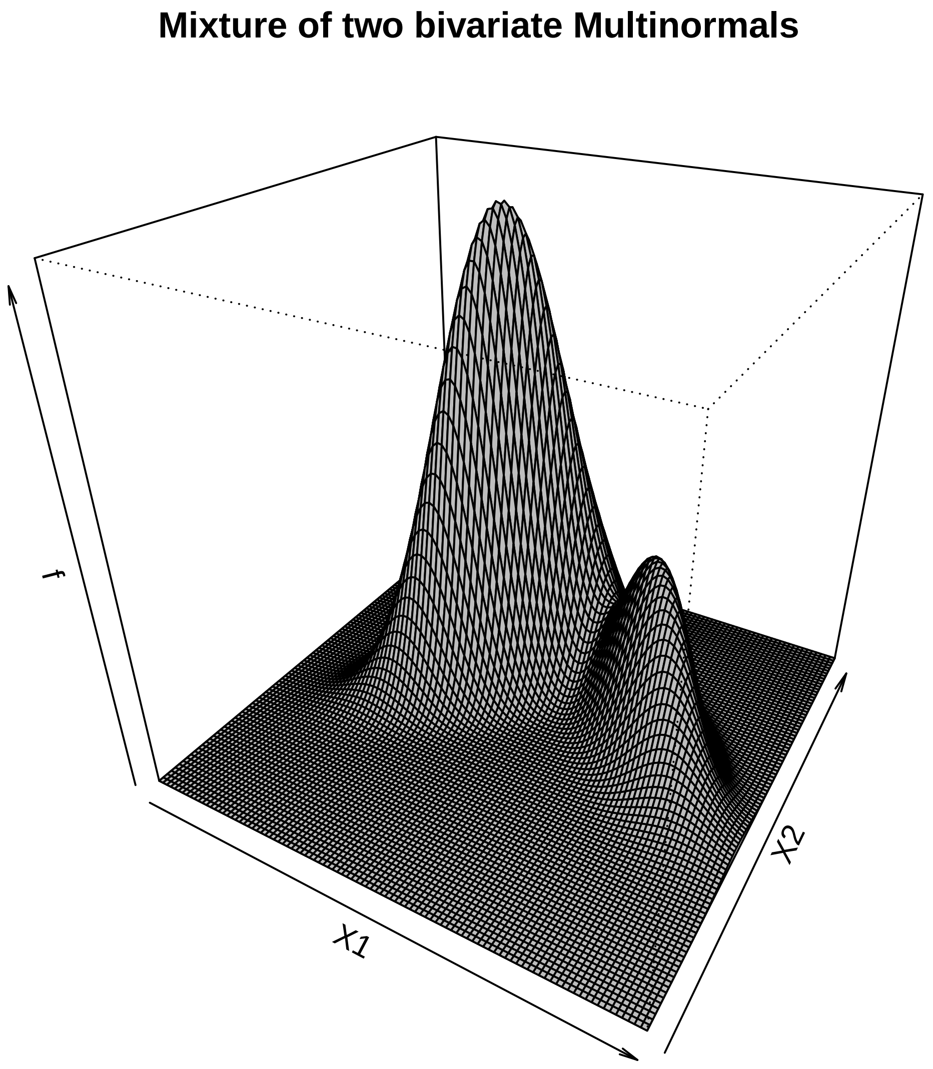
$$f(x) = 0.3\,N(2, 1^2) + 0.5\,N(3, 0.5^2) + 0.2\,N(3.4, 1.3^2)$$



In this case it is clear already by visual inspection that the three subcomponents will not be identifiable.

3.4.4 Example of a mixture of two bivariate normal densities

$$f(\boldsymbol{x}) = 0.7\,N_2\left(\begin{pmatrix}-1\\1\end{pmatrix}, \begin{pmatrix}1 & 0.7\\0.7 & 1\end{pmatrix}\right) + 0.3\,N_2\left(\begin{pmatrix}2.5\\0.5\end{pmatrix}, \begin{pmatrix}1 & -0.7\\-0.7 & 1\end{pmatrix}\right)$$



3.4.5 Sampling from a mixture model and latent allocation variable formulation

Assuming we know how to sample from the components $f_k(\boldsymbol{x})$ of the mixture model it is straightforward to set up a procedure for sampling from the mixture $f(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x})$.

This is done in a two-step generative process:

1. draw from categorical distribution with parameters $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_k)^T$:

   $$\boldsymbol{z} \sim \text{Categ}(\boldsymbol{\pi})$$

   the vector $\boldsymbol{z} = (z_1, \ldots, z_K)^T$ indicating the group allocation. The group index k is given by $\{k : z_k = 1\}$.

2. Subsequently, sample from the component k selected in step 1:

$$\boldsymbol{x} \sim F_k$$

This two-stage approach is also called *latent allocation variable formulation* of a mixture model, with $\boldsymbol{z}$ (or equivalently k) being the latent variable.

The two-step process needs to repeated for each sample drawn from the mixture (i.e. every time a new latent variable $\boldsymbol{z}$ is generated).

In probabilistic clustering the aim is to infer the state of $\boldsymbol{z}$ for all observed samples.

3.4.6 Predicting the group allocation of a given sample

If we know the mixture model and its components we can predict the probability that an observation $\boldsymbol{x}$ falls in group k using Bayes theorem:

$$z_k = \Pr(k|\boldsymbol{x}) = \frac{\pi_k f_k(\boldsymbol{x})}{f(\boldsymbol{x})}$$

Thus, assuming we can calculate this probability we can **perform probabilistic clustering** by assigning each sample to the class with the largest probability. Unlike in algorithmic clustering, we also get an impression of the uncertainty of the class assignment, since for each sample $\boldsymbol{x}$ get the vector

$$\boldsymbol{z} = (z_1, \ldots, z_K)^T$$

and thus can see if there are several classes with similar assignment probability. This will be the case, e.g., if $\boldsymbol{x}$ lies near the boundary between two classes. Note that $\sum_{k=1}^K z_k = 1$.

3.4.7 Variation 1: Infinite mixture model

It is possible to construct mixture models with infinitely many components!

Most commonly known example is Dirichlet process mixture model (DPM):

$$\sum_{k=1}^{\infty} \pi_k f_k(\boldsymbol{x})$$

with $\sum_{k=1}^{\infty} \pi_k = 1$ and where the weight π_k are taken from a infinitely dimensional Dirichlet distribution (=Dirichlet process).

DPMs are useful for clustering since with them it is not necessary to determine the number of clusters a priori (since it by definition has infinitely many!). Instead, the number of clusters is a by-product of the fit of the model to observed data.

Related: **"Chinese restaurant process"** - https://en.wikipedia.org/wiki/Chinese_restaurant_process

This describes an algorithm for the allocation process of samples ("persons") to the groups ("restaurant tables") in a DPM.

See also **"stick-breaking process"**: https://en.wikipedia.org/wiki/Dirichlet_process#The_stick-breaking_process

3.4.8 Variation 2: Semiparametric mixture model with two classes

A very common model is the following two-component univariate mixture model

$$f(x) = \pi_0 f_0(x) + (1-\pi_0) f_A(x)$$

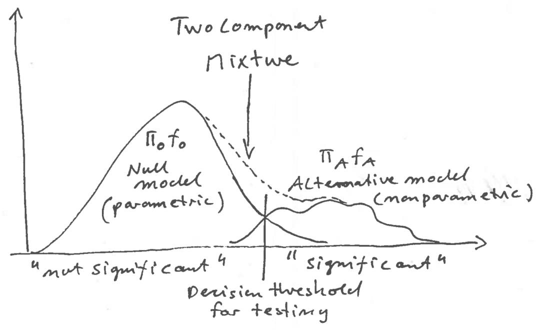
- f_0: null model, typically parametric such as normal distribution
- f_A: alternative model, typically nonparametric
- π_0: prior probability of null model

Using Bayes theorem this allows to compute probability that an observation x belongs to the null model:

$$\text{Pr}(\text{Null}|x) = \frac{\pi_0 f_0(x)}{f(x)}$$

This is called the *local false discovery rate*.

The semi-parametric mixture model is the foundation for statistical testing which is based on defining decision thresholds to separate null model ("not significant") from alternative model ("significant"):



See the lecture notes for Statistical Methods MATH20802 (year 2) for more details.

3.5 Fitting mixture models to data

3.5.1 Direct estimation of mixture model parameters

Given data matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$ with observations from n samples we would like to fit the mixture model $f(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x})$ and learn its parameters $\boldsymbol{\theta}$, for example by maximising the corresponding marginal log-likelihood function with regard to $\boldsymbol{\theta}$:

$$\log L(\boldsymbol{\theta}|\boldsymbol{X}) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}_i) \right)$$

For a Gaussian mixture model the parameters are $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K\}$.

However, in practise evaluation of this likelihood function may be difficult, in part due to the form of the log-likelihood function (note the sum inside the logarithm), but also due to its singularities and non-identiability problems.

The above log-likelihood function is also called the *observed data* log-likelihood, or the *incomplete data* log-likelihood, in contrast to the *complete data* log-likelihood described further below.

3.5.2 Estimate mixture model parameters using the EM algorithm

The mixture model may be viewed as an *incomplete* or *missing* data problem: here the missing data are the group allocation $\boldsymbol{k} = (k_1, \dots, k_n)^T$ belonging to each sample $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$.

If we would know which sample comes from which group the estimation of the parameters $\boldsymbol{\theta}$ would indeed be straightforward using the so-called *complete data log-likelihood* based on the joint distribution $f(\boldsymbol{x}, k) = f_k(\boldsymbol{x})\pi_k$

$$\log L(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{k}) = \sum_{i=1}^{n} \log \left(\pi_{k_i} f_{k_i}(\boldsymbol{x}_i) \right)$$

The idea of the EM algorithm (Dempster et al. 1977) is to exploit the simplicity of the complete data likelihood and to obtain estimates of $\boldsymbol{\theta}$ by first finding the probability distribution z_{ik} of the latent variable k_i, and then using this distribution to compute and optimise the corresponding expected complete-data log-likelihood. Specifically, the z_{ik} contain the *probabilities* of each class for each sample i and thus provide a *soft assignment* of classes rather that a $0/1$ *hard assignment* (as in the K-means algorithm or in the generative latent variable view of mixture models).

In the EM algorithm we iterate between the

1) estimation the probabilistic distribution z_{ik} for the group allocation latent parameters using the current estimate of the parameters $\boldsymbol{\theta}$ (obtained in step 2)
2) maximisation of the expected complete data log-likelihood to estimate the parameters $\boldsymbol{\theta}$. The expectation is taken with regard to the distribution z_{ik} (obtained in step 1).

Specifically, the EM algorithm applied to model-based clustering proceeds as follows:

1) Initialisation: Start with a guess of the parameters $\boldsymbol{\theta}^{(1)}$, then continue with "E" Step, Part A. Alternatively, start with a guess of $z_{ik}^{(1)}$, then continue

with "E" Step, Part B. The initialisation may be derived from some prior information, e.g., from running K-means, or simply be at random.

2) **E "expectation" step** — Part A: Use Bayes' theorem to compute new probabilities of allocation for all the samples $\boldsymbol{x}_i$:

$$z_{ik}^{(b+1)} \leftarrow \frac{\pi_k f_k(\boldsymbol{x}_i)}{f(\boldsymbol{x}_i)}$$

Note that to obtain $z_{ik}^{(b+1)}$ the current value $\boldsymbol{\theta}^{(b)}$ of the parameters is required.
— Part B: Construct the expected complete data log-likelihood function using the weights $z_{ik}^{(b+1)}$:

$$Q^{(b+1)}(\boldsymbol{\theta}|\boldsymbol{X}) = \sum_{i=1}^{n} \sum_{k=1}^{k} z_{ik}^{(b+1)} \log\left(\pi_k f_k(\boldsymbol{x}_i)\right)$$

3) **M "maximisation" step** — Maximise the expected complete data log-likelihood to update the mixture model parameters $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^{(b+1)} \leftarrow \arg\max_{\boldsymbol{\theta}} Q^{(b+1)}(\boldsymbol{\theta}|\boldsymbol{X})$$

4) Repeat with "E" Step until convergence of parameters $\boldsymbol{\theta}^{(b)}$ of the mixture model.

It can be shown that the output $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)}, \ldots$ of the EM algorithm converges to the estimate $\hat{\boldsymbol{\theta}}$ found when maximising the marginal log-likelihood. Since maximisation of the expected complete data log-likelihood is often much easier (and analytically tractable) than maximisation of the observed data log-likelihood function the EM algorithm is the preferred approach in this case.

To avoid singularities in the expected log-likelihood function we may wish to adopt a Bayesian approach (or use regularised/penalised ML) for estimating the parameters in the M-step.

3.5.3 EM algorithm for multivariate normal mixture model

For a GMM the EM algorithm can be written down analytically:

E-step:

$$z_{ik} = \frac{\hat{\pi}_k N(\boldsymbol{x}_i|\hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}}_k)}{f(\boldsymbol{x}_i)}$$

M-step:

$$\hat{n}_k = \sum_{i=1}^{n} z_{ik}$$

$$\hat{\pi}_k = \frac{\hat{n}_k}{n}$$

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{\hat{n}_k} \sum_{i=1}^{n} z_{ik} \boldsymbol{x}_i$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{1}{\hat{n}_k} \sum_{i=1}^{n} z_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T$$

Note that the estimators $\hat{\boldsymbol{\mu}}_k$ and $\hat{\boldsymbol{\Sigma}}_k$ are weighted versions of the usual empirical estimators (with weights z_{ik} being the soft assignment of classes resulting from the Bayesian updating).

3.5.4 Connection with K-means clustering method

The K-means algorithm is very closely related to probabilistic clustering with GMMS.

Specifically, it is straightforward to see that *K-means is effectively equivalent to fitting a Gaussian mixture model with the probabilities π_k of all classes identical and with the covariances $\boldsymbol{\Sigma}_k$ all of the form $\sigma^2 \boldsymbol{I}$*, i.e. all classes have the same diagonal covariance with identical variances. However, note that in K-means the class allocations are hard, whereas in GMMs they are soft. Thus, GMM-based clustering can be viewed as a probabilistic generalisation of K-means clustering!

See also Worksheet 7 where it is shown that the Bayesian update rule assuming equal probability for the classes together with the above covariance leads to the class assignment rule used in K-means.

3.5.5 Choosing the number of classes

Since GMMs operate in a likelihood framework we can use penalised likelihood model selection criteria to choose among different models (i.e. GMMs with different numbers of classes).

The most popular choices are AIC (Akaike Information Criterion) and BIC (Bayesian Information criterion) defined as follows:

$$\text{AIC} = -2 \log L + 2K$$

$$\text{BIC} = -2 \log L + K \log(n)$$

Instead of maximising the log-likehood we minimise AIC and BIC.

Note that in both criteria more complex models with more parameters (in this case groups) are penalised over simpler models in order to prevent overfitting.

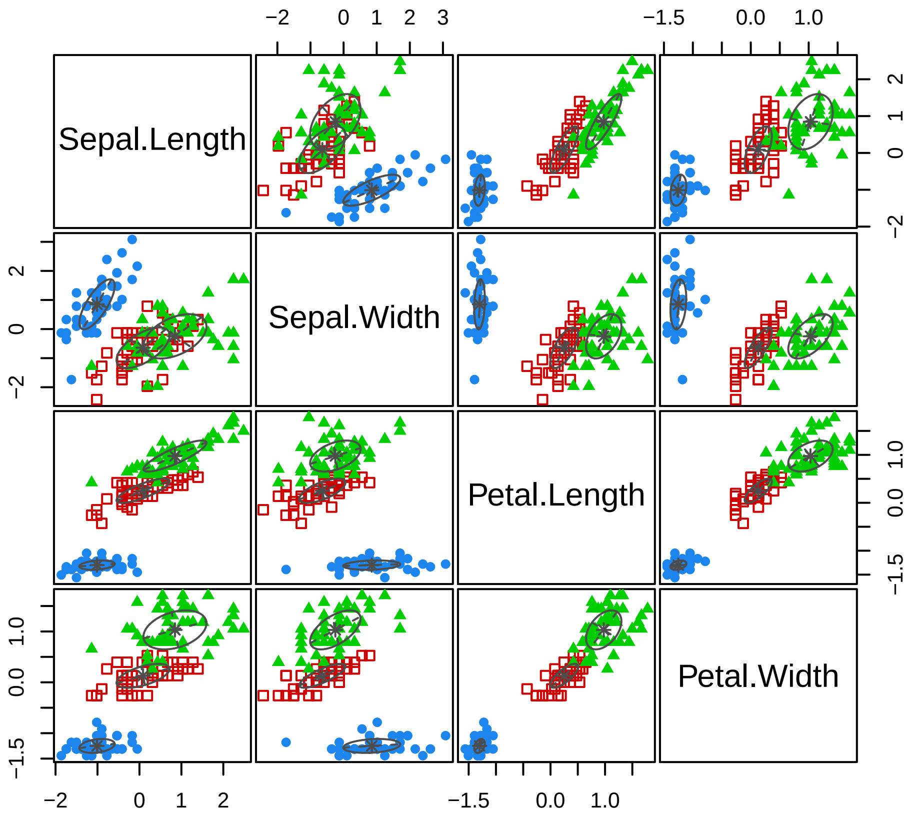
$\Longrightarrow$ find optimal number of groups K.

Another way of choosing optimal numbers of clusters is by cross-validation (see later chapter on supervised learning).

3.5.6 Application of GMMs to Iris flower data

We now explore the application of GMMs to the Iris flower data set we also investigated with PCA and K-means.

First, we run GMM with 3 clusters:



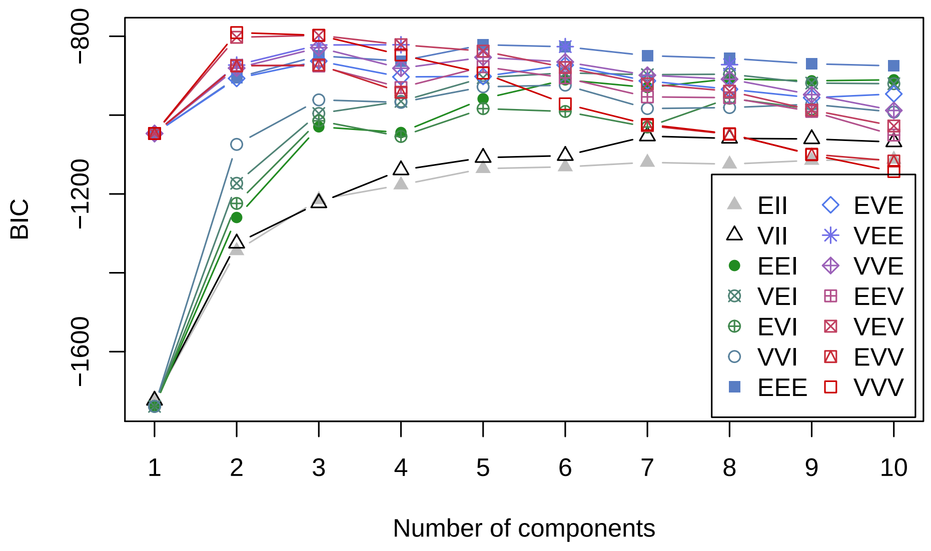
The GMM has a substantially lower misclassification error compared to K-means with the same number of clusters:

```
table(gmm3$classification, L.iris)
```

```
##    L.iris
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         45         0
##   3      0          5        50
```

Note that in the R software "mclust" to analyse GMMs the BIC criterion is defined with the opposite sign ($\text{BIC}_{\text{mclust}} = 2 \log L - K \log(n)$), thus we need to find the *maximum* value rather than the smallest value.

If we optimise BIC we find that the model with highest $\text{BIC}_{\text{mclust}}$ is a model with 2 clusters but the model with 3 cluster has nearly as good a BIC:


BIC
−800
−1200
−1600
1
2
3
4
5
6
7
8
9
10
Number of components
EII
VII
EEI
VEI
EVI
VVI
EEE
EVE
VEE
VVE
EEV
VEV
EVV
VVV

Chapter 4

Classification / supervised learning

4.1 Introduction

4.1.1 Supervised learning vs. unsupervised learning

Aim in **Unsupervised learning:**

For data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ find classes / labels groups $y_1, \ldots, y_n$ attached to each sample $\boldsymbol{x}_i$.

For example, if x_2 is assigned the label $y = 5$ this means sample 2 belongs to class 5.

If y is discrete unsupervised learning is called *clustering*.

Aim in **Supervised learning:**

We have *training data* available *with* labels: $\{\boldsymbol{x}_1^{train}, y_1^{train}\}$, ..., $\{\boldsymbol{x}_n^{train}, y_n^{train}\}$. Each $\boldsymbol{x}_i^{train} = (x_{i1}^{train}, \ldots, x_{id}^{train})^T$ contains the observations of d properties (predictor variables) of the sample i.

The training data (observations of predictors variables and response/labels) are used to determine a predictor function $f(\boldsymbol{x})$. This function is then used to predict the *unknown* labels / class y^{test} of new data $\boldsymbol{x}^{test}$ in a probabilistic fashion, i.e. with probabilities attached to the predicted outcome.

Thus, in contrast to unsupervised learning, supervised learning includes a training step with actual data with known labels.

For y discrete supervised learning is called *classification*.

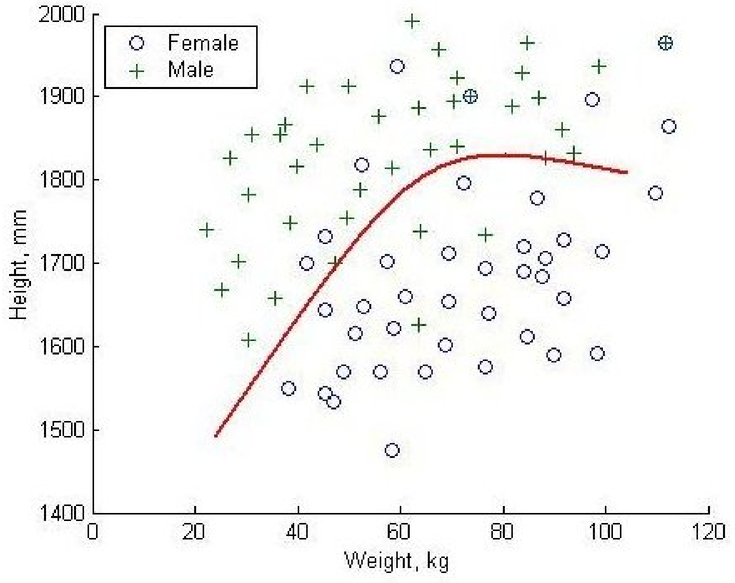
Note the similarity to regression (especially for continuous response y)! In fact, supervised learning *is* (generalised) regression.

4.1.2 Terminology

The function $f(x)$ that predicts the class y is called a *classifier*. There are many types of classifiers, we focus here primarily on probabilistic classifiers (i.e. those that output the predicted class along with a probability). In supervised learning the classifier is learned from the training data.

The challenge is to find a classifier that explains the current training data well *and* that also generalises well to future unseen data. Note that it is relatively easy to find a predictor that explains the training data but especially in high dimensions (i.e. with many predictors) there is often overfitting and then the predictor does not generalise well!

The classifier describes the decision boundary between the classes:



In general, simple decision boundaries are preferred over complex decision boundaries to avoid overfitting!

Some commonly used probabilistic methods for classifications: QDA (quadratic discriminant analysis), LDA (linear discriminant analysis), DDA (diagonal discriminant analysis), Naive Bayes classification, logistic regression, GPs (Gaussian processes).

Common non-probabilistic methods include: SVM (support vector machine), logistic regression, random forest, neural networks.

Depending on how the classifiers are trained there are many variations of the above methods, e.g. Fisher discriminant analysis, regularised LDA, shrinkage disciminant analysis etc.

4.2 Bayesian discriminant rule or Bayes classifier

Same setup as with mixture models:

- K groups with K prespecified
- each group has its own distribution F_k with own parameters $\boldsymbol{\theta}_k$
- the density of each class is $f_k(\boldsymbol{x}) = f(\boldsymbol{x}|k)$.
- prior probability of group k is $\text{Pr}(k) = \pi_k$ with $\sum_{k=1}^K \pi_k = 1$
- marginal density is the mixture $f(\boldsymbol{x}) = \sum_{k=1}^K \pi_k f_k(\boldsymbol{x})$

The posterior probability of group k is then

$$\text{Pr}(k|\boldsymbol{x}) = \frac{\pi_k f_k(\boldsymbol{x})}{f(\boldsymbol{x})}$$

The *discriminant function* is the logarithm of the posterior probability:

$$d_k(\boldsymbol{x}) = \log \text{Pr}(k|\boldsymbol{x}) = \log(\pi_k) + \log(f_k(\boldsymbol{x})) - \log(f(\boldsymbol{x}))$$

Since we use d_k to compare the different classes k we can simplify the discriminant function by dropping all constant terms that do not depend on k - in the above $\log(f(\boldsymbol{x}))$. Hence we get for the Bayes discriminant function

$$d_k(\boldsymbol{x}) = \log(\pi_k) + \log(f_k(\boldsymbol{x}))$$

This provides us with the probability of each class given the test data $\boldsymbol{x}$. For subsequent "hard" classification we need to use a decision rule, such as selecting the group $\hat{k}$ for that which the group probability / value of discriminant function is maximised:

$$\hat{k} = \arg\max_k d_k(\boldsymbol{x}) \,.$$

You have already encountered the Bayes classifier in the EM algorithm to predict the state of the latent variables. In a simplied versions it also plays a role in the K-means algorithm.

The Bayes classifier reduces to the likelihood classifier (see example class 3) if one assumes that prior probabilities π_k do not depend on k (and hence are uniform).

4.3 Normal Bayes classifier

4.3.1 Quadratic discriminant analysis (QDA) and Gaussian assumption

Quadratic discriminant analysis (QDA) is a special case of the Bayes classifier when all densities are multivariate normal with $f_k(\boldsymbol{x}) = N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.

This leads to the discriminant function for QDA:

$$d_k^{QDA}(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k) - \frac{1}{2}\log\det(\boldsymbol{\Sigma}_k) + \log(\pi_k)$$

There are a number of noteworthy things here:

- Again terms are dropped that do not depend on k, such as $-\frac{d}{2}\log(2\pi)$.
- Note the appearance of the Mahalanobis distance between $\boldsymbol{x}$ and $\boldsymbol{\mu}_k$ in the last term — recall $d^{Mahalanobis}(\boldsymbol{x}, \boldsymbol{\mu}|\boldsymbol{\Sigma}) = (\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$.
- The **QDA discriminant function is quadratic in $\boldsymbol{x}$** - hence its name! This implies that the **decision boundaries for QDA classification are quadratic** (i.e. parabolas in two dimensional settings). Thus **QDA is a non-linear classification method**!

For Gaussian models specifically it can useful be to multiply the discriminant function by -2 to get rid of the factor $-\frac{1}{2}$, but note that in that case we then need to look for the minimum of the simplified discriminant function rather than the maximum:

$$d_k^{QDA(v2)}(\boldsymbol{x}) = (\boldsymbol{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) + \log\det(\boldsymbol{\Sigma}_k) - 2\log(\pi_k)$$

In the literature you will find both versions of Gaussian discriminant functions so you need to check carefully which convention is used. In the following we will use the first version only.

4.3.2 Linear discriminant analysis (LDA)

LDA is a special case of QDA, with the assumption of common overall covariance across all groups: $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$.

This leads to a simplified discriminant function:

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

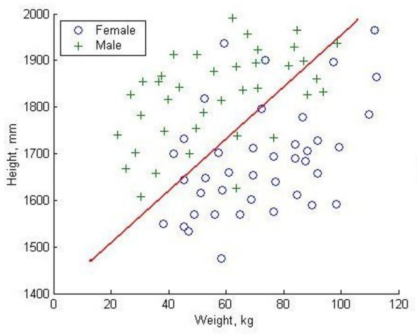
Note that term containing the log-determinant is now gone, and that LDA is essentially now a method that tries to minimize the Mahalanobis distance (while taking also into account the prior class probabilities).

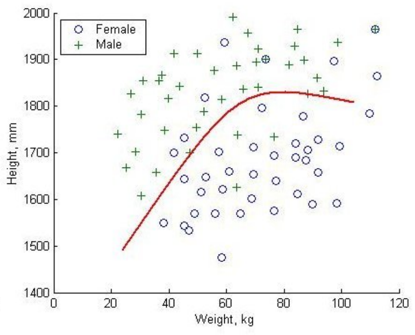
The above function can be further simplified, by noting that the quadratic term $\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$ does not depend on k and hence can be dropped:

$$\begin{aligned} d_k^{LDA}(\boldsymbol{x}) &= \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log(\pi_k) \\ &= \boldsymbol{b}^T\boldsymbol{x} + a \end{aligned}$$

Thus, the **LDA discriminant function is linear in $\boldsymbol{x}$, and hence the resulting decision boundaries are linear** as well (i.e. straight lines in two-dimensional settings). **LDA is a linear classification method**.

Comparison of decision boundary of LDA (left) compared with QDA (right):





Note that logistic regression (cf. GLM module) takes on exactly the above linear form and is indeed closely linked with the LDA classifier.

4.3.3 Diagonal discriminant analysis (DDA)

In DDA we assume the same setting as LDA, but now we simplify even further by assuming a **diagonal covariance** containing only the variances:

$$\boldsymbol{\Sigma} = \boldsymbol{V} = \begin{pmatrix} \sigma_1^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_d^2 \end{pmatrix}$$

This simplifies the inversion of $\boldsymbol{\Sigma}$ as

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{V}^{-1} = \begin{pmatrix} \sigma_1^{-2} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_d^{-2} \end{pmatrix}$$

and leads to the discriminant function

$$\begin{aligned} d_k^{DDA}(\boldsymbol{x}) &= \boldsymbol{\mu}_k^T \boldsymbol{V}^{-1} \boldsymbol{x} - \frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{V}^{-1} \boldsymbol{\mu}_k + \log(\pi_k) \\ &= \sum_{j=i}^{d} \frac{\mu_{k,j} x_j - \mu_{k,j}^2/2}{\sigma_d^2} + \log(\pi_k) \end{aligned}$$

As special case of LDA, the **DDA classifier is a linear classifier**.

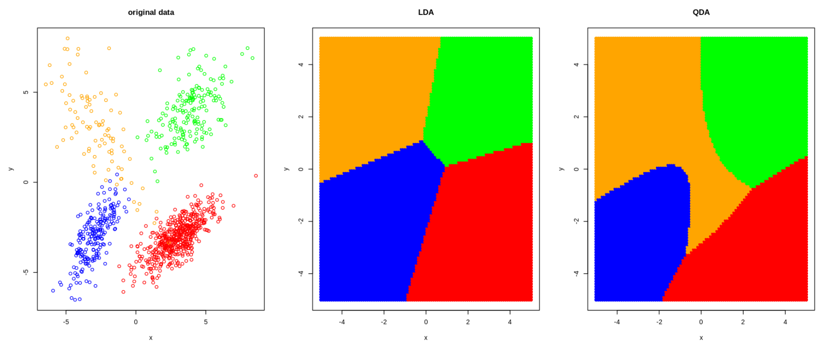
The **Bayes classifier** (using any distribution) **assuming uncorrelated predictors** is also known as the **naive Bayes classifier**.

Hence, **DDA is a naive Bayes classifier** assuming underlying Gaussian distributions.

However, don’t let you misguide because of the name “naive”: in fact DDA and other “naive” Bayes classfier are often very effective classifiers, especially in high-dimensional settings!

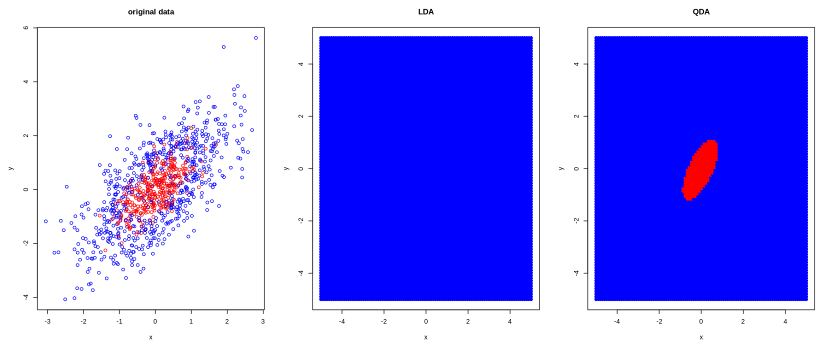
4.3.4 Comparison of decision boundaries: LDA vs. QDA

Non-nested case ($K = 4$)



Note the linear decision boundaries for LDA!

Nested case ($K = 2$):



There is no linear classifier that can seperate two nested classes!

4.4 The training step — learning QDA, LDA and DDA classifiers from data

In order to predict the class for new data using any of the above discriminant functions we need to first learn the underlying parameters from the training data $\boldsymbol{x}_i^{\text{train}}$ and y_i^{train}:

- For QDA, LDA and DDA we need to learn $\pi_1, \ldots, \pi_K$.
- For QDA we additionally require $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K$
- For LDA we need $\boldsymbol{\Sigma}$
- For DDA we estimate $\sigma_1^2, \ldots, \sigma_d$.

To obtain the above parameter estimates we use the labels y_i^{train} to sort the samples $\boldsymbol{x}_i^{\text{train}}$ into the corresponding classes, and then apply the usual estimators. Let $G_k = \{i : y_i^{\text{train}} = k\}$ be the set of all indices of training sample belonging to group k.

Then to obtain the ML estimate of the group means $k = 1, \ldots, K$ we compute

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i \in g_k} \boldsymbol{x}_i^{\text{train}}$$

Note this differs (for $K > 1$) from the the estimate of the global mean $\boldsymbol{\mu}_0$ that we get if we were to ignore the group labels (i.e. if we assume there is only a single class):

$$\hat{\boldsymbol{\mu}}_0 = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i^{\text{train}}$$

In order to get the ML estimate of the pooled variance $\boldsymbol{\Sigma}$ we use

$$\widehat{\boldsymbol{\Sigma}}^{ML} = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in g_k} (\boldsymbol{x}_i^{\text{train}} - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i^{\text{train}} - \hat{\boldsymbol{\mu}}_k)^T$$

Note that the pooled variance $\boldsymbol{\Sigma}$ (with $K > 1$) differs (substantially!) from the global variance Σ_0 that results from ignoring class labels (or in the single class case):

$$\widehat{\boldsymbol{\Sigma}}_0^{ML} = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i^{\text{train}} - \hat{\boldsymbol{\mu}}_0)(\boldsymbol{x}_i^{\text{train}} - \hat{\boldsymbol{\mu}}_0)^T$$

You will recognise the above from the variance decomposion in mixture models, with $\boldsymbol{\Sigma}_0$ being the total variance and the pooled $\boldsymbol{\Sigma}$ the unexplained/with-in group variance.

Overall, the total number of parameters to be estimated when learning the discriminant functions from training data is as follows:

- QDA: $K + Kd + K\frac{d(d-1)}{2}$
- LDA: $K + d + \frac{d(d-1)}{2}$
- DDA: $K + d$

We also need to make sure that the estimated covariance matrices are all positive definite (which for DDA is automatically guaranteed if all variances are positive).

If d (and K) is small and the number of available samples n is large then we can use maximum likelihood as sketched above to estimate the parameters.

However, if d is large compared to the sample size then the numbers of parameters to estimate grows very quickly. Especially QDA but also LDA is quite data hungry and ML estimation becomes an ill-posed problem. We thus need to use a regularised estimator for the covariance(s) such as penalised ML, Bayes, shrinkage estimator, cf. Section 1.5 and the Statistical Methods module.

Also, to reduce the number of parameters it is advised to used either LDA or DDA in rather than QDA. Often this has a beneficial effect because a simpler model will generalise better and avoid overfitting.

In the application to high-dimensional data we will employ in the computer labs a regularised version of LDA and DDA using the Stein-type shrinkage estimator of the covariance discussed in Section 1.5. Both are is implemented in the R package "sda".

4.5 Goodness of fit and variable selection

As in linear regression (cf. "Statistical Methods" module) we are interested in finding out whether the fitted mixture model is an appropriate model, and which particular predictor(s) x_j from $\boldsymbol{x} = (x_1, \ldots, x_d)^T$ are responsible prediction the outcome, i.e. for categorizing a sample into group k.

In order to study these problem it is helpful to rewrite the discriminant function to highlight the influence (or importance) of each predictor.

We focus on linear methods (LDA and DDA) and first look at the simple case $K = 2$ and then generalise to more than two groups.

4.5.1 LDA with $K = 2$ classes

For two classes using the LDA discriminant rule will choose group $k = 1$ if $d_1^{LDA}(\boldsymbol{x}) > d_2^{LDA}(\boldsymbol{x})$, or equivalently, if

$$\Delta_{12}^{LDA} = d_1^{LDA}(\boldsymbol{x}) - d_2^{LDA}(\boldsymbol{x}) > 0$$

Since $d_k(\boldsymbol{x})$ is the log-posterior (plus/minus identical constants) Δ^{LDA} is in fact the **log-posterior odds of class 1 versus class 2** (see Statistical Methods, Bayesian inference).

The difference Δ_{12}^{LDA} is

$$\underbrace{\Delta_{12}^{LDA}}_{\text{log posterior odds}} = \underbrace{(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{x} - \frac{\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2}{2}\right)}_{\text{log Bayes factor } \log B_{12}} + \underbrace{\log\left(\frac{\pi_1}{\pi_2}\right)}_{\text{log prior odds}}$$

Note that since we only consider simple non-composite models here the log-Bayes factor is identical with the log-likelihood ratio!

The log Bayes factor $\log B_{12}$ is known as the *weight of evidence* in favour of F_1 given $\boldsymbol{x}$. The *expected weight of evidence* assuming $\boldsymbol{x}$ is indeed from F_1 is the Kullback-Leibler discrimination information in favour of group 1, i.e. the KL divergence of from distribution F_2 to F_1:

$$\mathrm{E}_{F_1}(\log B_{12}) = KL(F_1||F_2) = \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) = \frac{1}{2}\Omega^2$$

This yields, apart of a scale factor, a population version of the Hotelling T^2 statistic defined as

$$T^2 = c^2 (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_2)^T \hat{\boldsymbol{\Sigma}}^{-1} (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_2)$$

where $c = (\frac{1}{n_1} + \frac{1}{n_2})^{-1/2} = \sqrt{n\pi_1\pi_2}$ is a sample size dependent factor (for $\text{SD}(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_2)$). T^2 is a measure of fit of the underlying two-component mixture.

Using the whitening transformation with $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x}$ and $\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{\Sigma}^{-1}$ we can rewrite the log Bayes factor as

$$\begin{aligned}\log B_{12} &= \left((\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{W}^T\right) \left(\boldsymbol{W}\left(\boldsymbol{x} - \frac{\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2}{2}\right)\right) \\ &= \boldsymbol{\omega}^T \boldsymbol{\delta}(\boldsymbol{x})\end{aligned}$$

i.e. as the product of two vectors:

- $\boldsymbol{\delta}(\boldsymbol{x})$ is the whitened $\boldsymbol{x}$ (centered around average means) and
- $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_d)^T = \boldsymbol{W}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$ gives the weight of each whitened component $\boldsymbol{\delta}(\boldsymbol{x})$ in the log Bayes factor.

A large positive or negative value of ω_j indicates that the corresponding whitened predictor is relevant for choosing a class, whereas small values of ω_j close to zero indicate that the corresponding ZCA whitened predictor is unimportant. Furthermore, $\boldsymbol{\omega}^T\boldsymbol{\omega} = \sum_{j=1}^d \omega_j^2 = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) = \Omega^2$, i.e. the squared ω_j^2 provide a component-wise decomposition of the overall fit Ω^2.

Choosing ZCA-cor as whitening transformation with $\boldsymbol{W} = \boldsymbol{P}^{-1/2}\boldsymbol{V}^{-1/2}$ we get

$$\boldsymbol{\omega}^{ZCA-cor} = \boldsymbol{P}^{-1/2}\boldsymbol{V}^{-1/2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

A better understanding of $\boldsymbol{\omega}^{ZCA-cor}$ is provided by comparing with the two-sample t-statistic

$$\hat{\boldsymbol{\tau}} = c\hat{\boldsymbol{V}}^{-1/2}(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_2)$$

With $\boldsymbol{\tau}$ the population version of $\hat{\boldsymbol{\tau}}$ we can define

$$\boldsymbol{\tau}^{adj} = \boldsymbol{P}^{-1/2}\boldsymbol{\tau} = c\boldsymbol{\omega}^{ZCA-cor}$$

as correlation-adjusted t-scores (cat scores). With $(\hat{\boldsymbol{\tau}}^{adj})^T \hat{\boldsymbol{\tau}}^{adj} = T^2$ we can see that the cat scores offer a component-wise decomposition of Hotelling's T^2.

Note the choice of ZCA whitening is to ensure that the whitened components are interpretable and stay maximally correlated to the original variables. However, you may also choose, e.g. PCA whitening, in which case the $\boldsymbol{\omega}^T\boldsymbol{\omega}$ provide the variable importance for the PCA whitened variables.

For DDA, which assumes that correlations among predictors vanish, i.e. $\boldsymbol{P} = \boldsymbol{I}_d$, we get

$$\Delta_{12}^{DDA} = \underbrace{\left((\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{V}^{-1/2}\right)}_{c^{-1}\boldsymbol{\tau}^T} \underbrace{\left(\boldsymbol{V}^{-1/2}\left(\boldsymbol{x} - \frac{\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2}{2}\right)\right)}_{\text{centered standardised predictor}} + \log\left(\frac{\pi_1}{\pi_2}\right)$$

Similarly as above, the t-score $\boldsymbol{\tau}$ determines the impact of the standardised predictor in Δ^{DDA}.

Consequently, in DDA we can rank predictors by the squared t-score. Recall that in standard linear regression with uncorrelated predictors we can find the most important predictors by ranking the squared marginal correlations – ranking by (squared) t-scores in DDA is the exact analogy but for discrete response.

4.5.2 Multiple classes

For more than two classes we need to refer to the so-called **pooled centroids formulation** of DDA and LDA (introduced by Tibshirani 2002).

We define the pooled centroid as $\boldsymbol{\mu}_0 = \sum_{k=1}^{K} \pi_k \boldsymbol{\mu}_k$ — this is the centroid if there would be only a single class. The corresponding frequency is $\pi_0 = 1$ and the distribution is called F_0.

The LDA discriminant function for this "group 0" is

$$d_0^{LDA}(\boldsymbol{x}) = \boldsymbol{\mu}_0^T \boldsymbol{\Sigma}^{-1} \boldsymbol{x} - \frac{1}{2} \boldsymbol{\mu}_0^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_0$$

and the log posterior odds for comparison of group k with the pooled group 0 is

$$\begin{aligned} \Delta_k^{LDA} &= d_k^{LDA}(\boldsymbol{x}) - d_0^{LDA}(\boldsymbol{x}) \\ &= \log B_{k0} + \log(\pi_k) \\ &= \boldsymbol{\omega}_k^T \boldsymbol{\delta}_k(\boldsymbol{x}) + \log(\pi_k) \end{aligned}$$

with

$$\boldsymbol{\omega}_k = \boldsymbol{W}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_0)$$

and

$$\boldsymbol{\delta}_k(\boldsymbol{x}) = \boldsymbol{W}(\boldsymbol{x} - \frac{\boldsymbol{\mu}_k + \boldsymbol{\mu}_0}{2})$$

The expected log Bayes factor is

$$\mathrm{E}_{F_k}(\log B_{k0}) = KL(F_k || F_0) = \frac{1}{2}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_k - \boldsymbol{\mu}_0) = \frac{1}{2}\Omega_k^2$$

With scale factor $c_k = (\frac{1}{n_k} - \frac{1}{n})^{-1/2} = \sqrt{n \frac{\pi_k}{1-\pi_k}}$ (for $\mathrm{SD}(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_0)$, with the minus sign before $\frac{1}{n}$ due to correlation between $\hat{\boldsymbol{\mu}}_k$ and pooled mean $\hat{\boldsymbol{\mu}}_0$) we get as correlation-adjusted t-score for comparing mean of group k with the pooled mean

$$\boldsymbol{\tau}_k^{adj} = c_k \boldsymbol{\omega}_k^{ZCA-cor} .$$

For the two class case ($K = 2$) we get with $\boldsymbol{\mu}_0 = \pi_1 \boldsymbol{\mu}_1 + \pi_2 \boldsymbol{\mu}_2$ for the mean difference $(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) = \pi_2(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$ and with $c_1 = \sqrt{n \frac{\pi_1}{\pi_2}}$ this yields

$$\boldsymbol{\tau}_1^{adj} = \sqrt{n \pi_1 \pi_2} \boldsymbol{P}^{-1/2} \boldsymbol{V}^{-1/2} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2),$$

i.e. the exact same score as in the two-class setting.

4.5.3 Choosing a threshold

From the above it is clear that in LDA and DDA the natural score to rank features with regard to importance in the predictor is the (squared) t-score (no correlation) or the squared correlation-adjusted t-score.

In order to determine a suitable threshold one can use any standard technique, such as multiple testing multiple testing or FDR thresholding.

In Computer Lab 4 we will perform feature selection on an example data set using both the t-score and the correlation-adjusted t-score.

This will also show that using feature selection it is often possible to construct compact models with fewer predictors that still generalise and predict well.

For large and high-dimensional models feature selection can also be viewed as a form of regularisation and also dimension reduction. Specifically, if there are many variables/ features that do no contribute to the prediction they can still deterioriate the overall predictive accuracy (sometimes dramatically) so these "noise variables" need to be filtered out in order to be able to construct good models and classifiers.

4.6 Estimating prediction error

4.6.1 Quantifying prediction error

For any prediction model we are interested in the predictive performance. We quantify the performance by comparing the prediction $\hat{y}$ with the true output y (assumed to be known).

For continuous response often the squared loss is used:

$$\text{err}(\hat{y}, y) = (\hat{y} - y)^2$$

For binary outcomes one often employs the 0/1 loss:

$$\text{err}(\hat{y}, y) = \begin{cases} 1, & \text{if } \hat{y} = y \\ 0, & \text{otherwise} \end{cases}$$

but we can of course use any other quantity derived from the confusion matrix (containing TP, TN, FP, FN).

The mean prediction error is then the expectation

$$PE = \text{E}(\text{err}(\hat{y}, y))$$

and thus the empirical mean prediction error is

$$\widehat{PE} = \frac{1}{m} \sum_{i=1}^{m} \text{err}(\hat{y}_i, y_i)$$

where m is the sample size of the **test data** (different from the **training data** used to construct the model!!).

Alternatively and more generally, we can also quantify prediction error in the framework of so-called **proper scoring rules**, where the whole probabilistic forecast is taken into account (e.g. the individual probabilities for each class, rather than just the selected most probable class). A commonly used scoring rule is the negative log-probability ("surprise"), and the expected surprise is the cross-entropy (cf. Statistical Methods module). So this leads back to entropy and likelihood.

Once we have an estimate of the prediction error of a model we can use this error to choose among the models (including models with different numbers of features).

4.6.2 Estimation of prediction error without test data

Unfortunately, quite oten we do not have any test data available to evaluate a classifier.

In this case we need to rely on a simple algorithmic procedure called **cross-validation**.

Idea:

- split the samples in the training data into a number (say K) parts ("folds")
- use each of the K folds as test data and the other $K-1$ folds as training data
- average over the resulting K estimates of prediction error

Note that in each case one part of the data is reserved for testing and not used for training.

We choose K such that the folds are not too small (to allow estimation of prediction error) but also not too large (to make sure that we actually train a good classifier from the remaining data). A typical value for K is 5 or 10.

In Computer labs 4 and 5 we employ cross-validation to estimate prediction accuracy and model selection.

Relevant reading for the technical details: **Section 5.1 Cross-Validation** in James et al. (2013) *An introduction to statistical learning with applications in R*. Springer.

Chapter 5

Multivariate dependencies

5.1 Measuring the association between two sets of random variables

5.1.1 Rozeboom vector correlation

In linear regression model the squared multiple correlation (also known as coefficient of determination) between y and $\boldsymbol{x}$

$$\text{Cor}(y, \boldsymbol{x})^2 = \boldsymbol{P}_{y\boldsymbol{x}} \boldsymbol{P}_{\boldsymbol{x}}^{-1} \boldsymbol{P}_{y\boldsymbol{x}}$$

is a standard measure to describe the strength of association between the predictors $\boldsymbol{x}$ and the response y. If there is only a single predictor the squared multiple correlation, or cofficient of determination, reduces to the squared Pearson correlation $\text{Cor}(x, y)^2$.

Now, if we consider two random vectors $\boldsymbol{x} = (x_1, \ldots, x_p)^T$ and $\boldsymbol{y} = (y_1, \ldots, y_q)^T$, what is a relevant measure to describe the association between $\boldsymbol{x}$ and $\boldsymbol{y}$ that generalises both simple correlation and multiple correlation?

An answer in the form of squared *vector correlation* was given by Rozeboom (1965) defined as follows:

$$\text{Cor}(\boldsymbol{x}, \boldsymbol{y})^2 \rho_{\boldsymbol{x}\boldsymbol{y}}^2 = 1 - \prod_{i=1}^{m} (1 - \lambda_i^2)$$

where $\lambda_1, \ldots, \lambda_m$ are the canonical correlations, i.e. the singular values of $\boldsymbol{K} = \boldsymbol{P}_{\boldsymbol{x}}^{-1/2} \boldsymbol{P}_{\boldsymbol{x}\boldsymbol{y}} \boldsymbol{P}_{\boldsymbol{y}}^{-1/2}$ (cf. Chapter 2).

The Rozeboom vector correlation is the complement of Hotelling's (1936) *vector alienation coefficient* given by

$$a(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{m} (1 - \lambda_i^2)$$

so $\text{Cor}(\boldsymbol{x}, \boldsymbol{y})^2 = 1 - a(\boldsymbol{x}, \boldsymbol{y})$.

Equivalent ways to write the squared vector correlation are

$$\begin{aligned}\mathrm{Cor}(\boldsymbol{x},\boldsymbol{y})^2 &= 1 - \frac{\det \boldsymbol{P}_{x,x}}{\det \boldsymbol{P}_x \ \det \boldsymbol{P}_y}\\ &= 1 - \det\left(\boldsymbol{I}_q - \boldsymbol{P}_y^{-1}\boldsymbol{P}_{yx}\boldsymbol{P}_x^{-1}\boldsymbol{P}_{xy}\right)\\ &= 1 - \det\left(\boldsymbol{I}_p - \boldsymbol{P}_x^{-1}\boldsymbol{P}_{xy}\boldsymbol{P}_y^{-1}\boldsymbol{P}_{yx}\right) \ .\end{aligned}$$

It is easy to see that for $p = 1$ or $q = 1$ the squared vector correlation reduces to the squared multiple correlation.

Using the Weinstein-Aronszajn determinant identity $\det(\boldsymbol{I} + \boldsymbol{AB}) = \det(\boldsymbol{I} + \boldsymbol{BA})$ we can see that

$$\begin{aligned}\det\left(\boldsymbol{I}_q - \boldsymbol{P}_y^{-1}\boldsymbol{P}_{yx}\boldsymbol{P}_x^{-1}\boldsymbol{P}_{xy}\right) &= \det\left(\boldsymbol{I}_q - \boldsymbol{P}_y^{-1/2}\boldsymbol{P}_{yx}\boldsymbol{P}_x^{-1}\boldsymbol{P}_{xy}\boldsymbol{P}_y^{-1/2}\right)\\ &= \det\left(\boldsymbol{I}_q - \boldsymbol{K}^T\boldsymbol{K}\right)\end{aligned}$$

Thus, the squared vector correlation between $\boldsymbol{x}$ and $\boldsymbol{y}$ written in terms of the matrix $\boldsymbol{K}$ is

$$\begin{aligned}\rho_{xy}^2 &= 1 - \det\left(\boldsymbol{I}_q - \boldsymbol{K}^T\boldsymbol{K}\right)\\ &= 1 - \det\left(\boldsymbol{I}_p - \boldsymbol{K}\boldsymbol{K}^T\right)\end{aligned}$$

which brings us back to the first definition as the complement of the vector alienation coefficient.

5.1.2 Other common approaches

A common appproach to measure association between is the RV coefficient defined as

$$RV(\boldsymbol{X},\boldsymbol{Y}) = \frac{\mathrm{Tr}(\boldsymbol{\Sigma}_{\boldsymbol{XY}}\boldsymbol{\Sigma}_{\boldsymbol{YX}})}{\mathrm{Tr}(\boldsymbol{\Sigma}_{\boldsymbol{X}}^2)\mathrm{Tr}(\boldsymbol{\Sigma}_{\boldsymbol{Y}}^2)}$$

While for $q = p = 1$ the RV coefficient becomes the pairwise squared correlation. However, the RV coefficient does not reduce to the multiple correlation coefficient for $q = 1$ and $p > 1$.

Another way to measure multivariate association is mutual information (MI) which not only covers linear but also non-linear association. See the next Chapter for details. MI applied to multivariate normal distribution is linked to the above Rozeboom vector correlation.

5.2 Graphical models

5.2.1 Purpose

Graphical models combine features from

- graph theory

- probability
- statistical inference

The literature on graphical models is huge, we focus here only on two commonly used models:

- DAGs (directed acyclic graphs), all edges are directed, no directed loops (i.e. no cycles, hence "acyclic")
- GGM (Gaussian graphical models), all edges are undirected

Graphical models provide probabilistic models for trees and for networks, with random variables represented by nodes in the graphs, and branches representing conditional dependencies. In this regard they generalise both the tree-based clustering approaches as well as the probabilistic non-hierarchical methods (GMMs).

However, the class of graphical models goes much beyond simple unsupervised learning models. It also includes regression, classification, time series models etc. See e.g. the reference book by Murphy (2012).

5.2.2 Basic notions from graph theory

- Mathematically, a graph $G = (V, E)$ consists of a a set of vertices or nodes $V = \{v_1, v_2, \ldots\}$ and a set of branches or edges $E = \{e_1, e_2, \ldots\}$.
- Edges can be undirected or directed.
- Graphs containing only directed edges are directed graphs, and likewise graphs containing only undirected edges are called undirected graphs. Graphs containing both directed and undirected edges are called partially directed graphs.
- A path is a sequence of of vertices such that from each of its vertices there is an edge to the next vertex in the sequence.
- A graph is connected when there is a path between every pair of vertices.
- A cycle is a path in a graph that connects a node with itself.
- A connected graph with no cycles is a called a tree.
- The degree of a node is the number of edges it connects with. If edges are all directed the degree of a node is the sum of the in-degree and out-degree, which counts the incoming and outgoing edges, respectively.
- External nodes are nodes with degree 1. In a tree-structed graph these are also called leafs.

Some notions are only relevant for graphs with directed edges:

- In a directed graph the parent node(s) of vertex v is the set of nodes $\text{pa}(v)$ directly connected to v via edges directed from the parent node(s) towards v.
- Conversely, v is called a child node of $\text{pa}(v)$. Note that a parent node can have several child nodes, so v may not be the only child of $\text{pa}(v)$.
- In a directed tree graph, each node has only a single parent, except for one particular node that has no parent at all (this node is called the root node).
- A DAG, or directed acyclic graph, is a directed graph with no directed cycles. A (directed) tree is a special version of a DAG.

5.2.3 Probabilistic graphical models

A graphical model uses a graph to describe the relationship between random variables $x_1, \dots, x_d$. The variables are assumed to have a joint distribution with density/mass function $\text{Pr}(x_1, x_2, \dots, x_d)$. Each random variable is placed in a node of the graph.

The structure of the graph and the type of the edges connecting (or not connecting) any pair of nodes/variables is used to describe the conditional dependencies, and to simplify the joint distribution.

Thus, a graphical model is in essence a visualisation of the joint distribution using structural information from the graph helping to understand the mutual relationship among the variables.

5.2.4 Directed graphical models

In a **directed graphical model** the graph structure is assumed to be a DAG (or a directed tree, which is also a DAG).

Then the joint probability distribution can be factorised into a *product of conditional probabilities* as follows:

$$\text{Pr}(x_1, x_2, \dots, x_d) = \prod_i \text{Pr}(x_i | \text{pa}(x_i))$$

Thus, the overall joint probability distribution is specified by local conditional distributions and the graph structure, with the directions of the edges providing the information about parent-child node relationships.

Probabilistic DAGs are also known as "Bayesian networks".

Idea: by trying out all possible trees/graphs and fitting them to the data using maximum likelihood (or Bayesian inference) we hope to be able identify the graph structure of the data-generating process.

Challenges

1) in the tree/network the internal nodes are usually not known, and thus have to be treated as *latent* variables.

Answer: To impute the states at these nodes we may use the EM algorithm as in GMMs (which in fact can be viewed as graphical models, too!).

2) If we treat the internal nodes as unknowns we need to marginalise over the internal nodes, i.e. we need to sum / integrate over all possible set of states of the internal nodes!

Answer: This can be handled very effectively using the **Viterbi algorithm** which is essentially an application of the generalised distributive law. In particular for tree graphs this means that the summations occurs locally at each nodes and propagates recursively accross the tree.

3) In order to infer the tree or network structure the space of all trees or networks need to be explored. This is not possible in an exhaustive fashion unless the number of variables in the tree is very small.

Answer: Solution: use heuristic approaches for tree and network search!

4) Furthermore, there exist so-called "equivalence classes" of graphical models, i.e. sets of graphical models that share the same joint probability distribution. Thus, all graphical models within the same equivalence class cannot be distinguished from observational data, even with infinite sample size!

Answer: this is a fundamental mathematical problem of identifiability so there is now way around this issue. However, on the positive side, this also implies that the search through all graphical models can be restricted to finding the so-called "essential graph" (e.g. https://projecteuclid.org/euclid.aos/1031833662)

Conclusion: using directed graphical models for structure discovery is very time consuming and computationally demanding for anything but small toy data sets.

This also explains why heuristic and non-model based approaches (such as hierarchical clustering) are so popular even though full statistical modelling is in principle possible.

5.2.5 Undirected graphical models

Another class of graphical models are models that contain only undirected edges. These **undirected graphical models** are used to represent the pairwise conditional (in)dependencies among the variables in the graph, and the resulting model is therefore also called **conditional independence graph**.

If x_i and x_j two selected random variables/nodes, and the set $\{x_k\}$ represents all other variables/nodes with $k \neq i$ and $k \neq j$. We say that variables x_i and x_j are conditionally independent given all the other variables $\{x_k\}$

$$x_i \perp\!\!\!\perp x_j | \{x_k\}$$

if the joint probability density of x_i, x_j and x_k factorises as

$$\Pr(x_1, x_2, \ldots, x_d) = \Pr(x_i | \{x_k\}) \Pr(x_j | \{x_k\}) \Pr(\{x_k\}) \,.$$

or equivalently

$$\Pr(x_i, x_j | \{x_k\}) = \Pr(x_i | \{x_k\}) \Pr(x_j | \{x_k\}) \,.$$

In a corresponding conditional independence graph, there is no edge between x_i and x_j, as in such a graph *missing edges correspond to conditional independencies* between the respective non-connected nodes.

5.2.5.1 Gaussian graphical model

Assuming that $x_1, \ldots, x_d$ are jointly normal distributed, i.e. $\boldsymbol{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, it turns out that it is straightforward to identify the pairwise conditional independencies. From $\boldsymbol{\Sigma}$ we first obtain the precision matrix

$$\boldsymbol{\Omega} = (\omega_{ij}) = \boldsymbol{\Sigma}^{-1} \,.$$

Crucially, it can be shown that $\omega_{ij} = 0$ implies $x_i \perp\!\!\!\perp x_j | \{x_k\}$! Hence, from the precision matrix $\boldsymbol{\Omega}$ we can directly read off all the pairwise conditional independencies among the variables $x_1, x_2, \ldots, x_d$!

Often, the covariance matrix $\boldsymbol{\Sigma}$ is dense (few zeros) but the corresponding precision matrix $\boldsymbol{\Omega}$ is sparse (many zeros).

The conditional independence graph computed for normally distributed variables is called a **Gaussian graphical model**, or **GGM**. A further alternative name is **covariance selection model**.

5.2.5.2 Related quantity: partial correlation

From the precision matrix $\boldsymbol{\Omega}$ we can also compute the matrix of pairwise full conditional *partial correlations*:

$$\rho_{ij|\text{rest}} = -\frac{\omega_{ij}}{\sqrt{\omega_{ii}\omega_{jj}}}$$

which is essentially the standardised precision matrix (similar to correlation but with an extra minus sign!)

The partial correlations lie in the range between -1 and +1, $\rho_{ij|\text{rest}} \in [-1, 1]$, just like standard correlations.

If $\boldsymbol{x}$ is multivariate normal then $\rho_{ij|\text{rest}} = 0$ indicates conditional independence between x_i and x_j.

Regression interpretation: partial correlation is the correlation that remains between the two variables if the effect of the other variables is "regressed away". In other words, the partial correlation is exactly equivalent to the correlation between the residuals that remain after regressing x_i on the variables $\{x_k\}$ and x_j on $\{x_k\}$.

5.2.6 Algorithm for learning GGMs

From the above we can devise a simple algorithm to to learn Gaussian graphical model (GGM) from data:

1. Estimate covariance $\hat{\boldsymbol{\Sigma}}$ (in such a way that it is invertible!)
2. Compute corresponding partial correlations
3. If $\hat{\rho}_{ij|\text{rest}} \approx 0$ then there is (approx). conditional independence between x_i and x_j.
   In practise this is done by statistical testing for vanishing partial correlations. If there are many edges we also need adjustment for simultaneous multiple testing since all edges are tested in parallel.

5.2.7 Example: exam score data (Mardia et al 1979:)

Correlations (rounded to 2 digits):

```
##              mechanics vectors algebra analysis statistics
## mechanics         1.00    0.55    0.55     0.41       0.39
## vectors          0.55    1.00    0.61     0.49       0.44
## algebra          0.55    0.61    1.00     0.71       0.66
## analysis         0.41    0.49    0.71     1.00       0.61
## statistics       0.39    0.44    0.66     0.61       1.00
```

Partial correlations (rounded to 2 digits):

```
##            mechanics vectors algebra analysis statistics
## mechanics       1.00    0.33    0.23     0.00       0.02
## vectors         0.33    1.00    0.28     0.08       0.02
## algebra         0.23    0.28    1.00     0.43       0.36
## analysis        0.00    0.08    0.43     1.00       0.25
## statistics      0.02    0.02    0.36     0.25       1.00
```

Note that that there are no zero correlations but there are **four partial correlations close to 0**, indicating **conditional independencies** between:

- analysis and mechanics,
- statistics and mechanics,
- analysis and vectors, and
- statistics and vectors.

Thus, of 10 possible edges four are missing, and thus the conditional independence graph looks as follows:

```
Mechanics      Analysis
   |     \    /    |
   |    Algebra    |
   |     /   \     |
 Vectors      Statistics
```

Chapter 6

Nonlinear and nonparametric models

In the last part of the module we discuss methods that go beyond the traditional linear methods in multivariate statistics.

Relevant textbooks:

The lectures for much of this part of the module follow selected chapters from the following three text books:

- James et al. (2013) *An introduction to statistical learning with applications in R*. Springer.
- Hastie et al. (2009) *The elements of statistical learning: data mining, inference, and prediction*. Springer.
- Rogers and Girolami (2017) *A first course in machine learning (2nd edition)*. CRC Press.

Please study the relevant section and chapters as indicated below in each subsection!

6.1 Limits of linear models and correlation

Linear models are very effective tools. However, it is important to recognise their limits especially when modelling complex nonlinear relationships.

6.1.1 Correlation only measures linear dependence

A very simple demonstration of this is given by the following example. Assume x is a normal distributed random variable with $x \sim N(0, 1)$. From x we construct a second random variable $y = x^2$ — thus y fully depends on x with no added extra noise. What is the correlation between x and y?

Let's ansers this question by doing a small computer simulation:

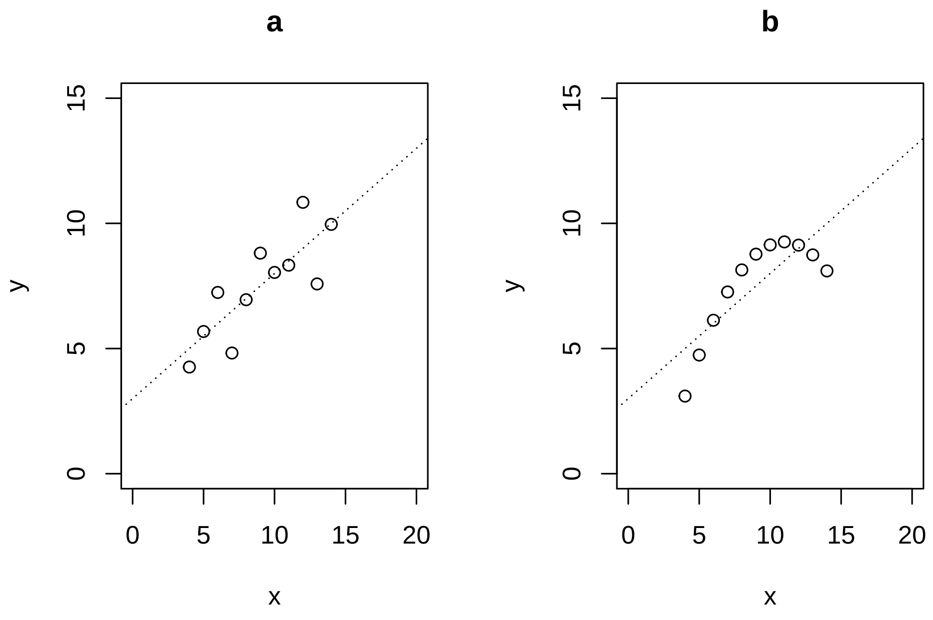
```r
x=rnorm(10000)
y = x^2
cor(x,y)
```

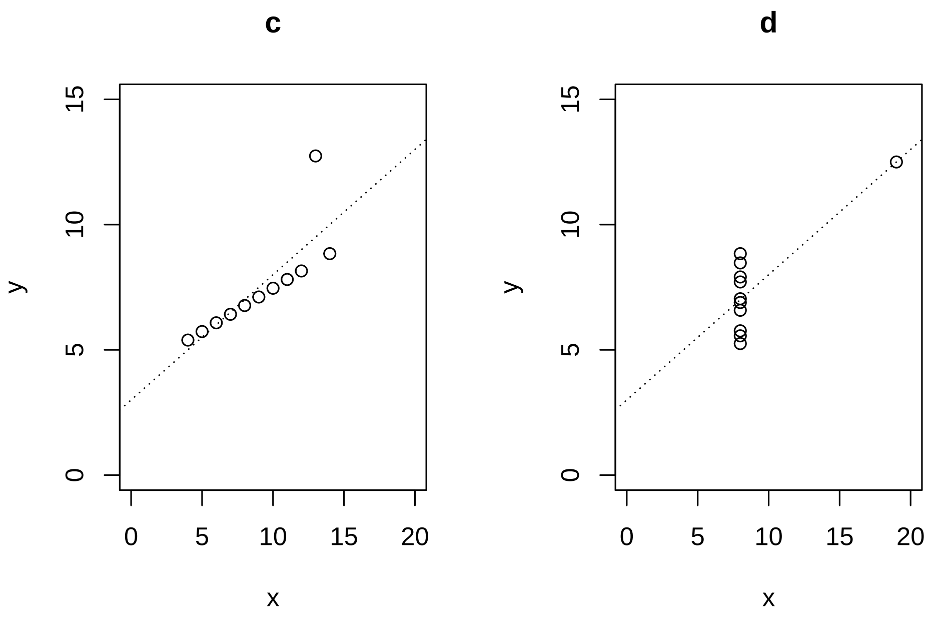
```
## [1] -0.01440713
```

Thus, correlation is (almost) zero even though *x* and y$ are full dependent! This is because correlation only measures linear dependence!

6.1.2 Anscombe data sets

Using correlation, and more generally linear models, blindly can thus hide complexities of the analysed data. A furthre classic example for this is demonstrated by the "Anscombe quartet" of data sets (F. J. Anscombe. 1973. Graphs in statistical analysis. The American Statistician 27:17-21, http://dx.doi.org/10.1080/00031305.1973.10478966):





As evident from the scatter plots the relationship between the two variables x and y is very different in the four cases! Intriguingly, all four data sets share exactly the same linear characteristics and summary statistics:

- Means $m_x = 9$ and $m_y = 7.5$
- Variances $s_x^2 = 11$ and $s_y^2 = 4.13$
- Correlation $r = 0.8162$
- Linear model fit with intercept $a = 3.0$ and slope $b = 0.5$

Thus, in actual data analysis it is always a **good idea to inspect the data visually** to get a first impression whether using a linear model makes sense.

In the above only data "a" follows a linear model. Data "b" represents a quadratic relationship. Data "c" is linear but with an outlier that disturbs the linear relationship. Finally data "d" also contains an outlier but also represent a case where y is (apart from the outlier) is not dependent on x.

In the Worksheet 10 a more recent version of the Anscomber quartet will be analysed in the form of the "datasauRus" dozen - 13 highly nonlinear datasets that all share the same linear characteristics.

6.2 Mutual information as generalised correlation

6.2.1 Definition of mutual information

Recall from the year 2 module "Statistical Methods" (MATH20802) the definition of Kullback-Leibler divergence, or relative entropy, between two distributions:

$$I^{KL}(F||G) := \mathrm{E}_F \log\left(\frac{f(x)}{g(x)}\right)$$

Here F plays the role of the true distribution and G is an approximating distribution, with f and g being the corresponding densities.

The *Mutual information* (MI) between two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as the KL divergence between the corresponding joint distribution and the product distribution:

$$\mathrm{MI}(\boldsymbol{x},\boldsymbol{y}) = KL(F_{\boldsymbol{x},\boldsymbol{y}}||F_{\boldsymbol{x}}F_{\boldsymbol{y}}) = \mathrm{E}_{F_{\boldsymbol{x},\boldsymbol{y}}} \log\left(\frac{f(\boldsymbol{x},\boldsymbol{y})}{f(\boldsymbol{x})\,f(\boldsymbol{y})}\right).$$

Thus, MI measures how well the joint distribution can be approximated by the product distribution (which would be the appropriate joint distribution if $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent). Since MI is an application of KL divergence is shares all its properties. In particular, $\mathrm{MI}(\boldsymbol{x},\boldsymbol{y}) = 0$ implies that the joint distribution and product distributions are the same. Hence the two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent if the mutual information vanishes.

6.2.2 Mutual information between two normal variables

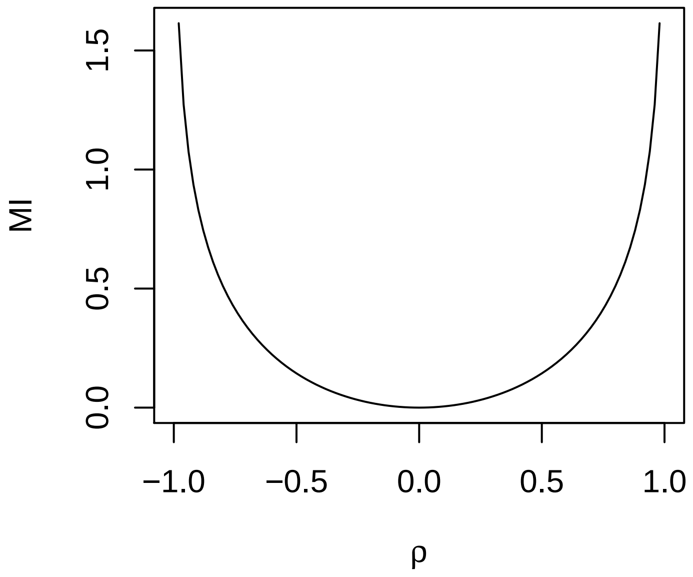
The KL divergence between two multivariate normal distributions F_0 and F is

$$I^{KL}(F_0||F) = \frac{1}{2}\left\{(\boldsymbol{\mu}-\boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}_0) + \mathrm{Tr}\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_0\right) - \log\det\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_0\right) - d\right\}$$

This allows compute the mutual information $\mathrm{MI}_{\mathrm{norm}(x,y)}$ between two univariate random variables x and y that are are correlated and assumed to be jointly bivariate normal. Let $\boldsymbol{z} = (x,y)^T$. The joint bivariate normal distribution is characterised by the mean $\mathrm{E}(\boldsymbol{z}) = \boldsymbol{\mu} = (\mu_x,\mu_y)^T$ and the covariance matrix $\boldsymbol{\Sigma} = \begin{pmatrix}\sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2\end{pmatrix}$ where $\mathrm{Cor}(x,y) = \rho$. If x and y are independent then $\rho = 0$ and $\boldsymbol{\Sigma}_{\mathrm{indep}} = \begin{pmatrix}\sigma_x^2 & 0 \\ 0 & \sigma_y^2\end{pmatrix}$. The mutual information between x and y is therefore

$$\begin{aligned}
\mathrm{MI}_{\mathrm{norm}}(x,y) &= I^{KL}(N(\boldsymbol{\mu},\boldsymbol{\Sigma})||N(\boldsymbol{\mu},\boldsymbol{\Sigma}_{\mathrm{indep}})) \\
&= \frac{1}{2}\left\{\mathrm{Tr}\left(\boldsymbol{\Sigma}_{\mathrm{indep}}^{-1}\boldsymbol{\Sigma}\right) - \log\det\left(\boldsymbol{\Sigma}_{\mathrm{indep}}^{-1}\boldsymbol{\Sigma}\right) - 2\right\} \\
&= \frac{1}{2}\left\{\mathrm{Tr}\left(\boldsymbol{P}_{x,y}\right) - \log\det\left(\boldsymbol{P}_{x,y}\right) - 2\right\} \\
&= -\frac{1}{2}\log\det\left(\boldsymbol{P}_{x,y}\right) \\
&= -\frac{1}{2}\log\det\begin{pmatrix}1 & \rho \\ \rho & 1\end{pmatrix} \\
&= -\frac{1}{2}\log(1-\rho^2) \\
&\approx \frac{\rho^2}{2}
\end{aligned}$$

Thus $\mathrm{MI}_{\mathrm{norm}}(x,y)$ is is a one-to-one function of the squared correlation ρ^2 between x and y:



For small values of correlation $2\text{MI}_{\text{norm}}(x, y) \approx \rho^2$.

6.2.3 Mutual information between two normally distributed random vectors

The mutual information $\text{MI}_{\text{norm}}(\boldsymbol{x}, \boldsymbol{y})$ between two multivariate normal random vector $\boldsymbol{x}$ and $\boldsymbol{y}$ can be computed in a similar fashion as in the bivariate case.

Let $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y})^T$ with dimension $d = p + q$. The joint multivariate normal distribution is characterised by the mean $\text{E}(\boldsymbol{z}) = \boldsymbol{\mu} = (\boldsymbol{\mu}_x^T, \boldsymbol{\mu}_y^T)^T$ and the covariance matrix $\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{x,x} & \boldsymbol{\Sigma}_{x,y} \\ \boldsymbol{\Sigma}_{x,y}^T & \boldsymbol{\Sigma}_{y,y} \end{pmatrix}$. If $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent then $\boldsymbol{\Sigma}_{x,y} = 0$ and $\boldsymbol{\Sigma}_{\text{indep}} = \begin{pmatrix} \boldsymbol{\Sigma}_{x,x} & 0 \\ 0 & \boldsymbol{\Sigma}_{y,y} \end{pmatrix}$. Mutual information between $\boldsymbol{x}$ and $\boldsymbol{y}$ is then

$$\begin{aligned}
\text{MI}_{\text{norm}}(\boldsymbol{x}, \boldsymbol{y}) &= I^{KL}(N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) || N(\boldsymbol{\mu}, \boldsymbol{\Sigma}_{\text{indep}})) \\
&= \frac{1}{2}\left\{ \text{Tr}\left(\boldsymbol{\Sigma}_{\text{indep}}^{-1} \boldsymbol{\Sigma} \right) - \log\det\left(\boldsymbol{\Sigma}_{\text{indep}}^{-1} \boldsymbol{\Sigma} \right) - d \right\} \\
&= -\frac{1}{2} \log\det\left(\boldsymbol{P}_{\text{indep}}^{-1} \boldsymbol{P} \right) \\
&= -\frac{1}{2} \log\left(\det\begin{pmatrix} \boldsymbol{P}_x & \boldsymbol{P}_{xy} \\ \boldsymbol{P}_{xy}^T & \boldsymbol{P}_y \end{pmatrix} / (\det \boldsymbol{P}_x \ \det \boldsymbol{P}_y) \right) \\
&= -\frac{1}{2} \log\det\left(\boldsymbol{I}_q - \boldsymbol{P}_y^{-1} \boldsymbol{P}_{yx} \boldsymbol{P}_x^{-1} \boldsymbol{P}_{xy} \right) \\
&= -\frac{1}{2} \log\det\left(\boldsymbol{I}_p - \boldsymbol{P}_x^{-1} \boldsymbol{P}_{xy} \boldsymbol{P}_y^{-1} \boldsymbol{P}_{y,x} \right)
\end{aligned}$$

The equality of

$$\det\left(\boldsymbol{I}_q - \boldsymbol{P}_y^{-1} \boldsymbol{P}_{yx} \boldsymbol{P}_x^{-1} \boldsymbol{P}_{x,y} \right) = \det\left(\boldsymbol{I}_p - \boldsymbol{P}_x^{-1} \boldsymbol{P}_{xy} \boldsymbol{P}_y^{-1} \boldsymbol{P}_{yx} \right)$$

follows from the Weinstein-Aronszajn determinant identity $\det(\boldsymbol{I}+\boldsymbol{AB}) = \det(\boldsymbol{I}+\boldsymbol{BA})$.

By comparison with the squared Rozeboom vector correlation coefficient $\rho^2_{\boldsymbol{x},\boldsymbol{y}}$ (cf. Chapter 5) we see that

$$\mathrm{MI}_{\text{norm}}(\boldsymbol{x},\boldsymbol{y}) = -\frac{1}{2}\log(1-\rho^2_{\boldsymbol{x},\boldsymbol{y}}) \approx \frac{1}{2}\rho^2_{\boldsymbol{x},\boldsymbol{y}}$$

Thus, in the multivariate case $\mathrm{MI}_{\text{norm}}(\boldsymbol{x},\boldsymbol{y})$ has exactly the same functional relationship with the vector correlation $\rho^2_{\boldsymbol{x},\boldsymbol{y}}$ as the MI for two univariate variables and squared Pearson correlation.

From the various expressions for the vector correlations we can also get further identities for the corresponding mutual information between $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\begin{aligned}\mathrm{MI}_{\text{norm}}(\boldsymbol{x},\boldsymbol{y}) &= -\frac{1}{2}\log\det\left(\boldsymbol{I}_q - \boldsymbol{K}^T\boldsymbol{K}\right)\\ &= -\frac{1}{2}\log\det\left(\boldsymbol{I}_p - \boldsymbol{K}\boldsymbol{K}^T\right)\\ &= -\frac{1}{2}\sum_{i=1}^{m}\log(1-\lambda_i^2)\end{aligned}$$

Note that the last line shows that $\mathrm{MI}_{\text{norm}}(\boldsymbol{x},\boldsymbol{y})$ is the sum of the MIs resulting from the individual canonical correlations (again with the same functional form as in the univeriate Pearson correlation case).

6.2.4 Using MI for variable selection

In principle, MI can be computed for any distribution and model and thus applies to both normal and non-normal models, and to both linear and nonlinear relationships.

In very general way we may denote by $F_{\boldsymbol{y}|\boldsymbol{x}}$ we denote a predictive model for $\boldsymbol{y}$ conditioned on $\boldsymbol{x}$ and $F_{\boldsymbol{y}}$ is the marginal distribution of $\boldsymbol{y}$ without predictors. Note that the predictive model can assume any form (incl. nonlinear). Typically $F_{\boldsymbol{y}|\boldsymbol{x}}$ is a complex model and $F_{\boldsymbol{y}}$ a simple model (no predictors).

Then mutual information between $\boldsymbol{x}$ and $\boldsymbol{y}$ can be also understood as expectated KL divergence between the conditional and marginal distributions:

$$\mathrm{E}_{F_{\boldsymbol{x}}} KL(F_{\boldsymbol{y}|\boldsymbol{x}}||F_{\boldsymbol{y}}) = \mathrm{MI}(\boldsymbol{x},\boldsymbol{y})$$

This can be shown as follows. The KL-divergence between $F_{\boldsymbol{y}|\boldsymbol{x}}$ and $F_{\boldsymbol{y}}$ is given by

$$I^{KL}(F_{\boldsymbol{y}|\boldsymbol{x}},F_{\boldsymbol{y}}) = \mathrm{E}_{F_{\boldsymbol{y}|\boldsymbol{x}}}\log\left(\frac{f(\boldsymbol{y}|\boldsymbol{x})}{f(\boldsymbol{y})}\right),$$

which is a random variable since it depends on $\boldsymbol{x}$. Taking the expectation with regard to $F_{\boldsymbol{x}}$ (the distribution of $\boldsymbol{x}$) we get

$$\mathrm{E}_{F_{\boldsymbol{x}}} I^{KL}(F_{\boldsymbol{y}|\boldsymbol{x}},F_{\boldsymbol{y}}) = \mathrm{E}_{F_{\boldsymbol{x}}}\mathrm{E}_{F_{\boldsymbol{y}|\boldsymbol{x}}}\log\left(\frac{f(\boldsymbol{y}|\boldsymbol{x})f(\boldsymbol{x})}{f(\boldsymbol{y})f(\boldsymbol{x})}\right) = \mathrm{E}_{F_{\boldsymbol{x},\boldsymbol{y}}}\log\left(\frac{f(\boldsymbol{x},\boldsymbol{y})}{f(\boldsymbol{y})f(\boldsymbol{x})}\right) = \mathrm{MI}(\boldsymbol{x},\boldsymbol{y}).$$

Because of this link of MI with conditioning the MI between response and predictor variables is often used for variable and feature selection in general models.

6.2.5 Other measures of general dependence

Besides mutual information there are others measures of general dependence between multivariate random variables.

The two most important ones that have been propsed in the recent literatur are i) distance correlation and ii) the maximal information coefficient (MIC and MIC_e).

6.3 Nonlinear regression models

Traditional linear (and generalised linear) models can been extended to nonlinear settings.

Relevant reading:

Please read: James et al. (2013) **Chapter 7 "Moving Beyond Linearity"**

Specifically:

- Section 7.1 Polynomial Regression
- Section 7.4 Regression Splines

6.3.1 Scatterplot smoothing

- lowess / loess algorithm

Locally weighted scatterplot smoothing (intended for exploratory analysis, not for probabilistic modelling).

6.3.2 Polynomial regression model

Advantage: - possible to use standard OLS tools to fit model and to do inference (relabeling trick for univariate models)

Disadvantage: - multivariate version complicated and intractable - high-oder polynomials are very erratic - prone to overfitting if degree/order is too high

6.3.3 Piecewise polyomial regression

- simple linear piece-wise model
- basis function approach
- regression splines
- natural splines

See Worksheet 10 for practical application in R!

6.4 Random forests

Another widely used approach for prediction in nonlinear settings is the method of random forests.

Relevant reading:

Please read: James et al. (2013) **Chapter 8 "Tree-Based Methods"**

Specifically:

- Section 8.1 The Basics of Decision Trees
- Section 8.2.1 Bagging
- Section 8.2.2 Random Forests

6.4.1 Stochastic vs. algorithmic models

Two cultures in statistical modelling: stochastic vs. algorithmic models

Classic discussion paper by Leo Breiman (2001): Statistical modeling: the two cultures. Statistical Science. Vol 16, pages 199-231. https://projecteuclid.org/euclid.ss/1009213726

6.4.2 Random forests

Invented by Breimann in 1996.

Basic idea:

- A single decision tree is unreliable and unstable (weak predictor/classifier).
- Use boostrap to generate multiple decision trees (="forest")
- Average over predictions from all tree (="bagging", bootstrap aggregation)

The averaging procedure has the effect of variance stabilisation. Intringuingly, averaging across all decision trees dramatically improves the overall prediction accuracy!

The Random Forests approach is an example of an **ensemble method** (since it is based on using an "ensemble" of trees).

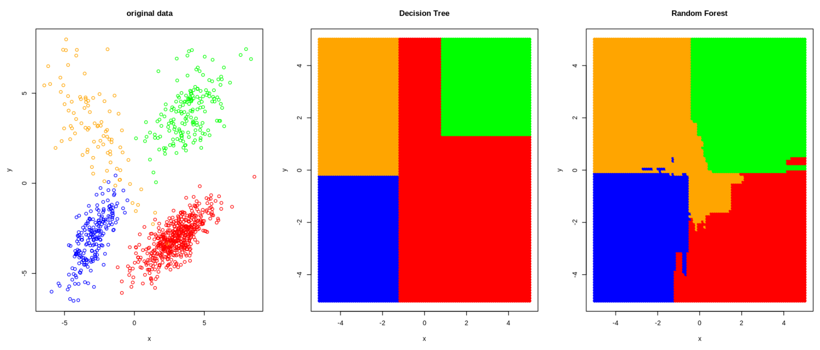
Variations: boosting, XGBoost (https://xgboost.ai/)

Random forests will be applied in Computer Lab 5.

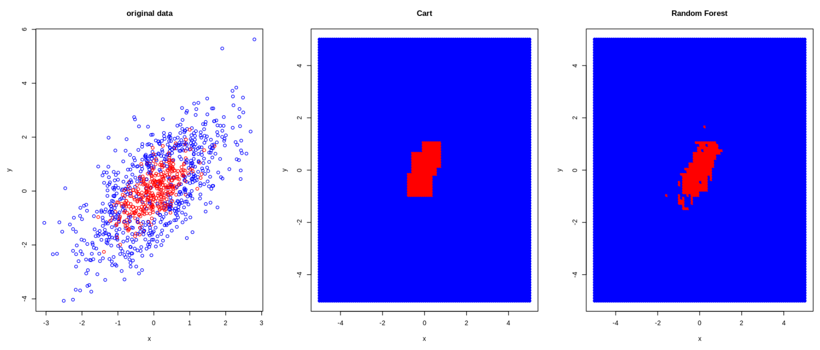
They are computationally expensive but typically perform very well!

6.4.3 Comparison of decision boundaries: decision tree vs. random forest

Non-nested case:



Nested case:



Compare also with the decision boundaries for LDA and QDA (previous chapter).

See Worksheet 11 for practical application of random forests in R!

6.5 Gaussian processes

Gaussian processes offer another nonparametric approach to model nonlinear dependencies. They provide a probabilistic model for the unknown nonlinear function.

Relevant reading

Please read: Rogers and Girolami (2017) **Chapter 8: Gaussian processes.**

6.5.1 Main concepts

- Gaussian processes (GPs) belong the the family of **Bayesian nonparametric models**
- Idea:
  - start with prior over a function (!),
  - then condition on observed data to get posterior distribution (again over all functions)
  - use an infinitely dimensional multivariate normal distribution as prior

6.5.2 Technical background:

GPs make use of the fact that marginal and conditional distributions of a multivariate normal are also multivariate normal.

Multivariate normal distribution:

$$\boldsymbol{z} \sim N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Assume:

$$\boldsymbol{z} = \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix}$$

with

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}$$

and

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^T & \boldsymbol{\Sigma}_{22} \end{pmatrix}$$

with corresponding dimensions d_1 and d_2 and $d_1 + d_2 = d$.

Marginal distributions:

Any subset of $\boldsymbol{z}$ is also multivariate normal distributed:

$$\boldsymbol{z}_i \sim N_{d_i}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_{ii})$$

Conditional multivariate normal:

The conditional distribution is also multivariate normal:

$$\boldsymbol{z}_i | \boldsymbol{z}_j = \boldsymbol{z}_{i|j} \sim N_{d_i}(\boldsymbol{\mu}_{i|j}, \boldsymbol{\Sigma}_{i|j})$$

with

$$\boldsymbol{\mu}_{i|j} = \boldsymbol{\mu}_i + \boldsymbol{\Sigma}_{ij} \boldsymbol{\Sigma}_{jj}^{-1} (\boldsymbol{z}_j - \boldsymbol{\mu}_j)$$

and

$$\boldsymbol{\Sigma}_{i|j} = \boldsymbol{\Sigma}_{ii} - \boldsymbol{\Sigma}_{ij} \boldsymbol{\Sigma}_{jj}^{-1} \boldsymbol{\Sigma}_{ij}^T$$

$\boldsymbol{z}_{i|j}$ and $\boldsymbol{\mu}_{i|j}$ have dimension $d_i \times 1$ and $\boldsymbol{\Sigma}_{i|j}$ has dimension $d_i \times d_i$

6.5.3 Covariance functions and kernel

The GP prior is a infinitely dimensional multivariate normal with mean zero and the **covariance specified by a function**:

A widely used covariance function is

$$\text{Cov}(x, x') = \sigma^2 e^{-\frac{(x-x')^2}{2l^2}}$$

This is known as the **squared-exponential kernel** or **Radial-basis function (RBF) kernel**.

Note that $(x, x) = \sigma^2$ and the autocorrelation $\text{Cor}(x, x') = e^{-\frac{(x-x')^2}{2l^2}}$.

The parameter l is the length scale parameter and describes the wigglyness or smoothness of the resulting function. Small values of l mean more complex, more wiggly functions, and low autocorrelation.

There are many other kernel functions, including periodic, polynomial and linear kernels.

6.5.4 GP model

Nonlinear regression in the GP approach is conceptually very simple:

- start with GP prior over all x
- then condition on the observed $x_1, \ldots, x_n$
- the resulting conditional multivariate normal can used to predict the function values at any unobserved values of x
- automatically provides credible intervals for predictions.

GP regression also provides a direct link with Bayesian linear regression (using a linear kernel).

Drawbacks: computationally expensive (n^3 because of the matrix inversion)

6.6 Neural networks

Another highly important class of models for nonlinear prediction (and nonlinear function approximation) are neural networks.

Relevant reading:

Please read: Hastie et al. (2009) **Chapter 11 "Neural networks"**

6.6.1 History

Neural networks are actually relatively old models, going back to the 1950s!

Three phases of neural networks (NN)

- 1950/60: replicating functions of neurons in the brain (perceptron)
- 1980/90: neural networks as universal function approximators
- 2010—today: deep learning

The first phase was biologically inspired, the second phase focused on mathematical properties, and the current phase is pushed forward by advances in computer science and numerical optimisation:

- backpropagation algorithm
- auto-differentiation,
- stochastic gradient descent
- use of GPUs and TPUs,
- availability and devlopment of software packages by major internet companies:
    - TensorFlow/Keras (Google),

- MXNet (Amazon),
- PyTorch (Facebook),
- PaddlePaddle (Baidu) etc.

6.6.2 Neural networks

Neural networks are essentially stacked systems of linear regressions, mapping input nodes (random variables) to outputs (response nodes). Each internal layer corresponds to internal latent variables. Each layer is connected with the next layer by **non-linear activation functions**.

- feedforward single layer NN
- stacked nonlinear multiple regression with hidden variables
- optimise by empirical risk minimisation

It can be shown that NN can approximate any arbitrary non-linear function mapping input and output.

"Deep" neural networks have many layers, and their optimisation requires advanced techniques (see above).

Neural networks are very highly parameterised models and require typically a lot of data for training.

Some of the statistical aspects of NN are not well understood: in particular it is known that NN overfit the data but can still generalise well. On the other hand, it is also know that NN can also be "fooled", i.e. prediction can be unstable (adversarial examples).

Current statistical research on NN focuses on interpretability and on links with Bayesian inference and models (e.g. GPs). For example:

- https://link.springer.com/book/10.1007/978-3-030-28954-6
- https://arxiv.org/abs/1910.12478

6.6.3 Learning more about deep learning

A good place to to learn more about deep learning and about the actual implementations in computer code on various platforms is the book "Dive into deep learning" by Zhang et al. (2020) available online at https://d2l.ai/

Appendix A

Brief refresher on matrices

This is intended a very short recap of some essentials you need to know about matrices For more details please consult the lecture notes of earlier modules (e.g. linear algebra).

A.1 Matrix notation

We will frequently make use of matrix calculations. Matrix notation helps to make the equations simpler and enables to understand them better.

In matrix notion we distinguish between scalars, vectors, and matrices:

Scalar: x, X, lower or upper case, plain type.

Vector: $\boldsymbol{x}$, lower case, bold type. In handwriting one uses an arrow $\vec{x}$ to indicate a vector.

Matrix: $\boldsymbol{X}$, upper case, bold type. In handwriting you use an underscore $\underline{X}$ to indicate a matrix.

Note that a vector may be viewed as a matrix (with only one column or only one row). Likewise, a scalar may also be considered as as special case of a matrix.

Note on **random** matrices and vectors:

In the above notation you need to determine from the context whether a quantity represents a random variable, or whether it is a constant. You cannot read this this from the case (upper vs. lower case) as in the standard notation commonly used in univariate statistics.

A.2 Simple special matrices

$\boldsymbol{I}_d$ is the identity matrix. It is a square matrix of dimension $d \times d$ with the diagonal filled with 1 and off-diagonals filled with 0.

$$\boldsymbol{I}_d = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & 0 & & 1 \end{pmatrix}$$

$\mathbf{1}$ is a matrix that contains only 1s. Most often it is used in the form of a column vector with d rows:

$$\mathbf{1}_d = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

A.3 Simple matrix operations

Matrices behave much like ordinary numbers. For example, you can apply operations such as matrix addition $\boldsymbol{A} + \boldsymbol{B}$ and matrix multiplication $\boldsymbol{AB}$. In order to conduct these operations the matrices need to be compatible: for addition the matrices need to have the same dimension, and for multiplication the number of columns of $\boldsymbol{A}$ must match the number of rows of $\boldsymbol{B}$. Note that $\boldsymbol{AB} \neq \boldsymbol{BA}$, i.e. matrix multiplication is in general not commutative.

If $\boldsymbol{A}$ is a squared matrix and is nonsingular (i.e. it has no zero eigenvalues) then you can define an inverse $\boldsymbol{A}^{-1}$ such that $\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$.

The matrix transpose $t(\boldsymbol{A}) = \boldsymbol{A}^T$ interchanges rows and columns.

The trace of the matrix is the sum of the diagonal entries $\text{Tr}(\boldsymbol{A}) = \sum a_{ii}$.

The sum of the squares of all entries of a rectangular matrix $\boldsymbol{A} = (a_{ij})$ can be written using the trace as follows: $\sum_{i,j} a_{ij}^2 = \text{Tr}(\boldsymbol{A}^T\boldsymbol{A}) = \text{Tr}(\boldsymbol{A}\boldsymbol{A}^T)$

A.4 Orthogonal matrices

An orthogonal matrix $\boldsymbol{Q}$ has the property that $\boldsymbol{Q}^T = \boldsymbol{Q}^{-1}$. This implies that $\boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$. An orthogonal matrix $\boldsymbol{Q}$ can be interpreted geometrically as a rotation-reflection operator since multiplication of $\boldsymbol{Q}$ with a vector will result in a new vector of the same length but with a change in direction. The identity matrix $\boldsymbol{I}$ is the simplest example of an orthogonal matrix but in general there are infinitely many rotation-reflection matrices.

A.5 Eigenvalues and eigenvalue decomposition

A vector $\boldsymbol{u}_i$ is called an eigenvector of a square matrix $\boldsymbol{A}$ and λ_i the corresponding eigenvalue if $\boldsymbol{A}\boldsymbol{u}_i = \boldsymbol{u}_i \lambda_i$.

If $\boldsymbol{A}$ is a symmetric matrix (i.e. $\boldsymbol{A} = \boldsymbol{A}^T$) with real entries (as assumed for the rest of this section) then it can be shown that all eigenvalues are real, and that the corresponding eigenvectors are all orthogonal.

The eigenvalue decomposition of a symmetric real-valued $\boldsymbol{A}$ is given by

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$$

with

$$\boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_d \end{pmatrix}$$

being a diagonal matrix containing all eigenvalues λ_i of $\boldsymbol{A}$. $\boldsymbol{U}$ is an orthogonal matrix containing the corresponding eigensystem (i.e. all eigenvectors $\boldsymbol{u}_i$) in the columns of $\boldsymbol{U}$. The eigenvectors in $\boldsymbol{U}$ are orthonormal, i.e. they are orthogonal to each other and have length 1.

If $\boldsymbol{A}$ is multiplied with $\boldsymbol{U}$ we immediately see that $\boldsymbol{\Lambda}$ contains the eigenvalues since $\boldsymbol{A}\boldsymbol{U} = \boldsymbol{U}\boldsymbol{\Lambda}$.

Furthermore, it can be shown that the above eigenvalue decomposition of $\boldsymbol{A}$ is **unique apart from the signs of the eigenvectors** (i.e. individual column signs of $\boldsymbol{U}$ can be changed and the eigenvalue decomposition is still valid). In order to make the eigenvalue decomposition fully unique you need to impose further restrictions (e.g. require a positive diagonal of $\boldsymbol{U}$). Note that this may be particularly important in computer applications where the sign can vary depending on the specific implementation of the underlying numerical algorithms.

Eigenvalue decomposition is also known as spectral decomposition.

A.6 Singular value decomposition

A generalisation of the eigenvalue decomposition (which is for squared matrices) is the **singular value decomposition** (SVD).

Let $\boldsymbol{A}$ be a $n \times m$ matrix. The SVD decomposition of $\boldsymbol{A}$ is $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$.

$\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices, $\boldsymbol{D}$ contains the singular values.

As the eigenvalue decomposition SVD is unique apart from the signs of the column vectors in $\boldsymbol{U}$ and $\boldsymbol{V}$.

A.7 Positive (semi-)definiteness, rank, condition

If all $\lambda_i \geq 0$ then $\boldsymbol{A}$ is is called positive semi-definite.
If all eigenvalues are strictly positive $\lambda_i > 0$ then $\boldsymbol{A}$ is called positive definite.

If one or more of the eigenvalues equal zero then $\boldsymbol{A}$ is said to be singular.

The rank of $\boldsymbol{A}$ counts how many eigenvalues are non-zero.
$\boldsymbol{A}$ is full rank if all eigenvalues are non-zero.

The condition number of $\boldsymbol{A}$ is the ratio between the largest and smallest absolute eigenvalue.
If $\boldsymbol{A}$ is singular then the condition number is infinite.

A.8 Trace and determinant of a matrix and eigenvalues

The trace of the matrix $\boldsymbol{A}$ can be also obtained as the *sum* of its eigenvalues: $\text{Tr}(\boldsymbol{A}) = \sum \lambda_i$.

The determinant of $\boldsymbol{A}$ is the *product* of the eigenvalues, $\det(\boldsymbol{A}) = \prod \lambda_i$. Therefore, if $\boldsymbol{A}$ is singular then $\det(\boldsymbol{A}) = 0$.

Determinants have a multiplicative property, $\det(\boldsymbol{AB}) = \det(\boldsymbol{A})\det(\boldsymbol{B})$. Another important identity is $\det(\boldsymbol{I}_n + \boldsymbol{AB}) = \det(\boldsymbol{I}_m + \boldsymbol{BA})$ where $\boldsymbol{A}$ is a $n \times m$ and $\boldsymbol{B}$ is a $m \times n$ matrix. This is called the Weinstein-Aronszajn determinant identity (also credited to Sylvester).

A.9 Functions of matrices

Again, we focus on symmetric, real-valued squared matrices $\boldsymbol{A}$.

A matrix function $f(\boldsymbol{A})$, generalising from a simple function $f(a)$ can then be defined via the eigenvalue decomposition as

$$f(\boldsymbol{A}) = \boldsymbol{U} f(\boldsymbol{\Lambda}) \boldsymbol{U}^T = \boldsymbol{U} \begin{pmatrix} f(\lambda_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & f(\lambda_d) \end{pmatrix} \boldsymbol{U}^T$$

Therefore, in order to obtain the corresponding matrix function, the function is simply applied on the level of eigenvalues. By construction $f(\boldsymbol{A})$ will also be symmetric and and real-valued as long as the transformed eigenvalues $f(\lambda_i)$ are real.

Examples:

Example A.1. matrix power: $f(a) = a^p$ (with p real number)

Special cases of matrix power include :

- matrix inversion: $f(a) = a^{-1}$
- the matrix square root: $f(a) = a^{1/2}$
  (since there are multiple solutions to the square root there are also multiple matrix square roots. The principal matrix square root is given by using the positive square roots of all the eigenvalues. Thus the **principal matrix square root** of a positive semidefinite matrix is also positive semidefinite and it is unique).

Example A.2. matrix exponential: $f(a) = \exp(a)$

Example A.3. matrix logarithm: $f(a) = \log(a)$

For a positive definite matrix $\boldsymbol{A}$ computing the trace of the matrix logarithm of $\boldsymbol{A}$ is the same as taking the log of the determinant of $\boldsymbol{A}$:

$$\text{Tr}(\log(\boldsymbol{A})) = \log\det(\boldsymbol{A})$$

because $\sum \log(\lambda_i) = \log(\prod \lambda_i)$.

A.10 Matrix calculus

A.10.1 First order vector derivatives

A.10.1.1 Gradient

The **nabla operator** (also known as **del operator**) is the *row* vector

$$\nabla = (\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_d}) = \frac{\partial}{\partial \boldsymbol{x}}$$

containing the first order partial derivative operators.

The **gradient** of a scalar-valued function $f(\boldsymbol{x})$ with vector argument $\boldsymbol{x} = (x_1, \ldots, x_d)^T$ is also a *row* vector (with d columns) and can be expressed using the nabla operator

$$\nabla f(\boldsymbol{x}) = \left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \ldots, \frac{\partial f(\boldsymbol{x})}{\partial x_d}\right) = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \text{grad} f(\boldsymbol{x})\,.$$

Note the various notations for the gradient.

Example A.4. $f(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x} + b$. Then $\nabla f(\boldsymbol{x}) = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{a}^T$.

Example A.5. $f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{x}$. Then $\nabla f(\boldsymbol{x}) = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = 2\boldsymbol{x}^T$.

Example A.6. $f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$. Then $\nabla f(\boldsymbol{x}) = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{x}^T(\boldsymbol{A} + \boldsymbol{A}^T)$.

A.10.1.2 Jacobian matrix

For a vector-valued function

$$\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^T\,.$$

the computation of the gradient of each component yields the **Jacobian matrix** (with m rows and d columns)

$$\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{x}) = \begin{pmatrix} \nabla f_1(\boldsymbol{x}) \\ \vdots \\ \nabla f_m(\boldsymbol{x}) \end{pmatrix} = \left(\frac{\partial f_i(\boldsymbol{x})}{\partial x_j}\right) = \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} = D\boldsymbol{f}(\boldsymbol{x})$$

Again, note the various notations for the Jacobian matrix!

Example A.7. $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$. Then $\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{x}) = \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{A}$.

If $m = d$ then the Jacobian matrix is a square matrix and this allows to compute the **Jacobian determinant**

$$\det \boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{x}) = \det \left(\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right)$$

If $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ is an invertible function with $\boldsymbol{x} = \boldsymbol{f}^{-1}(\boldsymbol{y})$ then the Jacobian matrix is invertible and the inverted matrix is in fact the Jacobian of the inverse function!

This allows to compute the Jacobian determinant of the backtransformation as as the inverse of the Jacobian determinant the original function:

$$\det D\boldsymbol{f}^{-1}(\boldsymbol{y}) = (\det D\boldsymbol{f}(\boldsymbol{x}))^{-1}$$

or in alternative notation

$$\det D\boldsymbol{x}(\boldsymbol{y}) = \frac{1}{\det D\boldsymbol{y}(\boldsymbol{x})}$$

.

A.10.2 Second order vector derivatives

The matrix of all second order partial derivates of scalar-valued function with vector-valued argument is called the **Hessian matrix** and is computed by double application of the nabla operator:

$$\nabla^T \nabla f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1^2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_d \partial x_1} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_d^2} \end{pmatrix} = \left(\frac{\partial f(\boldsymbol{x})}{\partial x_i \partial x_j} \right) = \left(\frac{\partial}{\partial \boldsymbol{x}} \right)^T \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} .$$

By construction it is square and symmetric.

A.10.3 First order matrix derivatives

The derivative of a scalar-valued function $f(\boldsymbol{X})$ with regard to a matrix argument $\boldsymbol{X}$ can also be defined and results in a matrix with transposed dimensions compared to $\boldsymbol{X}$.

Two important specific examples are:

Example A.8. $\frac{\partial \mathrm{Tr}(\boldsymbol{A}\boldsymbol{X})}{\partial \boldsymbol{X}} = \boldsymbol{A}$

Example A.9. $\frac{\partial \log \det(\boldsymbol{X})}{\partial \boldsymbol{X}} = \frac{\partial \mathrm{Tr}(\log \boldsymbol{X})}{\partial \boldsymbol{X}} = \boldsymbol{X}^{-1}$

Appendix B

Further study

In this module we can only touch the surface of the field of multivariate statistics and machine learning. If you would like to study further I recommend the following books below as a starting point.

B.1 Recommended reading

For multivariate statistics and machine learning:

- Härdle and Simar (2015) *Applied multivariate statistical analysis. 4th edition*. Springer.
- Hastie et al. (2009) *The elements of statistical learning: data mining, inference, and prediction*. Springer.
- James et al. (2013) *An introduction to statistical learning with applications in R*. Springer.
- Marden (2015) *Multivariate Statistics: Old School*
- Rogers and Girolami (2017) *A first course in machine learning (2nd Edition)*. Chapman and Hall / CRC.

B.2 Advanced reading

Additional (advanced) reference books for probabilistic machine learning are:

- Murphy (2012) *Machine learning: a probabilistic perspective*. MIT Press.
- Bishop (2006) *Pattern recognition and machine learning*. Springer.

Bibliography

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer. https://www.microsoft.com/en-us/research/people/cmbishop/prml-book/.

Härdle, W. K. and Simar, L. (2015). *Applied Multivariate Statistical Analysis*. Springer, Berlin.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition. https://web.stanford.edu/~hastie/ElemStatLearn/.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. Springer. http://faculty.marshall.usc.edu/gareth-james/ISL/.

Marden, J. I. (2015). *Multivariate Statistics: Old School*. CreateSpace. http://stat.istics.net/Multivariate.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

Rogers, S. and Girolami, M. (2017). *A first course in machine learning*. Chapman and Hall / CRC, 2nd edition.

Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2020). *Dive into Deep Learning*. https://d2l.ai.