Lecture Notes for

Unsupervised Learning and Evolutionary Computation Using R

Winter Term 2021/2022

Dr. Jakob Bossek

Last changed: September 30, 2024

Abstract: This course focuses on multivariate statistical methods in the context of data mining. Within the lecture, you will learn how to pre-process, exploratively analyse and understand data, e.g., by means of checking for correlations or identifying outliers. The main topic of this course is unsupervised learning, i.e., understanding the inherent structure of unlabelled data. In addition to learning the underlying (theoretical) concepts of the presented methods, you will learn how to actually apply the taught approaches (using the statistical software R and powerful packages) to given data sets. Moreover, the last part of the lecture will deal with (multi-objective) optimisation. Here, you will learn about how to tackle challenging optimisation problems with heuristic algorithms that take inspiration from nature.

Information: This document was originally written for the lecture *Data Analytics I* held at the University of Münster in winter term 2021/2022. The manuscript may suffer from occasional typos, missing words, formulation that might benefit from refactoring as well as some inconsistent mathematical notation even though I payed much attention to avoid the latter in particular. Therefore, I advise you to read the script with care. Bug-reports, constructive feedback and solutions to the many exercises are highly welcome (jakob.bossek@uni-paderborn.de).

The script will likely be slightly expanded in the course of winter term 2024/2025.

Contents

Chapter 1

Introduction

Nowadays data is ubiquitous. It is collected everywhere: sensors measure physical measurements, (online retail) sellers log every single transaction and social media platforms, e.g., Facebook, well, log everything we do. Raw data however is quite useless; it is just heaps of numbers. The goal is to derive useful information from data (e.g., a retailer may be interested in certain *groups* of customers, say frequent buyers, people that bought some stuff after clicking on certain advertisement links etc.) in order to start new marketing campaigns more focused with the goal to optimise his or her business. To achieve this, in the past decades research came up with countless methods to aggregate, visualise, group and mine data. In Data Analytics we will learn basic concepts and methods with main focus on unsupervised learning.

1.1 Introductory Example

As an simple introductory example consider the data set given in Table 1.1. This set comprises of $N = 34$ different cars (*observations*; each row is one observation). For each observation $p = 10$ aspects of automobile design and performance are logged (*variables* or *features*; each column contains values of one variable), e.g. the weight (`wt`), the number of cylinders `cyl` and so on. So in total we have $N \times p$ numerical values. Obviously, we cannot make much sense of the data by glancing at heaps of numbers only. We are not capable of identifying any kind of relationships between variables or general patterns like membership to some specific class of automobile. Recall that this is just a toy example with respect to N and p; real-world data is usually larger by many orders of magnitude with both N and/or p. Now instead of the "raw" data in Table 1.1 take a look at Figure 1.1 where we visualise a part of the data. More precisely we draw scatter-plots of the variables `qsec` (quarter of miles per second) and

`wt` (weight). Apparently, this is much more "readable" for human beings. The left plot shows the data only with dashed lines indicating the variable-wise average value. Looking at the data we can identify 3(?) groups or *clusters*. A so-called clustering algorithm automatically calculated the assignment to three groups given in Figure 1.1 (centre). We could now go on and split the data by cluster assignment and further inspect differences and similarities. However, we could also decide to classify the lonely red observation as an outlier (e.g., measurement error). In consequence, the data would only consist of two groups. Finally, consider the right-most plot in Figure 1.1. Here, the black dots illustrate new data that arrived over time (*stream data*) after we already determined our clustering. Now we would need to adapt the clustering and our interpretations/conclusions: it seems that the red observations might actually form a cluster with the new augmented data set and the decision to classify it as an outlier would be obsolete.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 1.1: First 10 observations of the `mtcars` data set.

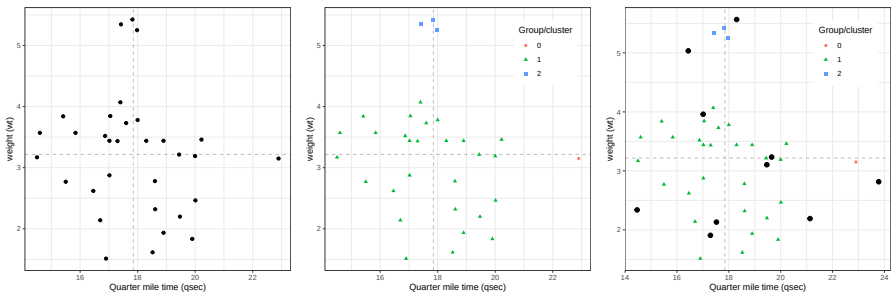


Figure 1.1: 2D-scatter-plots of two selected variables from the `mtcars` data set. Basic scatter-plot (left), scatter-plot coloured by cluster assignment calculated by some clustering algorithm (centre) and the problem of new data that arrived over time and requires the clustering to adapt (right).

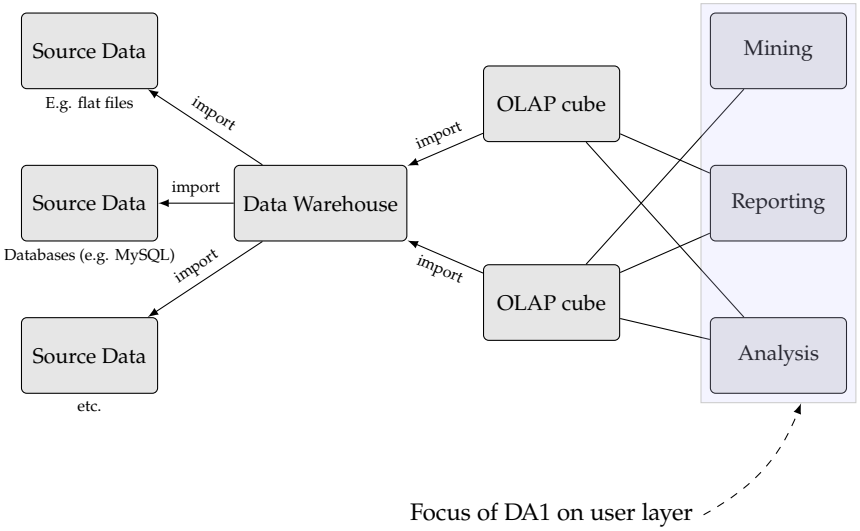


Figure 1.2: Schema of *Online Analytical Processing* (OLAP). I.e., the process of gathering, storing and analysing data.

1.2 DA1 Content

In Data Analytics 1 we will focus on the user layer in the context of the *Online Analytical Processing* (OLAP) pipeline (see Figure 1.2). OLAP is the process of gathering, storing and analysing data. Data Warehousing (DWH) concepts and the OLAP-pipeline as a whole will not be covered.[1]

More precisely (recall the introductory example) we will learn how to prepare/pre-process data for analysis. I.e., how to transform source data into a format that we can operate on nicely. Moreover, we will learn about outlier detection, dealing with missing data and different grouping/clustering algorithms to decompose data into several non-overlapping groups. Furthermore, we will learn about dimensionality reduction approaches that allow to map high-dimensional data into a low-dimensional space. In the last part we will learn about heuristic optimisation algorithms, so-called evolutionary or genetic algorithms, which mimic processes from Darwinian evolution theory to find good solutions to tackling (multi-objective) optimisation problems. This may seem as a completely different and unrelated topic. However, most unsupervised learning algorithms internally solve some kind of tackling optimisation problem as we will see later. Moreover, in supervised learning (see Data Analytics 2), optimisation is even more important when it comes to parameter tuning/configuration or feature selection.

[1] These topics are covered in *Management Information Systems and Data Warehousing* in the winter term.

Beside the theoretical concepts we put great emphasis on application. Therefore, we will use the statistical environment R (see Section 2) which gives us the necessary tools to apply the learned methods on real datasets.
The lecture notes will cover most of the theoretical concepts in depth while R will play a minor role here.

1.3 Exercises

The lecture notes provide you with a plethora of further exercises of different difficulty levels. Advanced, theoretically tackling exercises or exercises which require substantial implementations, are marked with a ★.

1.4 Attention

The lecture notes are new in this years DA1 edition and will constantly evolve. As you can imagine it takes a lot of time to write it and my time is unfortunately limited. Hence, it is very likely that the manuscript will contain typos, new content may be released with some delay etc. You can help to improve the lecture notes by reporting typos, giving constructive feedback and sending me solutions to the many additional exercises. Benefit for you: fame and honour and my gratitude.
I want to thank my former students Oliver Ludger Preuß (now PhD student at the MALEO group) and Till Siekmann who helped to improve the quality of the manuscript by reporting typos and errors.

Chapter 2

The R Programming Language

R [R C21] is an open-source *statistical programming environment* released under GNU GPL v2 license. The official R manual says: “"environment" is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.” This statement highlights that R was planned and developed as a language for statistical analysis and statistical learning exclusively. It is not a general purpose language as Python, Java or low-level languages like C or C++. However, it does not want to be either. It was developed with a goal in mind and it does a pretty good job at what it is designed for: making sense of data. The language itself already ships with many statistical visualisations and neat implementations of methods. However, the language and its popularity was boosted in the past decade by an rapidly increasing number of R packages, i.e., software libraries that extend R’s functionality with additional functions and/or datasets. In fact, in 2010 the number of released packages at the *Comprehensive R Archive Network* (CRAN) was $\approx 1\,000$. Today, CRAN lists over 11 000 packages already and the trend is increasing. Among the most popular R packages, to name only a few, are `ggplot2` [Wic09] for neat visualisations based on the grammar of graphics, the `tidyverse` [WG17] as a powerful collection of packages that make analysing data even more fun and `mlr` [Bis+16] or its modernised successor `mlr3` [Lan+19][1] with its many extension packages which offer a coherent, consistent and comprehensive interfaces to many supervised and unsupervised learning algorithms alongside many other extensible features that a machine learning pipeline requires.

[1] https://mlr3.mlr-org.com

In the TIOBE programming language popularity index[2] R ranks 14th (status: Oct. 18, 2021) with 1.20%. This is particularly interesting since the index does not measure the popularity of languages for data analysis, but the general popularity. Hence, ranking 14th is really impressive given Rs' niche application areas.

Beside these obvious selling points R is rather easy to learn[3] for beginners and even people with no experience in programming at all. This can be attributed to its design that is focused on data and the fact that few lines of code are enough to achieve some impressive results. In my honest opinion, another important driver for R's increasing spread is the availability of the user-friendly *Integrated Development Environment* (IDE) R-Studio[4] which was first released in 2011 and is further refined with regular updates by an active team of core developers and collaborators ever since.[5]

A thorough textual introduction is out of scope for this manuscript and makes little sense due to the zoo of good free online books. I believe that the presentation slides in the PANDA course are sufficient to get familiar with the language, `tidyverse` and `ggplot2`. For more in-depth introductions we refer the interested reader to a multitude of free online sources. [WG17] give an applied introduction to R for data science and should be a good starting point to most concepts used in the lecture. [Wic14] is a nice book on advanced R concepts. Readers interested in R package development should take a look at [Wic15]. A funny and deep-dive text on R is given by Patrick Burns [Bur11].[6]

2.1 Exercises

Exercise 1 (Data rotation)
Consider the data set $\mathcal{X} = \{(1,3),(4,5),(5,3),(3,1)\}$ of $N = 4$ observations of $p = 2$ real-values variables.

1. In R create a matrix X which contains these observations as rows.
2. Draw a basic scatter-plot of the data points.
3. Write a function `getRotationMatrix(theta)` which – as the name suggests – returns a (2×2) rotation matrix. The function expects the rotation angle θ as its sole parameter `theta`.

[2] https://www.tiobe.com/tiobe-index/
[3] Though it is hard to master and sometimes annoying due to some questionable design choices, the necessity for backwards compatibility and inconsistencies [**burns2012r**].
[4] https://www.rstudio.com/products/rstudio/download/
[5] Before the release of R-Studio one had to rely on quite bad R-editors. Even sophisticated and established IDEs had no or just unsatisfactory R support; this made working with R kind of painful.
[6] https://www.burns-stat.com/pages/Tutor/R_inferno.pdf

4. Use matrix multiplication to rotate the data points counterclockwise by 90 degrees and add the rotated points to your plot in a different colour.
5. Figure out (online) how to draw arrows from the original points to their rotated versions.
6. Now consider the `mtcars` dataset. Import the set and consider the subset of variables `wt` and `qsec` as your new data set X. Repeat all previous steps.

Exercise 2 (Sample variance)
Write a function `myvar(x, unbiased = TRUE)` which expects a numerical vector `x` and a logical parameter `unbiased` which defaults to the value `FALSE`. For `unbiased=FALSE` the function shall return the sample variance (see Appendix A)

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2.$$

With `unbiased = TRUE` the *unbiased sample variance* $\tilde{s}^2 = \left(\frac{n}{n-1}\right) s^2$ shall be returned.[7] Do not use R's implementations of `mean` and `var` in your function! Instead, try to come up with your own implementation(s), e.g., by using (different) loops.
Apply your function to all numeric variables of the `mtcars` data set in different ways, e.g., by iterating over the column numbers or names, by using a suitable function from the `*apply` function family etc.

Exercise 3 (Revert vector)
The function `rev(x)` takes a vector `x` and returns another vector where the elements of `x` are given in revere order. Write your own version of the function. Use a loop to iteratively swap two elements.

Exercise 4 (Coin tossing simulation)
The function `sample(...)` can be used to sample random numbers uniformly at random from a given set. Read the documentation of the function (`?sample`). Write a simulation study on tossing a single fair coin n times for different values of $n \to \infty$. Report the absolute/relative number of head/tail in some way (functions `barplot` or `table` could be useful).

Exercise 5 (Trimmed mean)

[7] Sometimes the unbiased sample variance is used. It has the property of being an *unbiased estimator* of the theoretical variance; this is interesting in the context of estimation theory, but not relevant in practice for large n, since $n/(n-1) \to 1$ for $n \to \infty$.

The arithmetic mean is likely the most popular and frequently used measure of mass. A related measure is the so-called *α-trimmed arithmetic mean* which for a series of data points $x_1, \ldots, x_n$ and $\alpha \in [0, 1)$ is defined as

$$\bar{x}_t(x_1, \ldots, x_n) = \frac{1}{n - 2\lfloor \alpha n \rfloor} \sum_{i=\lfloor \alpha n \rfloor}^{n-\lfloor \alpha n \rfloor} x_{(i)}.$$

Here, $\lfloor x \rfloor$ rounds its argument x to the nearest integer value lower than x and $x_{(i)}$ for $i = 1, \ldots, n$ is the ith order statistic (the ith order statistic is the value that comes at the ith position if the data is sorted in increasing order). Hence, the trimmed mean cuts of a fraction of α% lowest and highest observations and calculates the mean of the remaining ones. Implement a function `trimmedMean(x, alpha)` that implements the trimmed mean following the formula given above. Why should it be useful to use the trimmed mean instead of the standard arithmetic mean?

Exercise 6 (Re-scaling)
We will see later in the semester that for many multi-variate methods, i.e., methods that deal with data of at least two (numeric) variables $X_1, \ldots, X_p, p \geq 2$, different variable scales are problematic. Here, it is often useful to transform these variables to a common scale; oftentimes $[0, 1]$. For realizations $x_1, \ldots, x_N$ of a numeric variable X this *re-scaling* can be realized by calculating

$$\tilde{x}_i = \frac{x_i - \min_i x_i}{\max_i x_i - \min_i x_i}, i = 1, \ldots, n.$$

1. Implement a function `rescale(x)` that expects a numeric vector `x` as input and returns the re-scaled version.
2. Test your function on different input vectors.
3. Imagine the input vector contains at least one so-called NA-value (NA for not available, a special value type in R). Check your input on a vector with at least one NA value, e.g., `c(1, 5, NA, 10, 3)`. Modify your function by adding a logical parameter `na.rm` which defaults to `FALSE`. If set to `TRUE` NA values should be ignored during re-scaling.
4. Imagine now the input vector potentially contains either `Inf` or `Inf`. Again, check how your implementation behaves and come up with a possible solution.
   Hint: `is.infinite()` might be useful.

Literature

[Bur11] P. Burns. *The R Inferno*. Lulu Com, 2011.

[Wic14] H. Wickham. *Advanced R*. Chapman & Hall/CRC The R Series. Taylor & Francis, 2014.

[Wic15] Hadley Wickham. *R Packages*. 1st. O'Reilly Media, Inc., 2015.

[Bis+16] Bernd Bischl et al. "mlr: Machine Learning in R". In: *Journal of Machine Learning Research* 17.170 (2016), pp. 1–5.

[Lan+19] Michel Lang et al. "mlr3: A modern object-oriented machine learning framework in R". In: *Journal of Open Source Software* (2019). DOI: 10.21105/joss.01903.

[R C21] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2021.

Chapter 3

Outlier Detection

In the words of Hawkins [Haw80]: "An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism". *Outlier detection* (often *anomaly detection*) is the process of identifying observations that differ significantly from the majority of the data. We can go a step further and distinguish two classes of outliers: (i) (encoding) mistakes and (ii) extreme values. For instance imagine survey data being entered manually by some student assistant from sheets of paper. Here, it is perfectly possible that after long hours of boring (and badly paid) work some 19 year old survey participant gets a record of 190 years by accidentally inserting a trailing zero; clearly a mistake. On the other hand if we work with salary data extremely high values are rare, but definitely possible.

Application areas are manifold: data cleaning before the application of machine learning methods (this will be our focus). Anomaly detection plays a crucial role in fraud detection where the data consists of financial transactions (credit card usage patterns usually change if it gets stolen) or network intrusion detection (data may be log files where unusual traffic might indicate a possible attack). Quality control is another area where outlier/anomaly detection is used to detect, e.g., deviations from intended specifications for fabricated goods. Figure 3.1 shows a so-called *quality control chart* from the fabrication of screws. Here, the x-axis shows different points in time where a sample of produced screws was checked. The specified mean diameter of the screws is 15mm and natural deviations within [14.7mm; 15.3mm] are allowed (emphasised by the green area). At some point in time however, there is a sequence of measurements that are clearly and systematically below the specifications. The reason could be, e.g., wear and tear of some parts of the machine or an unintended incorrect calibration of the machine.

Outlier detection is a broad area. We will learn about the very basics here:

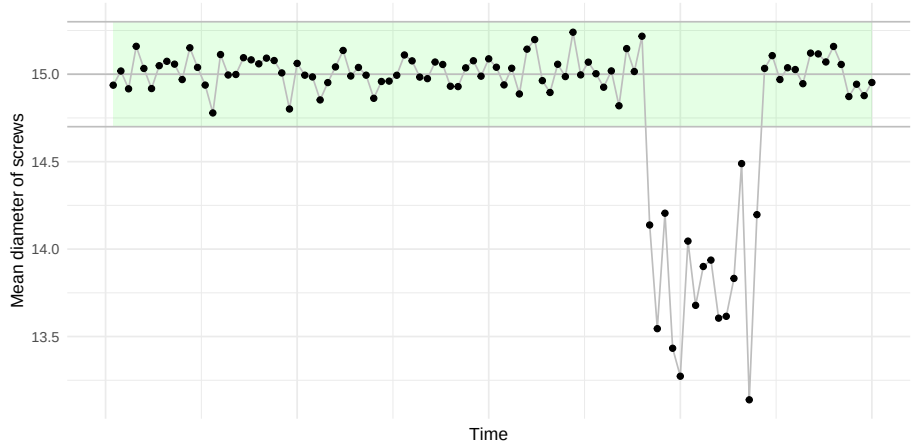


Figure 3.1: Quality control chart of sample measurements of the diameter of screws fabricated at a specific machine. The measurements within the green area are in line with the intended specifications (15 ± 0.3 [mm]). The sequence of points "out of bounds" is an indicator for some abnormal behaviour of the machine. Likely, some adjustments were necessary to fix the issues.

simple detection approaches of uni-variate (single variable) and multi-variate (multiple variables are involved) outliers.

3.1 Treatment of Outliers

How do we treat outliers? Well, it depends. In any case it is a bad idea to run one or several of the algorithms introduced in this section and automatically remove all observations which are labelled outliers. Instead, outliers must be carefully investigated and verified prior to some action. In most cases it will make sense to remove (or fix) mistakes. However, for extreme values it strongly depends on the goal of your analysis. Keeping outliers might have negative effects on linear regression tasks; if the goal is to model some trend in the data, outliers may strongly effect the outcome. Moreover, many statistical methods are parametric, i.e., they rely on some assumptions on the data distribution. Outliers usually have a negative effect on such methods wheres non-parametric methods are more robust. Another possibility is to perform studies twice: with and without outliers. The comparison of results may give valuable insights.

3.2 Detection by Visualisation

The first approach is straight-forward. So far we learned about exploratory data analysis and in particular about visualisation (with `ggplot2`). Visualisation is

a perfect tool to get a feeling for the data and to detect suspicious data points. In Figure 3.2 we see a histogram and a box-plot of the `carat` variable from the `diamonds` data set. The box-plot in particular is an excellent graphical tool to

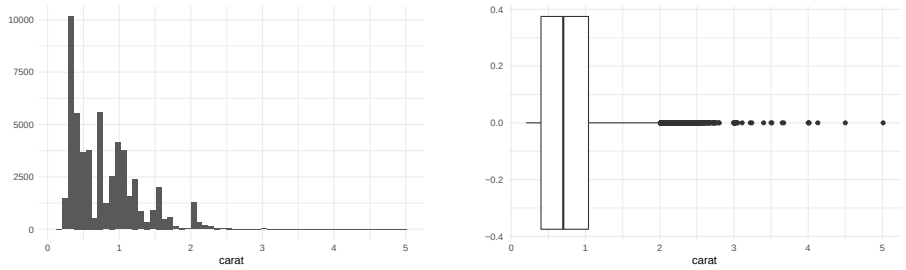


Figure 3.2: Histogram (left) and box-plot (right) of the `carat` variable from the `diamonds` data set.

grasp the big picture of uni-variate numeric data as it is composed of the median value $\tilde{x} = q_{0.5}$ (0.5-quantile), the quartiles $q_{0.25}$ and $q_{0.75}$ (0.25-quantile and 0.75-quantile) and it uses the so-called *interquartile range* (IQR) $IQR := q_{0.75} - q_{0.25}$ to draw the whiskers. We see that the vast majority of the observations are tightly concentrated around the median value $\tilde{x} = 0.7$ (thick black vertical line within the box). 50% of the observations are in $[q_{0.25}; q_{0.75}] = [0.4; 1.040]$ (inside the box) while 75% take values smaller than $q_{0.75} = 1.040$ (lower than the upper box border). The 0.95-quantile is $q_{0.95} = 1.7$. In R's box-plot calculations all points outside of the interval

$$[q_{0.25} - 1.5 \cdot IQR; q_{0.75} + 1.5 \cdot IQR] \tag{3.1}$$

are considered as potential outliers (black dots in box-plot). I.e., all points the deviate by more than 1.5 times the IQR from the quartiles. Ta-da! Here, we actually learned our first non-graphical indicator for outlier detection. In our example, a couple of observations show `carat` values greater than the upper limit $q_{0.75} + 1.5 \cdot IQR$ of the interval. However, keep in mind, that `carat` measures the weight of a diamond and heavy diamonds are rare. Thus, it is likely that the outliers in this example are extreme values and no encoding mistakes.

So far we aimed to find *uni-variate* outliers. To tackle the multi-variate case we could simply apply the uni-variate outlier detection to each variable separately and mark the union of labelled outliers as "multi-variate" outliers. However, this will likely produce in weird results. Figure 3.3 shows sample data where different situations occur: (a) uni-variate outlier can also be multi-variate outliers (red point), (b) points which are normal for the marginal distributions can be outliers for the joint distribution (blue point) and (c) points that are clearly

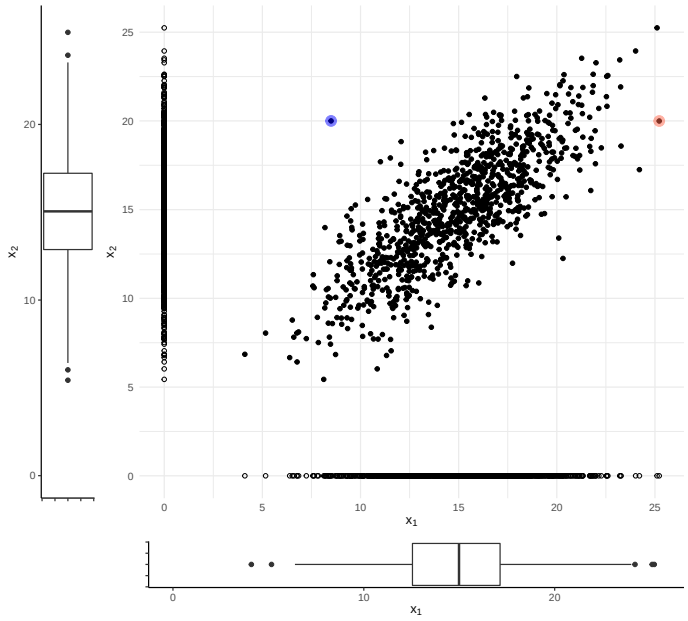


Figure 3.3: Exemplary bi-variate data. The points are also shown as projections on the respective axis to facilitate the visual detection of uni-variate outliers. The red point is a uni-variate outlier for x_1; it could also be labelled a multi-variate outlier. In contrast, the blue point is no uni-variate outlier at all (i.e., neither for x_1 nor for x_2), but clearly suspicious with respect to the joint distribution of x_1 and x_2. Most of the remaining uni-variate outliers fit well into the joint distribution.

uni-variate outliers, but fit very well into the joint distribution. Hence, we need more sophisticated approaches to detect multi-variate outliers.

3.3 A Parametric Method

For a normally distributed variable $X \sim \mathcal{N}(\mu, \sigma^2)$ the following approach is reasonable: normalise the data by centring the distribution and dividing by the standard deviation, i.e., build

$$Z = \frac{X - E(X)}{\sigma_X}.$$

Note that if $X \sim \mathcal{N}(\mu, \sigma^2)$, then $Z \sim \mathcal{N}(0, 1)$. I.e., the standardised variable follows a standard normal distribution with zero mean and unit variance. Normally distributed data have the characteristic property that the vast majority of the probability mass is concentrated around its expectation. I.e., most realisations are located quite close to the expected value while realisations far

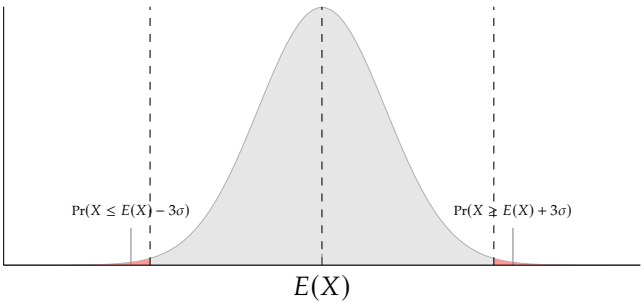


Figure 3.4: Density curve of a normally distributed variable X with expectation $E(X)$ and standard deviation σ. Recall that for continuous random variables the probability to take values in some interval $[a;b]$, $a < b$ equals the area under the density curve. Hence, the red shaded areas illustrate the probability to take values outside of $[E(X) - 3\sigma; E(X) + 3\sigma]$.

off the expectation have a certain non-zero likelihood, but are quite rare.[1] In particular, for $Z \sim \mathcal{N}(0,1)$ the likelihood for a realisation to be in the interval $[E(Z) - 2\sigma; E(Z) + 2\sigma]$ is 99.45% and for $[E(Z) - 3.5\sigma; E(Z) + 3.5\sigma]$ it is 99.95%. In consequence, the probability for a realisation outside of the latter interval is just 0.05%, i.e., very unlikely (see Figure 3.4).

Now let us finally come back to outlier detection. We can actually use this theoretical stuff to come up with a simple numerical method: if observations are close to the centre of mass they are likely no outliers. If they are far away though, we can consider them to be outliers. Hence, if our data shows realisations outside of a certain interval around the centre of mass and the data is in fact normally distributed, it gives us a hint for an unlikely event and the respective observations should be checked. Empirically, for N observations $x_1, \ldots, x_N$ we first calculate the average $\bar{x}$ and the standard deviation s and get

$$z_i = \frac{x_i - \bar{x}}{s}, i = 1, \ldots, N. \tag{3.2}$$

Next, we check which observations are located outside of $[\bar{x} - 3.5s; \bar{x} + 3.5s]$. I.e., we classify an observation x_i as an outlier if

$$|x - \bar{x}| \geq 3.5s.$$

The multi-variate case is trickier. For two variables we can again use visualisations (see, e.g., Figure 3.3). In this example there is a uni-variate outlier that does fit into the two-dimensional linear relationship perfectly. On the other hand, there is a bi-variate outlier that – from the uni-variate perspective – is

[1] Imagine the height (in cm) of 1 000 adult men. Such data usually fits a normal distribution very well. The mean value will be somewhere around 170cm and the majority of observations will be in the interval [150; 190]. A body size of 100cm or 210cm is possible, but a rather rare event.

no uni-variate outlier at all. This phenomenon of “hidden” outliers naturally transfers to more than two dimensions. The consequence is that data has to be checked very thoroughly and conclusions should be drawn with care.

We can generalise the "distance to the centre of mass" idea for the multi-variate case. Imagine we have p normally distributed variables $X_1, \ldots, X_p$ with respective mean values $\mu_i = E(X_i)$ and standard deviations σ_i, $i = 1, \ldots, p$. We could go ahead and calculate the sum of squared distances

$$(X - E(X))^T \cdot (X - E(X)) = \sum_{i=1}^{p} \left(\frac{X_i - \mu_i}{\sigma_i} \right)^2 \tag{3.3}$$

and classify an observation as an outlier if it is "large enough" to be considered unusual. Figure 3.5 shows a cloud of points with outliers marked accordingly. The left plots is so-called χ^2 plot. The name is due to the sum of squared distances in Eq. 3.3 being χ^2_p-distributed. The plots shows the theoretical χ^2_p-quantiles against the ordered distances. Distances higher than the 0.95-quantile of the χ^2_p-distribution are unlikely and thus marked as outliers (see coloured points in the right plot).[2]

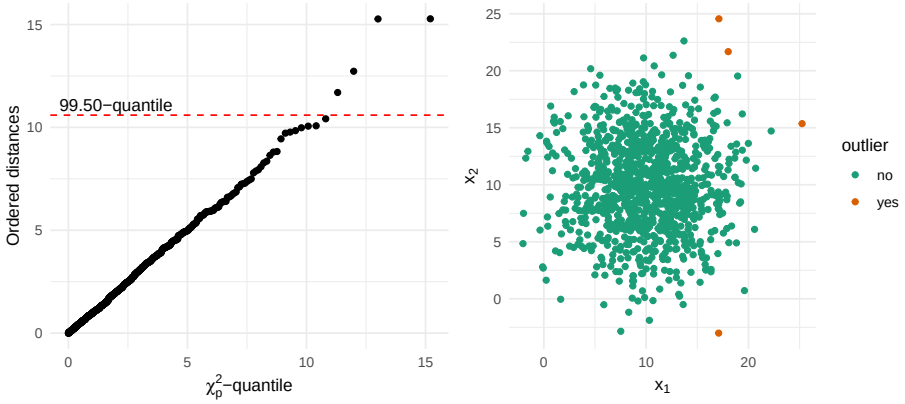


Figure 3.5: Parametric approach considering the sum of squared distances (Eq. 3.3) on spherical data.

However, there is problem. This approach assumes that the observations are scattered around the centre of mass in a spherical manner (see Figure 3.5). In the non-spherical case, e.g., if the data is ellipsoidal as it is often the case for correlated data the output is not as one might expect (see Figure 3.6). We need to take the correlations/directions into account. The solution is to calculate the

[2] Actually, for this method to work we do not need sorting. We can just filter all points whose distance is above the threshold. However, the χ^2-plot can be used for another purpose which is not discussed here.

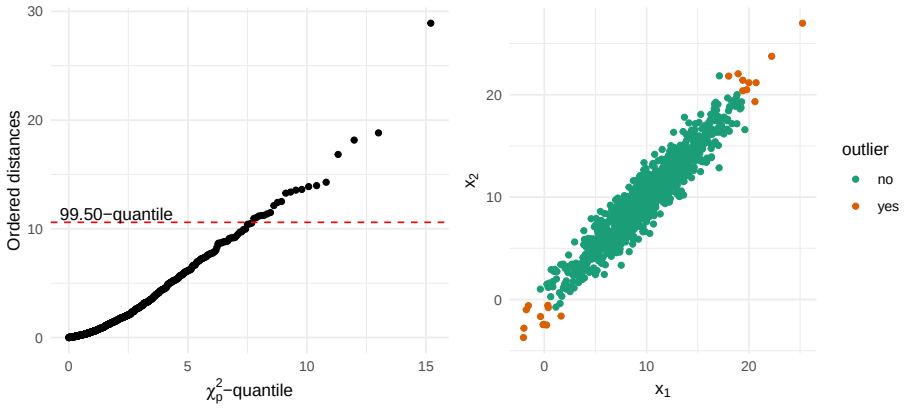


Figure 3.6: Parametric approach considering the sum of squared distances (Eq. 3.3) on non-spherical data.

so-called *generalised squared distance / Mahalanobis-distance*

$$d = (X - E(X))^T \cdot \Sigma^{-1} \cdot (X - E(X)) \tag{3.4}$$

Here, Σ^{-1} is the inverse of the covariance matrix Σ which normalises the distances with the correlations. Figure 3.7 shows the effect.

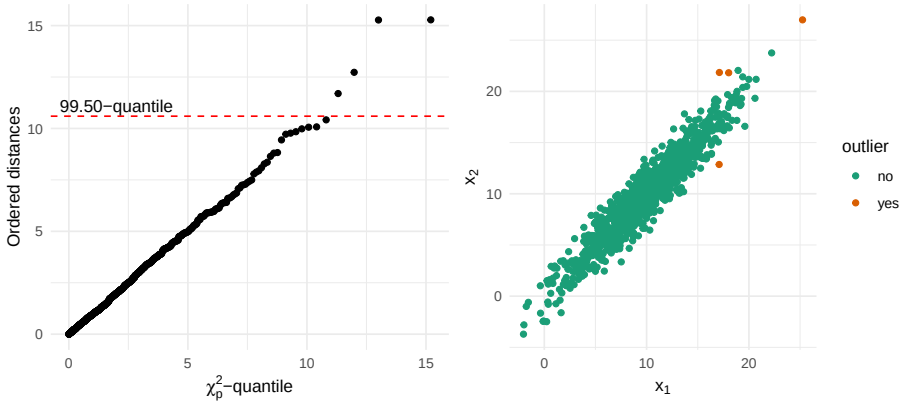


Figure 3.7: Parametric approach considering the Mahalanobis distance (Eq. 3.4) on non-spherical data.

Like all ML methods, this approach has weaknesses. The major drawback is its parametric nature. I.e., the method works reliably only if the data follows a multi-variate normal distribution. If non-linear relationships occur in the data the method will yield weird results as illustrated in Figure 3.8. Another weakness is that the arithmetic mean and the population variance (as a mean value of deviations) are non-robust measures, i.e., these measures are very

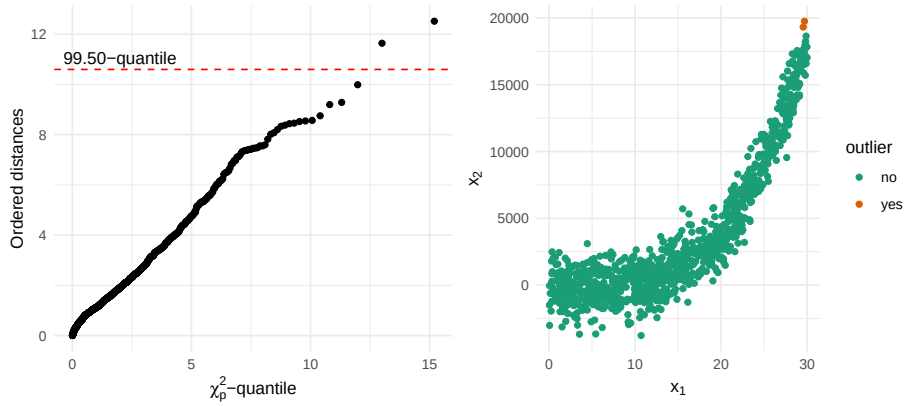


Figure 3.8: Parametric approach considering Mahalanobis distance (Eq. 3.4) on data with a non-linear trend. Two points are labelled outliers, but actually they fit into the big picture nicely.

sensitive to outliers. However, the generalised distance calculation obviously also involves the outliers. In consequence, the method is rather limited and should be used with care.

3.4 A Depth-Based Approach

We next discuss a method that was introduced by Ruts et al. [RR96] and relies on a model described by Tuckey [Tuk77]. It is independent of statistical distributions and of appealing simplicity. The core idea is that outliers are located at the border of the space spanned by the data. The algorithm works as follows: calculate the convex hull of the data. Initialise a layer counter $L = 1$. Assign all points on the convex hull to layer L. Next, remove all layer L points from the data, set $L = L + 1$, and start another iteration. Repeat the process until all points are assigned a layer. Given a parameter $k \geq 1$ all points assigned to layers at most k are labelled outliers. A helpful analogy is onion-peeling; onions are build up of different layers and the outermost layers correspond to the outliers.

Figure 3.9 shows an example of the approach on different data sets (spherical, elliptical).

A major drawback of the approach is the computational complexity of the underlying geometric problem. The Graham-scan algorithm calculates the convex-hull in time $O(N \log N)$ for $p = 2, 3$ [Gra72]. With h describing the number of points on the convex hull there also exist output-sensitive algo-

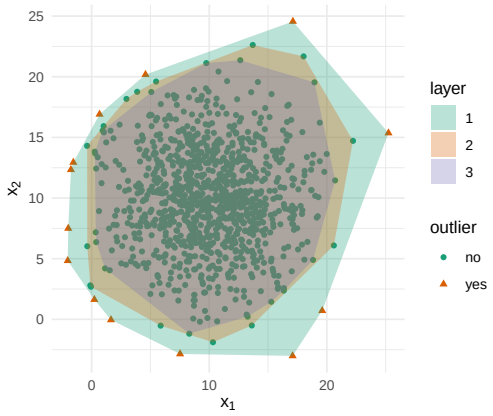


Figure 3.9: Visualisation of the onion-peeling character of the depth-based approach [RR96] on a spherical cluster. The first three convex layers are highlighted. In this example we classify the points on the outermost layer, the convex hull of the whole data set, as outliers.

rithms[3] with a runtime of $O(N \log h)$ which is of benefit if $h \ll N$. However, for $p \geq 4$ the best algorithms run in time $O(N^{p/2})$ which quickly becomes infeasible (for $p = 20$ this yields an upper bound of $O(N^{10})$ which is infeasible for usual data base sizes N).

3.5 Angle-based method

All methods discussed so far lose their advantages in high-dimensional spaces. Here, the so-called *curse of dimensionality* comes into play: the concept of neighbourhood/distances becomes increasingly meaningless and due to the sparsity of data all points can be labelled outliers. Here, more robust distance measures are necessary. One interesting approach is the *Angle-Based-Outlier-Degree* (ABOD) from 2008 [KSZ08]. As the name suggests the algorithm relies on angle calculations. For each point $x \in \mathcal{X}$ ABOD calculates the angle between x and two other points $y, z \in \mathcal{X} \setminus \{x\}$ with $y \neq z$. The rationale is that for outliers – which are by assumption in border regions of the data space – all other points are located in roughly similar directions. In contrast, "normal" points are rather centred and surrounded by many other normal points. For each point we can visualise all angles, the so-called *spectrum*. The diversity of these angles can be used as an indicator/score for outliers. This, however, is useful for very small toy examples only since there are $\Theta(N^2)$ different angles for each point. To automate the process we calculate the normalised variance

[3] An algorithm is called *output-sensitive* if its running time is not only a function of the input size, but also of the output size.

of the angles as the *ABOD-score* which is the variance of the normalised angles between x and other pairs of points. High values indicate a broad spectrum and can be interpreted as outliers while low values indicate normal points. To identify the top-k outliers we need to identify the k points with the highest ABOB-scores.
The naïve algorithm requires $O(N^3)$, i.e., cubic-time, since there are N points and we need $\binom{N-1}{2} = \frac{(N-1)(N-2)}{2} = O(N^2)$ pairs of other points to consider. [KSZ08] also present an approximation algorithm which solves the problem in time $O(N^2 + N \cdot k^2)$ where k is the number of samples taken ($k = N$ in the exact algorithm) which is more attractive, but performance deteriorates with increasing dimension.

3.6 Exercises

Exercise 1 (Standardisation)
Let X be a random variable with mean $E(X)$ and variance σ_X^2. Show that for the normalised variable $Z = \frac{X - E(X)}{\sigma_X}$ it holds that $E(Z) = 0$ and $\sigma_Y^2 = 1$.

Exercise 2 (Standardisation in R)
Apply the uni-variate standardisation method for outlier detection (see Eq. 3.2) to the `price` variable of the `diamonds` dataset. Subset all diamonds which are classified as outliers. There will be 346 classified outliers. Among these, 11 Fair-cut diamonds. Investigate what makes these diamonds special.

Exercise 3 (Generalised distances ★)
Let $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ for $i = 1, \ldots, p$ be independent random variables. Show that for the random vector $X = (X_1, \ldots, X_p)$ with covariance matrix $\Sigma \sim (p, p)$ the generalised distance

$$d = (X - E(X))^T \cdot \Sigma^{-1}(X - E(X))$$

follows a χ_p^2-distribution.

Exercise 4 (Depth-based approach in R)
In this exercise we aim to implement the depth-based outlier detection approach introduced in Section 3.4. To this end implement a respective function `outliers_depth_based(x, k = 1)` which expects a data frame `x` as mandatory input. The positive integer parameter `k` decides on the number of the outermost layers whose points shall be labelled outliers. The Function returns an integer vector with the row numbers of points classified as outliers.

Apply your function with different values of k on several two-variable combinations of the `mtcars` data set. Use `ggplot2` to highlight the detected outliers in scatter-plots. Describe your observations.
Hint: `grDevices::chull()` computes the subset of points which lie on the convex hull of a set.

Literature

[Gra72] R.L. Graham. "An efficient algorithm for determining the convex hull of a finite planar set". In: *Information Processing Letters* 1.4 (1972), pp. 132–133. DOI: https://doi.org/10.1016/0020-0190(72)90045-2.

[Haw80] D. M. Hawkins. *Identification of outliers*. Monographs on applied probability and statistics. Chapman and Hall, 1980.

[RR96] Ida Ruts and Peter J. Rousseeuw. "Computing Depth Contours of Bivariate Point Clouds". In: *Comput. Stat. Data Anal.* 23.1 (1996), 153–168. DOI: 10.1016/S0167-9473(96)00027-8.

[KSZ08] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. "Angle-Based Outlier Detection in High-Dimensional Data". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '08. Las Vegas, Nevada, USA: Association for Computing Machinery, 2008, 444–452. DOI: 10.1145/1401890.1401946.

Chapter 4

Cluster Analysis

As an (upcoming) data analyst we are often confronted with the following situation. We gathered heaps of data points in p-dimensional space in a certain application and our goal is to somehow make sense of all these data values. We already know how to use different visualization techniques and statistics to aggregate and summarize data and learn about the information contained therein. Another quite natural step is to ask the question whether the data can be sub-divided into several groups where observations within the groups are somehow similar to each other with respect to one or multiple features while observations which are in different groups are rather dissimilar. Imagine being employed by a successful online retail seller. The data you are faced with contains information on transactions of clients where each feature describes a certain product that the seller offers in his online shop. The goal now would be to identify certain patterns in the data, e.g., groups of customers that tend to buy few but expensive technical stuff or others that purchase many, but less costly items frequently. This information can be used for targeted and user-centred marketing. Another example stems from astronomy. Here, ultra-modern telescopes (e.g., Hubble by NASA) gather a plethora of data literally looking into the past scanning for stars, galaxies and other mind-blowing phenomena like black-holes or supernovae. Here, we have petabytes of data consisting of measurements of different spectra (infra-red etc.) of numerous interstellar objects. Here the goal could be to partition, e.g., stars, into groups with similar measurements (e.g. a certain size, temperature range etc.). I am sure that you can come up with countless other interesting application areas.

4.1 Foundations

In this chapter we are going to learn about different approaches to group observations, i.e., to build so-called *clusters/groups* of data. The process of searching for groups is called *clustering* and it essentially aims to find a k-partition of the data (cf. Appendix A).

Definition 4.1 (k-partition)**.** A *k-partition* of a set $\mathcal{X}$ is a decomposition of $\mathcal{X}$ into $k > 0$ *non-empty* subsets $C_1, \ldots, C_k$ such that the following holds:

1. $C_i \cap C_j = \emptyset$ for $1 \leq i \neq j \leq k$, i.e., the subsets are pairwise disjoint.
2. $\bigcup_{i=1}^{k} C_i = \mathcal{X}$, i.e., the union of all the subsets is $\mathcal{X}$ itself. We say that the partition *covers* $\mathcal{X}$.

Algorithm 1 finds a clustering for a fixed value of k and it does so in time which grows only linearly with N. However, you certainly agree that such a clustering will not be helpful at all. Obviously we are not interested in some arbitrary clustering. The challenge is to find a "good" clustering with respect to some quality criterion and – in this context – to find the "best" value of k.

Algorithm 1 Random clustering

Require: Data set $\mathcal{X}$, number of clusters k.
1: Assign each observation $x \in \mathcal{X}$ to one of the k clusters $C_1, \ldots, C_k$ uniformly at random.
2: **return** $C_1, \ldots, C_k$

In order to describe a "good clustering" we need to first define the notion of "similar/close" observations and "dissimilar/distant" observations.

Definition 4.2 (Distance function)**.** A function $d : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is called a *distance function* (or *dissimilarity function/measure*) if the following conditions hold for all $x, y, z \in \mathbb{R}^p$

(d1) $d(x, y) \geq 0$ and $d(x, y) = 0$ if and only if $x = y$,
(d2) $d(x, y) = d(y, x)$ (symmetry),
(d3) $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality).

Functions that respect Definition 4.2 are also called *metrics* in mathematics. Take a moment to think about it. The first property means that a distance cannot be negative and the distance of an observation to itself takes the minimum possible value. The second property requires symmetry. The third requirement is that the distance between two points is the shortest among all possible paths. The *Euclidean distance* (L_2-norm) is what people usually have in mind when

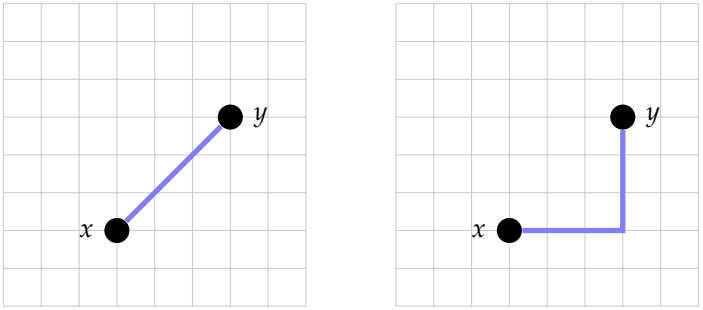


Figure 4.1: Euclidean distance (left) and Manhattan-block distance (right).

talking about "distance": it is the length of the straight line connecting the two observations defined as

$$d(x, y) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}.$$

Another famous metric is the *Manhattan-block distance* (city-block distance, L_1-norm) which is simply the sum of the component-wise absolute differences.[1]

$$d(x, y) = \sum_{i=1}^{p} |x_i - y_i|.$$

Analogously, we can think of the analogue concept of a *similarity measure*, i.e., a measure that describes how much alike two observations are.

Definition 4.3 (Similarity function)**.** A function $s : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is called a *similarity function* (or *similarity measure*) if the following conditions hold for all $x, y \in \mathbb{R}^p$

(s1) $s(x, y) = s(y, x)$ (symmetry),
(s2) $s(x, y) \leq s(x, x)$ (no object can be more similar to another object than to itself),
(s3) $s(x, y) \in [0, 1]$ (optional, but often required).

An often used similarity measure is Pearson's coefficient of correlation (cf.

[1] The name most likely stems from the way streets are arranged in a chess-board like manner in many modern metropolis like the Manhattan district in New York. Imagine the cross-roads to be the points. Then the distance between two points corresponds to the distance of the shortest walk path between them. The Euclidean distance would be the air-line distance that you actually cannot traverse in general.

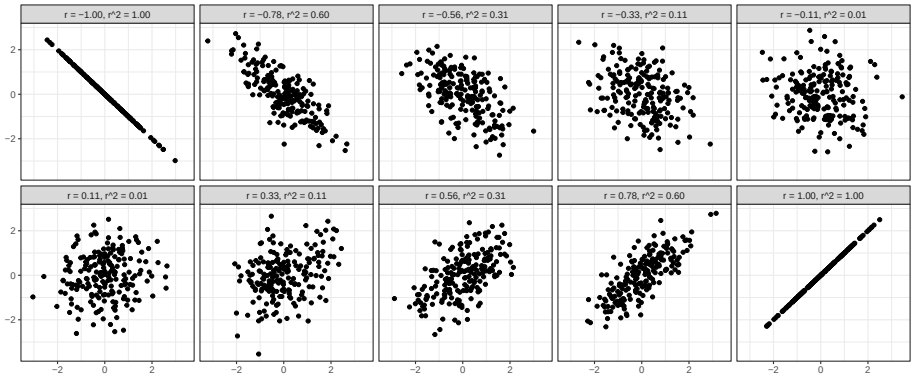


Figure 4.2: Samples from bi-variate $\mathcal{N}(\mu, \Sigma)$-distributions with different specified Pearson correlation r_{xy}.

Definition A.16)

$$r_{xy} = \frac{s_{xy}}{s_x \cdot s_y} = \frac{\sum_{i=1}^{p}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{p}(x_i - \bar{x})^2 \cdot \sum_{i=1}^{p}(y_i - \bar{y})^2}} \in [-1, 1].$$

The rationale is that two elements are more similar the stronger they depend on each other. Perfect similarity is given if there is perfect (linear) relationship. The correlation satisfies (s1) and (s2) in Definition 4.3, but not (s3). Using square transformation though, we can make the measure satisfy (s3), i.e., $r_{xy}^2 \in [0, 1]$. Figure 4.2 shows examples of different correlation structures and the respective r_{xy} and r_{xy}^2 values.

Under certain conditions dissimilarity measures can be converted into similarity measures or the other way around. E.g., if d is a dissimilarity function and non-decreasing function of d is a similarity function. A popular transformation is

$$s(x, y) = \exp\left(-\frac{d(x, y)^2}{t}\right), t > 0.$$

I.e., the similarity between x and y drops exponentially with the distance between both points. Let us examine the requirements of Definition 4.3. (s1) is trivially met. $s(x, x) = \exp(0) = 1$ and for each $x \neq y$, since d is a distance function, $d(x, y) \geq 0$ and thus $s(x, y) \leq s(x, x)$ due to the non-decreasing nature of the exponential function. Thus, (s2) is satisfied. Finally, $\exp(x) > 0$ for all $x \in \mathbb{R}$ and $s(x, y) \leq s(x, x) = 1$. Therefore, even (s3) holds.

As you might imagine there is no silver bullet approach, no golden path to clustering. There are countless algorithms and even more variations. In the following we will introduce three basic method. These methods are very dif-

ferent in terms of the cluster model and each comes with its advantages and disadvantages. In practice it is usually necessary to experiment: try different approaches to obtain the best results.

4.2 k-Means Clustering

We will first introduce a simple, yet powerful and highly used clustering algorithm. The algorithm solves the problem discussed earlier in this section. I.e., it calculates a partition $C_1, \dots, C_k$ of the input data $\mathcal{X}$ such that every object $x \in \mathcal{X}$ is assigned to exactly one cluster. Interestingly, it builds upon the random clustering algorithm (see Algorithm 1). The algorithm called *k-means* was first introduced by MacQueen decades ago in 1967 [Mac67]. We will see later why it is named the way it is. The algorithm requires the parameter $1 \leq k \leq N$ which indicates the desired number of clusters. The core idea is to come up with clusters where the distances between points within the clusters is rather low while the distance between clusters is high. This follows our intuition as in "visible" clusters points within a cluster are likely to be grouped together closely while there is a gap to other clusters. Mathematically, this idea can be expressed as the so-called *within-cluster sum of squares*. For a single cluster C it is simply the mean sum of squared Euclidean distances[2] of each pair of points in the cluster:

$$W(C) = \frac{1}{|C|} \sum_{x,y \in C} ||x - y||^2. \tag{4.1}$$

Now k-means aims to minimize the objective function

$$\min_{C_1,\dots,C_l} \sum_{l=1}^{k} \frac{1}{|C_l|} \sum_{x,y \in C_l} ||x - y||^2. \tag{4.2}$$

From a statistical perspective there is an equivalent interpretation as stated in the following Theorem which will prove useful for understanding how our actual implementation of the algorithm works.

Theorem 4.1. *For each data set $\mathcal{X}$ and $C \subset \mathcal{X}$ it holds that*

$$\frac{1}{|C|} \sum_{x,y \in C} ||x - y||^2 = 2 \sum_{x \in C} ||x - \mu_l||^2 = 2 \cdot |C| \cdot Var(C). \tag{4.3}$$

Here, $\mu_l = \frac{1}{|C|} \sum_{x \in C} x$ is the mean vector of the l^{th} cluster.

[2] The notation is to be read as follows: $||x - y||^2 = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}^2 = \sum_{i=1}^{p} (x_i - y_i)^2$. I.e., the squared Euclidean distance between the two points x and y.

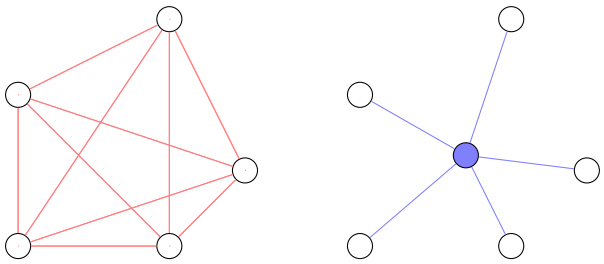


Figure 4.3: Exemplary illustration of the pairwise distances $\sum_{x,y \in C_l} ||x - y||$ (left) and the distance to the cluster center $\sum_{x \in C_l} ||x - \mu_l||$ (right).

This means that the within-cluster sum of squares (see Eq. 4.1) is the same as twice the sum of squared distances of the points assigned to the cluster which is equivalent to two times the (scaled) within-cluster variation. See Fig. 4.3 for a illustration of the distances involved.

Proof. The proof is a little bit odd and lengthy.

$$\begin{aligned}
\frac{1}{|C|} \sum_{x,y \in C} ||x - y||^2 &= \frac{1}{|C|} \sum_{x,y \in C_l} ||(x - \mu_l) - (y - \mu_l)||^2 \\
&= \underbrace{\frac{1}{|C|} \sum_{x,y \in C} ||x - \mu_l||^2}_{\text{independent of } y} + \underbrace{\frac{1}{|C|} \sum_{x,y \in C} ||y - \mu_l||^2}_{\text{independent of } x} \\
&\quad - \frac{2}{|C_l|} \underbrace{\sum_{x,y \in C} (x - \mu_l)^T (y - \mu_l)}_{=0} \\
&= \frac{|C_l|}{|C_l|} \sum_{x \in C} ||x - \mu_l||^2 + \frac{|C|}{|C|} \sum_{y \in C} ||y - \mu_l||^2 + 0 \\
&= 2 \sum_{x \in C} ||x - \mu_l||^2.
\end{aligned}$$

With this we can finally derive

$$2 \sum_{x \in C} ||x - \mu_l||^2 = 2 \cdot \underbrace{|C| \cdot \frac{1}{|C|}}_{=1} \sum_{x \in C} ||x - \mu_l||^2 = 2 \cdot |C| \cdot \mathrm{Var}(C)$$

which completes the proof. □

In my eyes a minor reformulation of the theorem is more memorable:

$$\frac{1}{2 \cdot |C|^2} \sum_{x,y \in C} ||x - y||^2 = \frac{1}{|C|} \sum_{x \in C} ||x - \mu_l||^2 = \mathrm{Var}(C). \tag{4.4}$$

Using Theorem 4.1 the objective function in Eq. 4.2 can be equivalently replaced with

$$\min_{C_1,\dots,C_k} \sum_{l=1}^{k} 2 \cdot |C_l| \cdot \mathrm{Var}(C_l). \tag{4.5}$$

We can even drop the constant factor 2 as it does not affect the solution (Why?). Unfortunately we need to cope with some bad news. The problem of minimising Eq. 4.2 or Eq. 4.5 respectively is actually very hard. Indeed finding the optimal cluster assignment is $\mathcal{NP}$-hard as shown by Aloise et al. [Alo+09]. Computational complexity is far beyond the scope of this lecture and hence we cannot introduce the concept of $\mathcal{NP}$-hardness here.[3] You can just think of the following: there is no (known) algorithm that given any data set $\mathcal{X}$ calculates the optimal k-means clustering within reasonable time limits and it is unlikely that such an algorithm exists. Therefore, in order to solve clustering problems with k-means in practice – and this is what we want to do here – we need to rely on heuristic algorithms. These kind of algorithms usually cannot guarantee to find the best clustering, but they are capable of finding locally optimal cluster assignments. In the following we present a simple heuristic which goes back to Lloyd [Llo82]. It approximates a solution via iterative refinement as follows: given the desired number of clusters k first assign each point uniformly at random to one of the k clusters (recall Algorithm 1).[4] Next, calculate the mean point $\mu_l = \frac{1}{|C_l|} \sum_{x \in C_l} x$, also called the *centroid*, *cluster center* or *center of mass*, for each cluster $l = 1, \dots, k$. Now re-label the points such that each point $x \in \mathcal{X}$ is assigned to the cluster center located closest (this is the refinement). Formally, cluster $C_l, l = 1, \dots, k$ is re-defined as

$$C_l = \{x \in \mathcal{X} \mid ||x - \mu_l||^2 = \min_{j=1,\dots,k} ||x - \mu_j||^2\}.$$

Iterate this process until some termination condition is met; usually iterate until the cluster assignment does not change anymore (it converges) or a pre-specified number of iterations is reached. See Algorithm 2 for pseudo-code. It is easy to see that the algorithm in each iteration improves upon Eq. 4.2. There are many variations of this basic algorithm which improve on the runtime, introduce an

[3] I am sure though that most of you do not want this either ☺.

[4] *Uniformly at random* means that each point is assigned to each cluster with probability $1/k$.

Algorithm 2 k-means clustering (Lloyd's version [Llo82])

Require: Data set $\mathcal{X}$, number of clusters k.

1: Assign each $x \in \mathcal{X}$ to one of the k clusters uniformly at random.
2: Build cluster centres/centroids $\mu_l = \frac{1}{|C_l|} \sum_{x \in C_l} x$ for $l = 1, \ldots, k$.
3: Assign each point $x \in \mathcal{X}$ to its closest centroid, i.e.
$C_l = \{x \in \mathcal{X} \mid ||x - \mu_l||^2 = \min_{j=1,\ldots,k} ||x - \mu_j||^2\}$.
4: Repeat from step 2 if termination condition is not met; quit otherwise.

alternative initialisation (e.g. the so-called *Forgy*-approach is to select k out of the N observations as the initial cluster centres) etc. However, understanding Loyd's version is essential to understanding the manifold modifications.

We can see the algorithm in action with $k = 3$ on the iris data set (2-dimensions `sepal.width` and `petal.width` used) in Figure 4.4. We can observe that after the chaotic initial assignment the algorithm converges quite fast. Figure 4.4 shows another run on the same data set which yields a very different and unexpected result. Here, the algorithm converged into a local optimum with respect to its objective function where it got trapped. This simple example emphasises the the necessity to restart the algorithm multiple times (say 25 or 50) for a given k and to return the best clustering found (with respect to the objective function) as the quality over the runs may vary significantly.

4.2.1 On the Right Choice of k

There is one important point left that needs thorough discussion: how to choose the parameter k in practice? This parameter has to be set *a-priori* which is indeed one of the major drawbacks of k-means clustering since in practice we have no clue how many heterogeneous groups there are; if we knew we would not need clustering at all. One common heuristic is the so-called *elbow-method*. Here, we experiment with a series of values for k, i.e., $k = 1, 2, 3, \ldots, K$. For each value we run the algorithm multiple times and log the best WCSS (see Eq. 4.2). We then draw a line-plot with k on the ordinate/x-axis and WCSS on the abscissa/y-axis and search for an "elbow" in the plot, i.e., a value of k^* such that the within-cluster sum of squares drops strongly for all $k < k^*$, but decreases only slightly for $k > k^*$. Fig. 4.6 shows the elbow-plot for the iris-flower data set. We can identify an elbow at $k = 2$ or $k = 3$ where the latter is the "true" number of groups here. Note that in practice we do not know the class labels. However, explaining pros and cons of clustering approaches is easier if we can compare the output with some kind of ground truth for validation. An alternative is the so-called *gap statistic* introduced by Tibshirani et al. [TGH01]. The method builds upon the idea of the elbow-method. However, it is more sophisticated

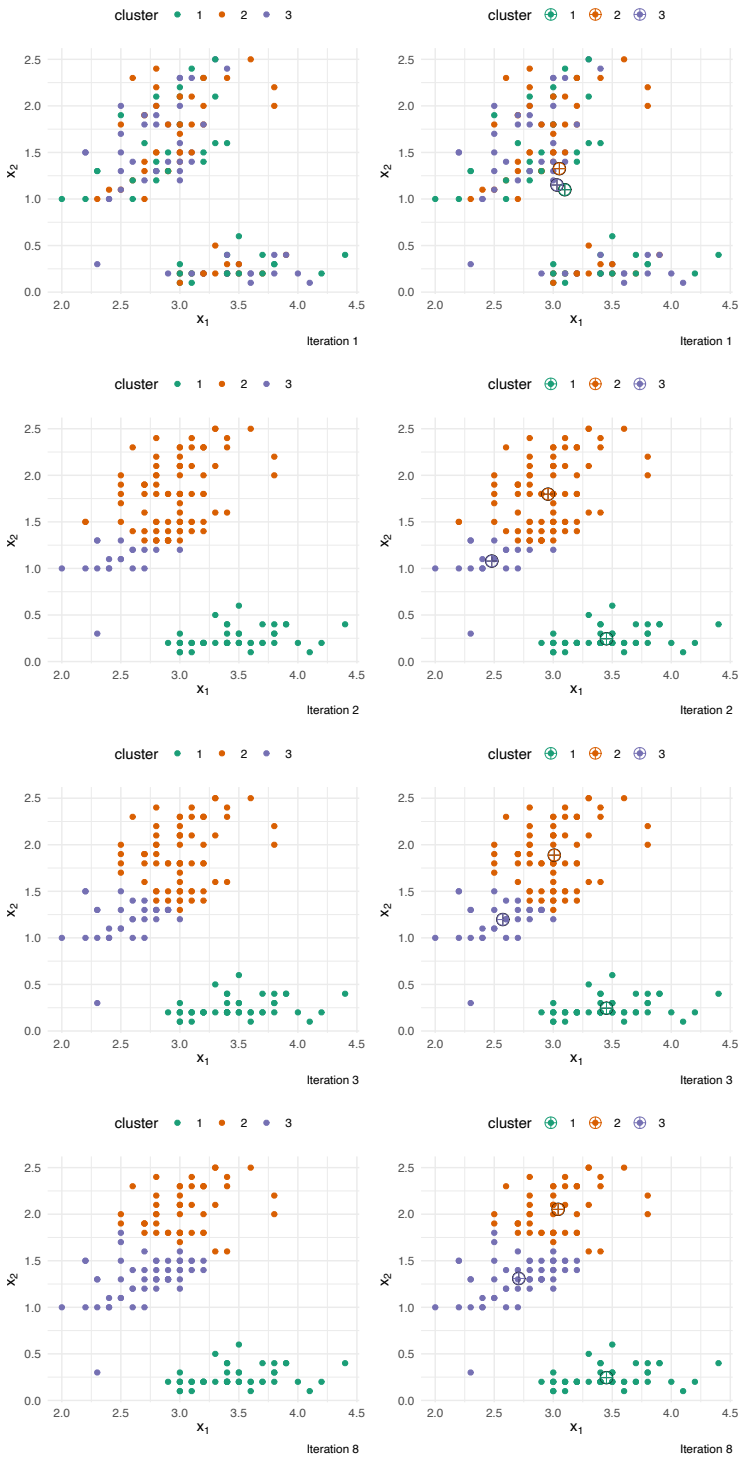


Figure 4.4: Exemplary run of k-means on Fisher's iris flower data set. Each row shows the cluster assignment in the respective iteration without (left) and with recalculated cluster centres (right). The algorithm converges after 8 iterations (iterations 4 to 7 are omitted here due to little changes).

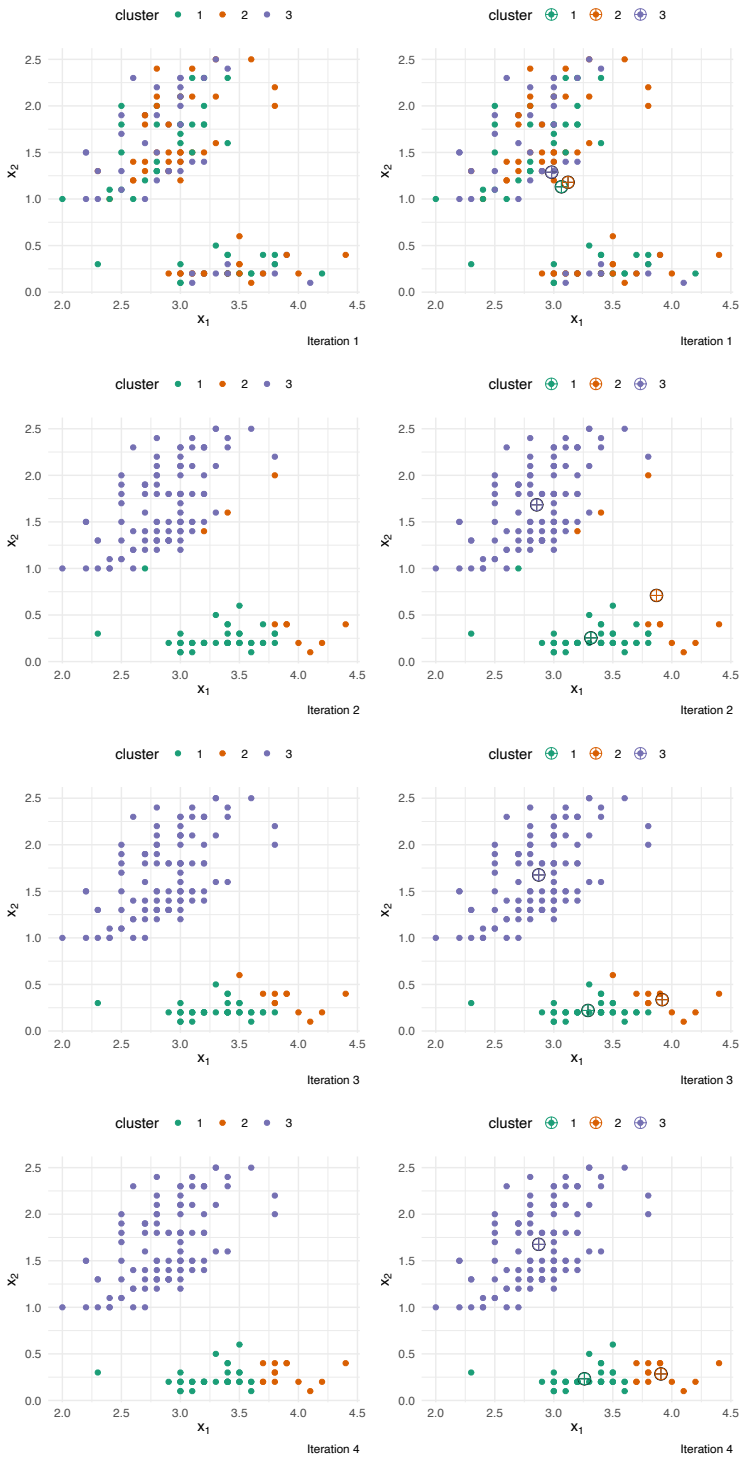


Figure 4.5: Another run of k-means on Fisher's iris flower data set. The algorithm converges after 4 iterations with a completely different results (compare to Figure 4.4 due to the algorithms stochasticity).

Figure 4.6: Elbow-method on iris. The plot suggests that $k = 2$ is the best choice since the drop in the total WCSS is steepest from $k = 1$ to $k = 2$.

and mathematical. With the words of [TGH01] the core idea is to "standardise the graph of $\log(W_k)$ by comparing it with its expectation under an appropriate null reference distribution of the data". This sentence is quite overwhelming. Let's slowly dive into the details. Ignoring the log-transformation aspect the core idea is to compare the WCSS for different k with the WCSS obtained on some reference data. The reference data here is just a sample of N points sampled uniformly at random within the boundaries of the data points in our data $\mathcal{X}$. Note that for data sampled uniformly at random clustering does not make much sense. This will be visible in a respective elbow-plot: we won't see any elbow but a slow decrease in W_k. The method now works as follows:

1. Run k-means for varying $k = 1, 2, \ldots, K$ and calculate W_k
2. Generate B reference sets sampling uniformly at random within the bounds of the data. This yields WCSS values W^*_{kb} for each combination of $b = 1, \ldots, B$ and $k = 1, \ldots, K$.
3. Calculate the *estimated gap statistic* as the difference between the average log-transformed W^*_{kb} values and the log-transformed W_k of the data, i.e.,

$$\text{Gap}(k) = \underbrace{\left(\frac{1}{B}\sum_{b=1}^{B}\log(W^*_{kb})\right)}_{=:W^*} - \log(W_k).$$

4. Compute the standard deviations of the reference set WCSS values as

$$\text{sd}_k = \sqrt{\frac{1}{B}\sum_{b=1}^{B}\left(\log(W^*_{kb} - W^*\right)^2}$$

and set $s_k = \text{sd}_k \cdot \sqrt{1 + 1/B}$ (kind of regularisation for technical reasons;

see [TGH01] for details). We call the latter values $s_k, k = 1, \ldots, K$ *standard-error* in the following.

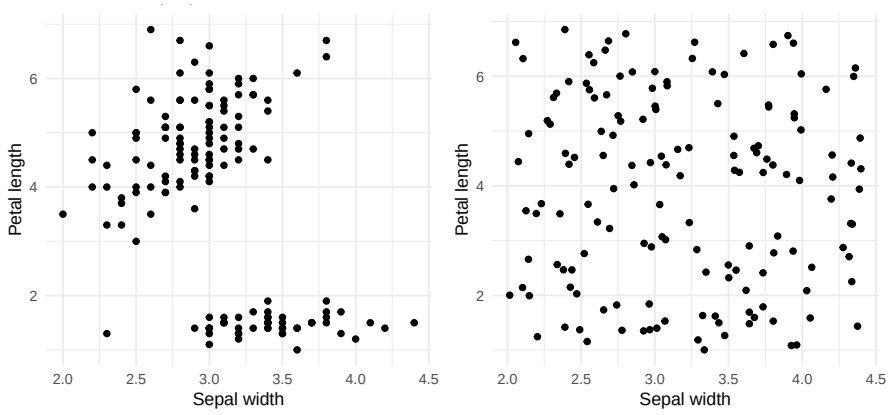
Now we can choose the k^* that maximises the gap statistic. I.e., the number of clusters where the deviation from the expectation is largest. This approach works well if the data is clearly well-separated into clusters. In many real-world data-sets however, the separation is less obvious and we need to strive to find a trade-off between the maximisation of the gap statistic and the model assumptions. To do so, [TGH01] propose to select k^* such that

$$k^* = \arg\min_{k=1,\ldots,K} \text{Gap}(k) \geq \text{Gap}(k+1) - s_{k+1}. \tag{4.6}$$

Here, finally the standard deviations come into play. Formally, this means that we decide for the first k where the gap is larger than the gap for $(k+1)$ minus one standard-error to account for randomness. Informally, this criterion prefers the k where the gap statistic starts to slow down in a sense. Figure 4.7 illustrates the method on the iris data set. The gap statistic plot takes its maximum at $k = 5$. However, Eq. 4.6 suggests $k = 2$.

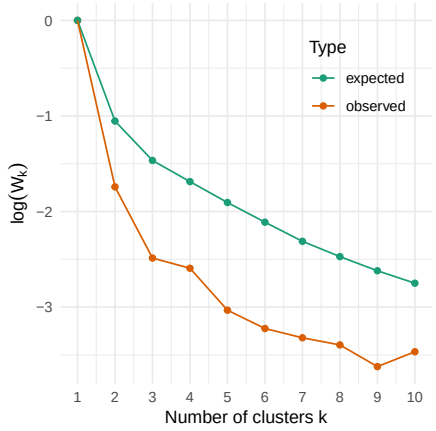
4.2.2 Runtime ★

Finding the global optimum to Eq. 4.2 is $\mathcal{NP}$-hard as already mentioned [Alo+09; MNV12]. Lloyd's local search algorithm (see Algorithm 2) finds a local – not necessarily global – optimum in time $O(t \cdot N \cdot k \cdot p)$ where N is the number of data points of dimension p, k is the specified number of clusters and t is the number of iterations (see Appendix B for an explanation of the O-notation). Why is this? Consider Algorithm 2. Assuming that sampling random numbers can be performed in constant time, line 1 takes time $O(N)$. Building cluster centres (line 2) requires $O(N \cdot p)$. Line 3 is apparently the most time-consuming step: Here, for each pair of cluster center and observation we need to calculate the distance which takes $O(N \cdot p \cdot k)$; re-assigning cluster centres is negligible. This process is now iterated over and over again until convergence (stable assignment that does not change from some iteration $t-1$ to t) after t iterations or a predefined number iterations t. In any case, the algorithm terminates after $O(t \cdot N \cdot k \cdot p)$ steps. Usually, in practice $N \gg p$ and k will be rather low. Hence, the runtime of the heuristic is pretty neat. However, keep in mind that we need to re-run the algorithm multiple times and for different k (see previous section). Luckily, these runs are independent of each other and thus parallel execution is possible.

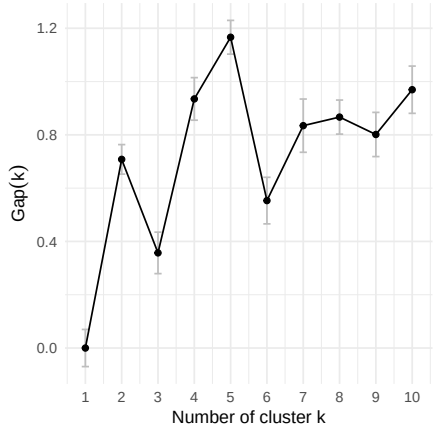


(a) Data set $\mathcal{X}$

(b) An exemplary reference set



(c) Elbow plot



(d) Gap statistic

Figure 4.7: Actual data set $\mathcal{X}$ (top left), an example of the reference set (top right), elbow plots of $\log(W_k)$ for the expected version based on reference sets and the observed data (bottom left) and the gap statistic (bottom right). Gray error-bars indicate $\text{Gap}(k) \pm s_k$, i.e., the gap plus/minus one standard-error. Here, the maximum is taken for $k = 5$, but according to the decision rule in Eq. 4.6 $k = 2$ is the better choice.

4.3 Hierarchical Clustering

One drawback of k-means is that the user is required to set k *a-priori*. *Hierarchical clustering* resp. *hierarchical cluster analysis* (HCA) takes a different approach. Here, the plain-simple *Hierarchical Agglomerative Clustering* (HAC; see Algorithm 3) algorithm starts with N clusters (each observation forms its own cluster). In $N-1$ steps *fusion* of clusters occurs: in each step the two *closest* clusters are merged; see Algorithm 3 for high-level pseudo-code.[5] The key challenge is now to define what *closest* means with respect to sets of points. This depends on the distance/dissimilarity measure chosen (giving the distance between two points) and a so-called *linkage* measure which extends the dissimilarity measure two sets of multiple points.

Again there is a multitude of possibilities to define a distance between point sets. We present the most common in the following (see Figure 4.8 for visualisations). In *single-linkage* we define it to be the minimum distance between points of both sets, i.e.,

$$D(C_i, C_j) := \min_{x \in C_i, y \in C_j} d(x, y).$$

Complete linkage instead takes the maximum point-wise distance

$$D(C_i, C_j) := \max_{x \in C_i, y \in C_j} d(x, y).$$

Average linkage defines the distance to be the mean over all pairwise distances.

$$D(C_i, C_j) := \frac{1}{|C_i| \cdot |C_j|} \sum_{x \in C_i} \sum_{y \in C_j} d(x, y)$$

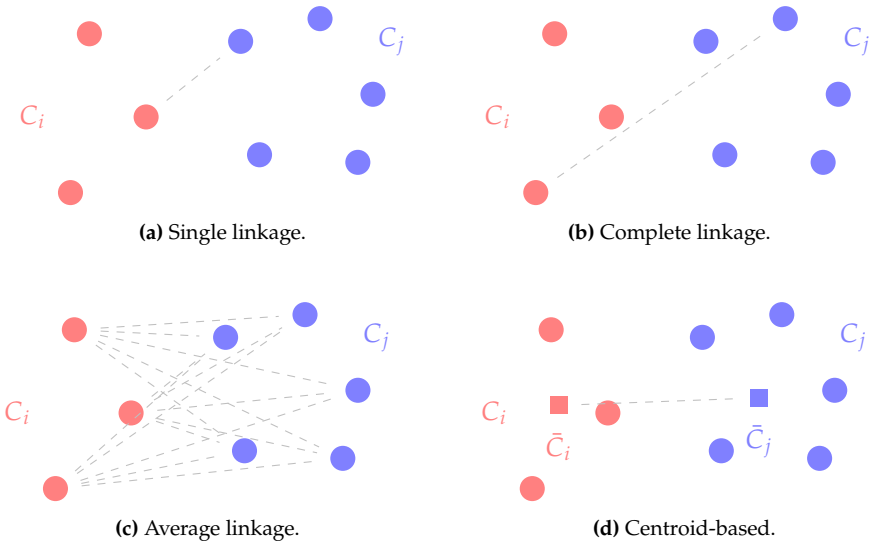
Finally, the centroid-based approach first calculates the mean vectors $\bar{C}_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$ and $\bar{C}_j = \frac{1}{|C_j|} \sum_{x \in C_j} x$, also called the *centroids* (cf. k-means), for each cluster and takes the distance between those as the distance of the clusters:

$$D(C_i, C_j) := d(\bar{C}_i, \bar{C}_j).$$

Let us apply HCA with single-linkage to a simple set of four observations $\{a, b, c, d\}$ with pairwise distances given by the following dissimilarity/distance matrix.[6] The row- and column-names indicate the respective clusters. In

[5] Alternatively one can start with a single cluster and iteratively split clusters until each observations forms its own cluster. This approach is known as the *divisive* approach, but is used less often in practice.

[6] A dissimilarity matrix or distance matrix contains the pairwise distances between observations/clusters.



(a) Single linkage. **(b)** Complete linkage. **(c)** Average linkage. **(d)** Centroid-based.

Figure 4.8: Visualization of the different linkage functions on sample data.

Algorithm 3 (Agglomerative) Hierarchical Agglomerative Clustering (HAC)

Require: Data set $\mathcal{X}$, linkage function D
1: Assign each observation $x \in \mathcal{X}$ its own cluster.
2: Calculate all pairwise inter-cluster dissimilarities with D.
3: **for** $i \leftarrow 1, \ldots, N-1$ **do**
4: Find the two most similar clusters and merge them.
5: Update inter-cluster dissimilarities.
6: **end for**

the beginning each observation forms its own cluster (cf. Algorithm 3):

$$\begin{array}{c} \\ \{a\} \\ \{b\} \\ \{c\} \\ \{d\} \end{array} \begin{array}{c} \begin{array}{cccc} \{a\} & \{b\} & \{c\} & \{d\} \end{array} \\ \begin{pmatrix} 0 & 0.3 & 0.4 & 0.7 \\ 0.3 & 0 & 0.5 & 0.8 \\ 0.4 & 0.5 & 0 & 0.8 \\ 0.7 & 0.8 & 0.8 & 0 \end{pmatrix} \end{array}. \tag{4.7}$$

The minimum distance is between clusters $\{a\}$ and $\{b\}$. Therefore, we merge these two clusters at a height of $D(\{a\},\{b\}) = 0.3$ in the so-called *dendrogram* (see Figure 4.9). The dendrogram is a tree-like structure that we construct during the HCA-algorithm's execution. On the ordinate it contains the single observations. Every time a fusion takes place vertical lines from the merged clusters are drawn at the height/distance where the clusters were merged. We will discuss the interpretation later. Let's get back to our example. We get $D(\{a,b\},\{c\}) = \min\{D(\{a\},\{c\}), D(\{b\},\{c\})\} = \min\{0.4, 0.5\} = 0.4$ and

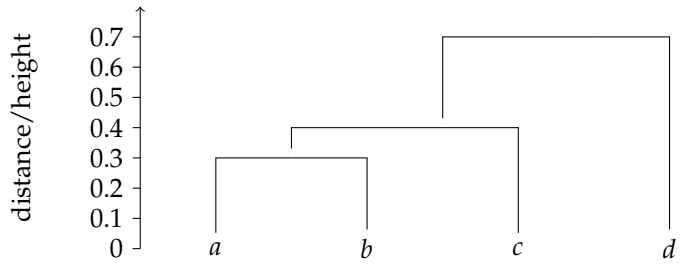


Figure 4.9: Dendrogram obtained from HCA with single-linkage on the distance matrix given in Eq. 4.7. The large gap between $\{a, b, c\}$ and $\{d\}$ indicates that two clusters are the best choice.

$D(\{a,b\},\{d\}) = \min\{D(\{a\},\{d\}), D(\{b\},\{d\})\} = \min\{0.7, 0.8\} = 0.7$. With these values we can update the dissimilarity matrix to

$$\begin{array}{c} \\ \{a,b\} \\ \{c\} \\ \{d\} \end{array} \begin{array}{c} \begin{array}{ccc} \{a,b\} & \{c\} & \{d\} \end{array} \\ \begin{pmatrix} 0 & 0.4 & 0.7 \\ 0.4 & 0 & 0.8 \\ 0.7 & 0.8 & 0 \end{pmatrix} \end{array}$$

and the next iteration begins. The minimum distance is $D(\{a,b\},\{c\}) = 0.4$. In consequence clusters $\{a, b\}$ and $\{c\}$ are merged at height 0.4. The distance of the augmented cluster $\{a, b, c\}$ to cluster $\{d\}$ is according to single-linkage $D(\{a,b,c\},\{d\}) = \min\{D(\{a,b\},\{d\}), D(\{c\},\{d\})\} = \min\{0.7, 0.8\} = 0.7$. Thus, the updated dissimilarity matrix is

$$\begin{array}{c} \\ \{a,b,c\} \\ \{d\} \end{array} \begin{array}{c} \begin{array}{cc} \{a,b,c\} & \{d\} \end{array} \\ \begin{pmatrix} 0 & 0.7 \\ 0.7 & 0 \end{pmatrix}. \end{array}$$

The last step is always obvious as there are only two clusters left and there is no room for decisions. Hence, we merge clusters $\{a, b, c\}$ and $\{d\}$ at height 0.7 and the algorithm terminates.

Looking at the dendrogram in Figure 4.9, we could *cut* the dendrogram at, say height 0.6, to obtain two clusters $C_1 = \{a, b, c\}$ and $C_2 = \{d\}$ which seems most plausible. In general a rule of thumb is to look for the largest gap between two merged clusters.

4.3.1 Runtime ★

The runtime of the general algorithm is $O(N^3)$ since there are exactly $N - 1$ merge steps and in each step we need quadratic time to update distances. The

memory requirement is $\Omega(N^2)$ due to the distance matrix storage. The runtime can be reduced to $O(N^2 \log N)$ using smarter data-structures. However, this improvement in runtime goes hand in hand with even higher space complexity. As a consequence, both versions are rather slow and infeasible for large data-sets. For special cases of problems often properties of linkage-functions can be leveraged to design better algorithms. SLINK [Sib73] for single-linkage and CLINK [Def77] for complete-linkage based HC are examples with $O(N^2)$ runtime.

4.4 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

So far we considered two types of clustering algorithms. So-called *partition-based* algorithms like k-means partition the feature space into k predefined areas (the clusters) minimising a cost function. Subsequently, each point $x \in \mathcal{X}$ is assigned a unique cluster, namely the cluster whose center of mass (the cluster center) is closest. This procedure implies that (1) every single point is assigned a cluster and (2) the shape of the clusters is necessarily convex.[7] To see why both properties are generally undesired and pose severe restrictions consider the ☺ data set in Figure 4.10. This is certainly a toy example, but it nicely depicts the drawbacks of partition-based approaches. The data set obviously consists of four clusters. However, the mouth is a non-convex shape (check the definition of convexity). The second figure shows the smiley data extended by some noise. Noise points are points that cannot be assigned to any cluster and seem to be rather anomalies, e.g., measurement errors in case of data collected by some kind of sensor (see Chapter 3). Now consider the k-means clustering with $k = 4$ in right-most plot in Figure 4.10. Here, the points are coloured by the cluster assignment calculated by the k-means algorithm. In addition, the background is coloured the same way. The background colour indicates which cluster a point would be assigned if it would reside exactly at that place. The assignment is clearly of convex structure. Obviously, the clustering result is unsatisfactory since both (1) and (2) are not handled appropriately. In this section we will introduce an alternative clustering approach called DBSCAN for ***D**ensity **B**ased **S**patial **C**lustering of **A**pplications with **N**oise*. The meaning of the lengthy name will become clear as we dive into the working principles. The algorithm was proposed back in 1996 in a seminal paper by Ester et al. [Est+96]. Due to its huge impact this paper was awarded the *test of time award* by the ACM

[7] A shape C is called *convex* if for any two points $p, q \in C$ in the shape all points on the straight line between x and y are in C, too.

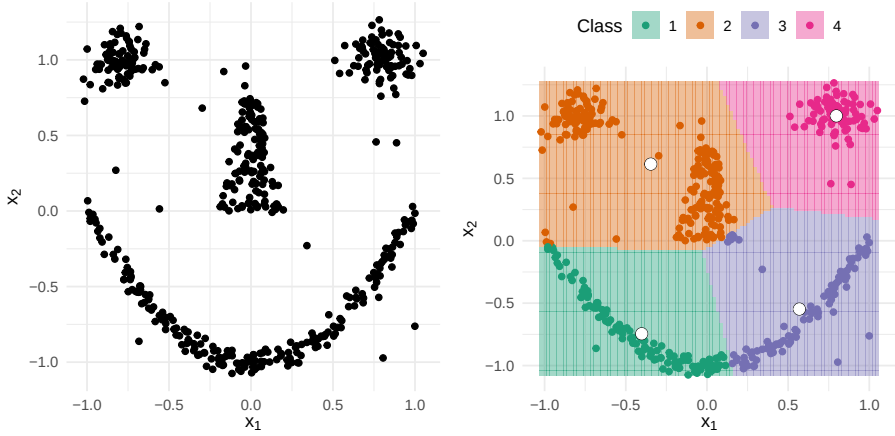


Figure 4.10: Smiley data set augmented with some noise/outliers (left) and k-means clustering with $k = 4$ (right).

SIGKDD conference, a major AI conference, in 2014.[8] With 22 881 citations[9] it is one of the most cited papers on clustering in the literature and the algorithm itself and its many modifications still enjoys high popularity in applications and theory [Sch+17]. You see, it seems worthy to learn how it works!

The rough idea is to first define a formal model of "density" and later on derive an algorithm that iteratively applies this formal model. In a nutshell the model relies on the following intuition: points that belong to clusters are located in rather dense areas of the feature space (they have many neighbours) while noise points are located in more or less uncrowded regions. Based on this the algorithm defines that a point belongs to a cluster if it is "surrounded" by a minimal number of neighbour points.[10] Based on this model DBSCAN takes two parameters `minPts` and $\varepsilon > 0$. It then selects a random point to start with. If there are more than `minPts` points within distance at most ε of this point (including the point itself) the algorithm starts forming a cluster. The cluster eventually grows by checking the same conditions for all the neighbours. If a cluster cannot grow any more the process starts anew from another random, not yet processed, point.

We will now formalise this hand-wavy explanation. In order to do so we need to mathematically define the notion of neighbourhood. A neighbourhood is defined on top of a distance measure: "close" points are neighbours while "distant" points are not considered neighbours. So let us assume that we have a distance function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which assigns a distance value $d(x, y)$ for

[8] URL: http://www.kdd.org/News/view/2014-sigkdd-test-of-time-award

[9] Queried Google scholar on 21 November, 2021 at 5pm.

[10] Interactive animation: https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

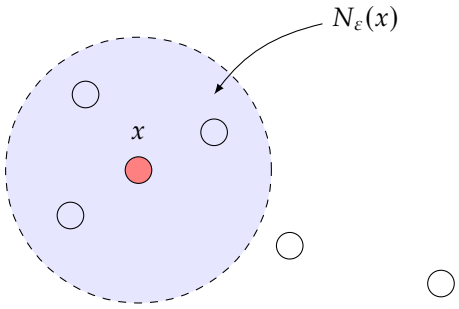


Figure 4.11: ε-neighborhood $N_\varepsilon(x)$ of a point x for Euclidean distance.

each two points $x, y \in \mathcal{X}$. In the following we will work with the Euclidean distance, but DBSCAN does not pose any restrictions on the choice of the distance function.

Definition 4.4 (ε-neighbourhood)**.** The *ε-neighbourhood* of a point $x \in \mathcal{X}$ for some $\varepsilon > 0$ is defined as

$$N_\varepsilon(x) := \{y \in \mathcal{X} \mid d(x, y) \leq \varepsilon\}.$$

That means that for a given "radius" $\varepsilon > 0$ the neighbourhood of x contains all other points that have a distance of at most ε to x. Certainly, this includes x itself for any valid ε. Figure 4.11 shows an illustration of the neighbourhood in the 2-dimensional space for the Euclidean distance.
Given a parameter `minPts` for *minimal number of points* and $\varepsilon > 0$ we define that a point x is a *core point* if $|N_\varepsilon(x)| \geq$ `minPts`. This definition alone however is not enough to characterize clusters, since naturally points near the cluster border will have less neighbors than points in the center of mass. We call such points *border points* in the following (see Figure 4.12) for an example. Hence, we extend our set of definitions accordingly:

Definition 4.5 (directly density-reachable)**.** A point $x \in \mathcal{X}$ is *directly density-reachable* from $y \in \mathcal{X}$ with regard to ε and `minPts` if

$$(1) \quad x \in N_\varepsilon(y) \quad \text{and} \quad (2) \quad |N_\varepsilon(y)| \geq \texttt{minPts}.$$

Hence, for every point $x \in C$ in a cluster $C \subset \mathcal{X}$ we require that there exists another point $y \in C$ such that x is in the neighborhood of y and y is a core point (see Figure 4.13).
This definition can be extended quite naturally leading to

Definition 4.6 (density-reachable)**.** A point $x \in \mathcal{X}$ is *density-reachable* from $y \in \mathcal{X}$

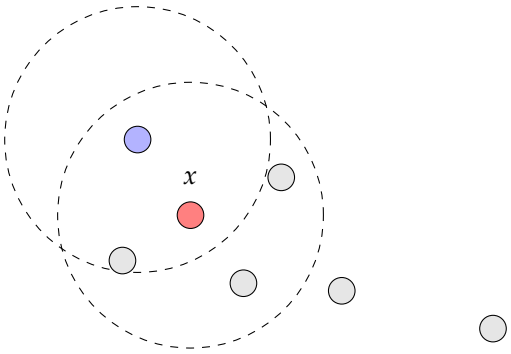


Figure 4.12: In this example the blue point is a border point of the cluster while the red point is a core point for `minPts`= 4.

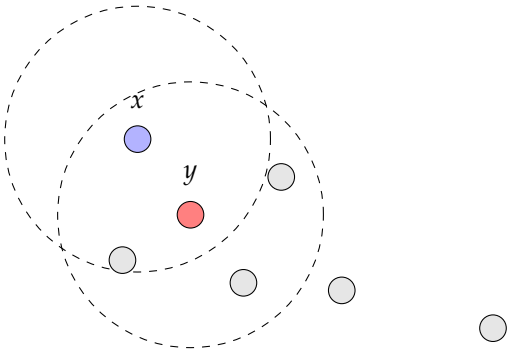


Figure 4.13: In this example, for `minPts` = 3, point x is directly density-reachable from point y.

with regard to `minPts` and $\varepsilon > 0$ if there is a chain/sequence of points $x_1, \ldots, x_l$ such that $x_1 = y$, $x_l = x$ such that x_{i+1} is directly density-reachable from x_i for $1 \leq i < l$.

Unfortunately, this relation is not symmetric as can be seen in Figure 4.14: point y is density-reachable from point x through the chain $y = x_1, x_2, x_3 = x$, but the vise-versa is not the case since x_2 is not directly density-reachable from $x_3 = y$. I.e., if x is density-reachable from y, the vice versa does not hold true in general! It can however. For an algorithmic implementation though, i.e., for the iterative extent of a cluster, symmetry is a necessary property. Hence, we extend the definition once again to so-called *density-connectedness* (see Figure 4.15 for an illustration).

Definition 4.7 (density-connected)**.** A point $x \in \mathcal{X}$ is *density-connected* with a point $y \in \mathcal{X}$ with regard to `minPts` and $\varepsilon > 0$ if there is a point $z \in \mathcal{X}$ such that x and y are both density-reachable from z.

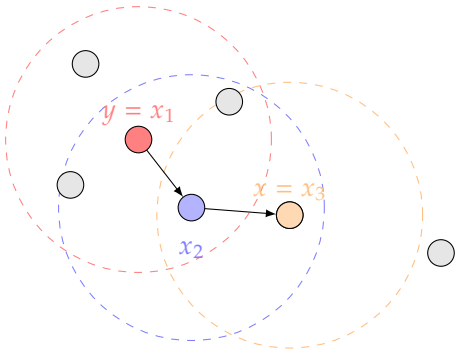


Figure 4.14: Here, for `minPts`= 4, x is density-reachable from y via $y = x_1, x_2, x_3 = x$. However, y is not density-reachable from y.

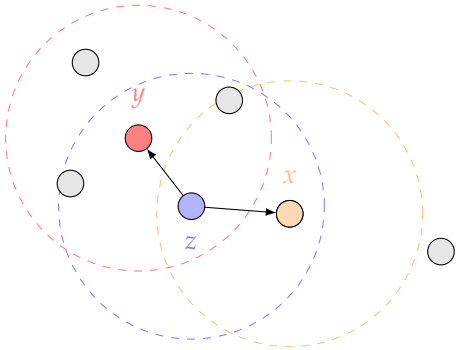


Figure 4.15: Here, both x and y are density-reachable from z (for `minPts` = 4). Hence, x and y are density-connected (note the direction of the arrows).

Based on these definitions a DBSCAN cluster is defined as follows:

Definition 4.8 (DBSCAN cluster)**.** A cluster C is a subset of the the data set $\mathcal{X}$ such that the following two conditions hold:

1. $\forall x, y$: if $x \in C$ and y is density-reachable from x w.r.t. `minEps` and ε, then $y \in C$. (Maximality)
2. $\forall x, y \in C : x$ is density-connected to y w.r.t. `minEps` and ε. (Connectivity)

Now we are able to translate these concepts into an actual algorithm. Detailed steps are outlined in the pseudo-code in Algorithm 4 which basically allows for a direct translation into real code. The algorithm operates as follows: first, the cluster assignment is initialised to `undefined` for all points $x \in \mathcal{X}$. Then, the algorithm iterates over all points in an outer loop. For each point $x \in \mathcal{X}$ it checks the neighbourhood $N_\varepsilon(x)$. If the size of the neighbourhood does not contain at least `minPts` points, x is marked as noise. Note that while cluster assignments are permanent (i.e., once a point is assigned to a cluster, the assignment will not

Algorithm 4 Density-Based Spatial Clustering for Applications with Noise

Require: Data set $\mathcal{X}$, radius $\varepsilon \geq 0$, minimal number of points to be considered a core point `minPts`, distance function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$.

```
C ← 0
cluster(x) ← undef ∀x ∈ X
for x ∈ X do
    if cluster(x) ≠ undef then
        next
    end if
    N_ε(x) ← getNeighbors(X, x, ε, minPts, d)
    if |N_ε(x)| < minPts then
        cluster(x) ← noise
        next
    end if
    C ← C + 1
    cluster(x) ← C
    L = N_ε(x) \ {x}
    for y ∈ L do
        if cluster(y) = noise then
            cluster(y) ← C
        end if
        if cluster(y) ≠ undef then
            next
        end if
        cluster(y) ← C
        N_ε(y) ← getNeighbors(X, y ε, minPts, d)
        if |N_ε(y)| ≥ minPts then
            L ← L ∪ N_ε(y)
        end if
    end for
    return cluster
end for
```

change), points marked as noise can later be relabelled if the point turns out to be density-reachable from another point. Otherwise, if $|N_\varepsilon(x)| \geq$ `minPts`, the point is a core point and is assigned a cluster label. Next, all points $y \in N_\varepsilon(x) \setminus \{x\}$ are traversed in an inner loop. Here, a case distinction takes place: noise points may be relabelled if they are density-reachable/density-connected while points which are already assigned to different clusters are left untouched.

Let's apply DBSCAN on our ☺ data set with noise. Figure 4.16 compares both algorithms. DBSCAN impressively shows its capability to identify non-convex, arbitrarily shaped clusters.

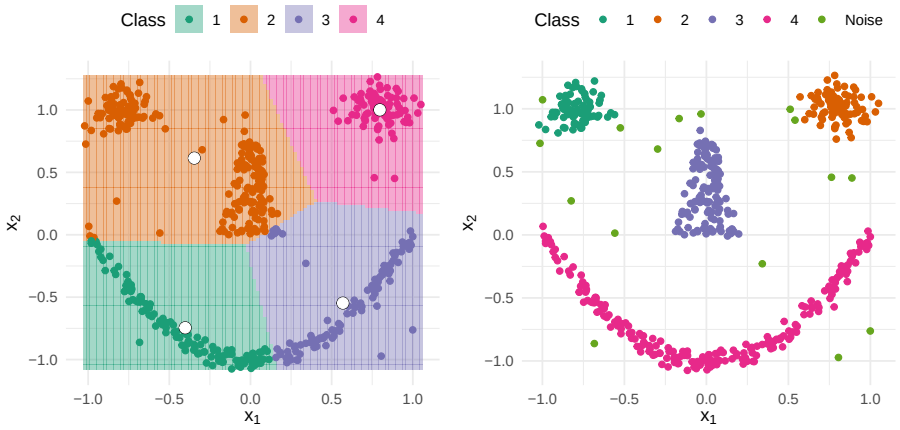


Figure 4.16: k-means with $k = 4$ on smiley data with noise (left) and DBSCAN with $\varepsilon = 0.1$ and `minPts`= 5 (right).

4.4.1 On the Choice of the Parameters

It is easy to see that DBSCAN is strongly sensitive to the choice of ε and/or `minPts`. Figure 4.17 illustrates the problem if inappropriate parameters are chosen. How can we determine good parameter values? Ester et al. [Est+96] propose a simple, appealing and effective heuristic. They define the so-called *k-dist function*

$$k\text{-dist} : \mathcal{X} \to \mathbb{R}^+$$

which maps each data point $x \in \mathcal{X}$ to its k^{th}-nearest neighbour (even though arbitrary values of k are possible, experiments in [Est+96] show that $k = 4$ is usually a good choice). Next, they propose to sort the k-dist values in descending order and to draw the respective graph. Figure 4.18 shows the *sorted k-dist graph* for our smiley example. Now in this graph they recommended to search for the first "valley", i. e., a threshold point x', such that at $(x', k\text{dist}(x'))$ the curve starts to slow down its downward trend. Finally, set $\varepsilon = k\text{-dist}(x')$ and `minPts`= k. Looking at Figure 4.18 the heuristic recommends $\varepsilon = 0.1229$. In Figure 4.19 we see the results of DBSCAN with this choice of ε and a much higher value of 0.3705 (see upper dashed line in Figure 4.18). Apparently, the heuristic choice yields superior results. The rationale behind this approach is that for a given k, with d being the k^{th} nearest neighbour of a point (i) $|N_d(x)|$ will contain exactly $k + 1$ points with high probability and (ii) reducing k will not have a strong effect on d for a cluster point. Now selecting the threshold point x' in the sorted k-dist graph implies – according to Ester el al. [Est+96, p. 4] – that (a) all points to the left (in the sorted order of the k-dist graph) will

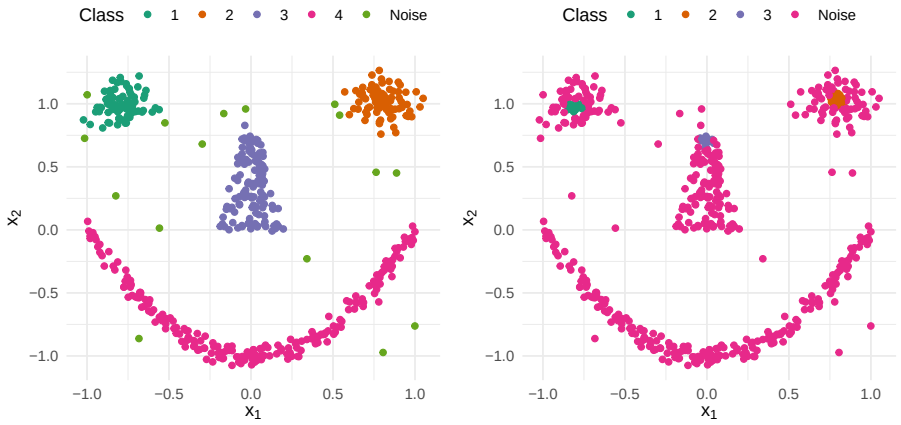


Figure 4.17: DBSCAN on smiley data set with different parameterisation: $\varepsilon = 0.1$ and `minPts`= 4 (left), $\varepsilon = 0.05$ and `minPts`= 15 (right). The result for the latter choice is obviously unsatisfying.

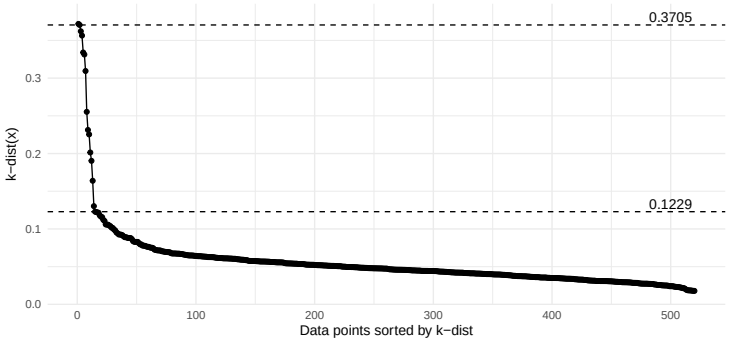


Figure 4.18: Sorted k-dist graph for $k = 4$ on our smiley data. The "elbow" at $\varepsilon = 0.1229$ seems to be a good choice according to the heuristic.

be declared noise points while (b) for all points to its right will belong to some cluster for $\varepsilon = k\text{-dist}(x')$ (since their k^{th} nearest neighbour is closer). We stress here, that we are unsure about the truth value of statement (a). These points can certainly be border-points (i. e., density-reachable from a core point). However, we have not yet been able to find a formal proof of the statement, nor have we been able to disprove it by counterexample.

4.4.2 Runtime ★

It was long claimed that the worst-case runtime is $O(N \log N)$ [Est+96]. However, just recently in 2015, Gan and Tao [GT15], showed that the worst-case complexity is indeed $O(N^2)$. This is actually intuitively easy to see: for each point we need to calculate its ε-neighbourhood which takes worst-case linear

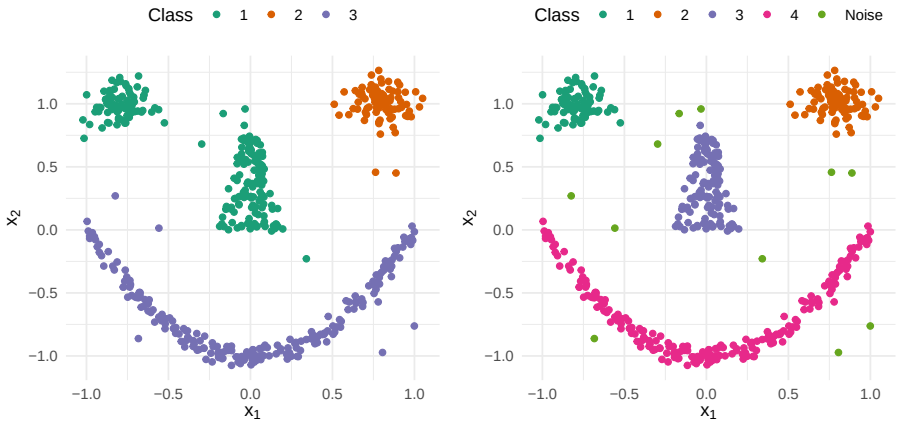


Figure 4.19: DBSCAN with `minPts` = 4 and (a) bad $\varepsilon = 0.3705$ (left) and (b) good $\varepsilon = 0.1229$ (right).

time. In case all points are within a distance of ε this step already takes $O(N)$ time and need to be performed N times. In particular due to its extensive use in applications it is quite surprising that the mis-claim persisted for 17 years. For $p = 2$ variables actually a $O(N \log N)$ is known, but for $p \geq 3$ it is an open problem if the algorithm can be "tweaked" towards better runtime. However, chances are low as Gan and Tao point out, since such a tweak would require a major runtime improvement for the *unit-spherical emptiness checking* problem (cf. [GT15, p. 4]) which is widely considered impossible. The authors however also propose a randomised algorithm which runs in expected linear time, i.e., $O(N)$, regardless of p, sacrificing accuracy for a massive speed-up. With respect to the ubiquitous availability of *big data*, such approximate algorithms are the future, since even quadratic time algorithms get computationally untraceable.

4.5 Measuring Cluster Quality

How do we measure the quality of a clustering? I.e., how can we evaluate how good a partition of the data actually is? This questions is as hard as the task of clustering itself. Basically, there are two approaches for cluster evaluation. In *external (extrinsic) evaluation* some ground truth or golden standard is known. I.e., we know the optimal partition $G_1, \ldots, G_l$ of the data. This is certainly the case for illustrative toy examples like our smiley data set. In practice a ground truth could be achieved by passed, labelled data, or manual identification of groups by experts. The more relevant approach is the so-called *internal (intrinsic) evaluation* since it is much more practical. Here, no ground truth is known. Instead, the evaluation is based on the clustering itself. Unsurprisingly, most

measures proposed for the latter approach use the notion of distance/dissimilarity and/or closeness/similarity. They all rely on the intuition that points assigned to the "right" cluster are rather close to all other points in the cluster and points from other clusters are more dissimilar.[11]

4.5.1 External Evaluation

We will now briefly discuss one external measure which was proposed by Rand [Ran71].

Definition 4.9 (Rand index)**.** Let $C = \{C_1, \ldots, C_k\}$ be a clustering and $G = \{G_1, \ldots, G_l\}$ a ground-truth partition. Let

- TP (true positive) be the number of pairs of elements in $\mathcal{X}$ which are in the same subsets of C and G,
- TN (true negative) be the number of pairs of elements in $\mathcal{X}$ which are in different subsets of C and G,
- FP (false positive) be the number of pairs of elements in $\mathcal{X}$ which are in the same subset of C but in different subsets of G,
- FN (false negative) be the number of pairs of elements in $\mathcal{X}$ which are in different subsets of C but in the same subset of G.

The *Rand index* is then calculated as follows

$$RI = \frac{TP + TN}{TP + FP + FN + TN} = \frac{TP + TN}{\binom{N}{2}} \in [0, 1].$$

TP, TN, FN and FP are disjoint sets. I.e., a pair of points is in exactly one of these set. The denominator $\binom{N}{2}$ stems from the fact that there are exactly $\binom{N}{2} = \frac{N(N-1)}{2}$ possibilities to choose two out of N elements. The rand index is percentage of correct cluster assignments. Obviously, the higher RI, the better the cluster quality. A value of 1 indicates that all pairs of points are either true positive or true negative, i.e., $FN = FP = 0$, and thus correctly assigned.

4.5.2 Internal Evaluation

We no switch to evaluating cluster quality based on the clustered data itself; the only way possible if no ground truth is available. There are more than a dozen different measures. Here, we briefly discuss Dunn's index [Dun74] and the index by Davies & Bouldin [DB79]. Furthermore, we spent more time on the silhouette [Rou87] due to its nice visual character. Unsurprisingly, all these

[11] Certainly, this is not necessarily true and is based on convex cluster shapes. Recall the smiley data. Here, points located in the corners of the mouth a quite dissimilar, but nevertheless belong to the same cluster.

measures in some way relate *intra*-cluster distances (which should be rather low) to *inter*-cluster distances (which should be rather high).
Let's start with Proposed by Dunn [Dun74].

Definition 4.10 (Dunn index)**.** For a clustering $C_1, \ldots, C_k$ the *Dunn index* is defined as

$$D(C_1, \ldots, C_k) := \frac{\min_{1 \leq i < j \leq k} d(C_i, C_j)}{\max_{1 \leq l \leq k} d'(C_l)}$$

where $d(C_i, C_j)$ is the *distance between the i-th and j-th cluster* and $d'(C_l)$ is the *intra-cluster distance* of cluster C_l.

In a good clustering the numerator should be high and the denominator should be low. Think about it! Hence, high values are preferable. Recall that the numerator measures the distance between clusters, i.e., sets of points. This means that all linkage functions (see Section 3) can be applied. Likewise the inter-cluster distances can be measured differently
Davies & Bouldin [DB79] in 1979 proposed the eponymous Davies-Bouldin index.

Definition 4.11 (Davies-Bouldin index)**.** For a clustering $C_1, \ldots, C_k$ the *Davies-Bouldin index* is defined as

$$D(C_1, \ldots, C_k) := \frac{1}{k} \sum_{i=1}^{k} \max_{1 \leq i \neq j \leq k} \left(\frac{\sigma_i + \sigma_j}{d(\mu_i, \mu_j)} \right)$$

where μ_i is the centroid of the i-th cluster and $\sigma_i = \frac{1}{|C_i|} \sum_{x \in C_i} d(x, \mu_i)$ denotes the average distance of the points in the respective cluster to its centroid.

The formula looks complex. So let's approach it from the insight out. The fraction within the parenthesis relates the sum of average distances of points to their cluster center for pairs of cluster. This is divided by the distances between the respective cluster center. In a good separation the latter, the denominator, should be as high as possible. In consequence, the fraction should be as close to zero as possible. Next, we take the maximum (i.e., we take worst-case perspective) an eventually average over all clusters. Put differently the measure can be interpreted as the average similarity of clusters to their most similar neighbouring cluster.
Finally, we discuss a very nice graphical measure with a great interpretation introduced by Rousseeuw [Rou87].

Definition 4.12 (Silhouette)**.** Let $C_1, \ldots, C_k$ be a clustering. For $x \in C_l, l =$

$1, \dots, k$ let

$$a(x) = \frac{1}{|C_l| - 1} \sum_{\substack{y \in C_l \\ y \neq x}} d(x, y)$$

be the *mean distance between x and all other points in x's cluster*. Let further for $x \in C_l$

$$b(x) = \min_{\substack{1 \leq i \leq k \\ i \neq l}} \frac{1}{|C_i|} \sum_{y \in C_i} d(x, y)$$

be the *smallest mean distance of x to all points in any other cluster*, i.e., the neighboring cluster. Then the *silhouette (value/width)* of $x \in C_l$ is defined as

$$[-1, 1] \ni s(x) = \begin{cases} \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}, & \text{if } |C_l| > 1 \\ 0, & \text{otherwise} \end{cases} = \begin{cases} 1 - \frac{a(x)}{b(x)}, & \text{if } a(x) < b(x) \\ 0 & \text{if } a(x) = b(x) \\ \frac{b(x)}{a(x)} - 1, & \text{if } a(x) > b(x) \end{cases}$$

This is quite a mouthful, isn't it? Let's identify the meaning by inspecting the ingredients $a(x)$ and $b(x)$ first. By the way: the silhouette is calculated point-wise, i.e., for each data point $x \in \mathcal{X}$ in contrast to indices like the Dunn index. Even though we can certainly aggregate, e.g., building the mean silhouette value, visualising the point-wise silhouette values turns out to be very helpful. So, for each point $x \in \mathcal{X}$, assigned to some cluster C_l, we calculate $a(x)$ as the average distance of x to all other points assigned to C_l (since x is fixed, we divide by $|C_l| - 1$ and not $|C_l|$). Conversely, $b(x)$ is the average distance of x to the points in the (on average) closest cluster. Intuitively, $b(x)$ should be higher than $a(x)$, right? The rationale is that we would not expect points of another cluster $C_k, k \neq l$ be closer to x than points in C_l. Therefore, if we compute $b(x) - a(x)$ a large positive value is an indicator for x being assigned to the right cluster while negative values rather point out that x is assigned to the wrong cluster. Eventually, Rousseeuw divides by the maximum of $a(x)$ and $b(x)$. Since $|b(x) - a(x)| \leq \max\{a(x), b(x)\}$, this step serves as a standardisation and leads to the silhouette values being bounded between the extreme values -1 and 1. Figure 4.20 shows a k-means clustering with $k = 4$ on smiley alongside the silhouette plot. The latter displays the silhouette values split by cluster in decreasing order. What do we see here? First of all almost all silhouette values are positive even though the clustering is obviously sub-optimal. However, note that all values being positive is no good decision rule on great cluster quality. Let us therefore glance and the nuances. The silhouette plot reveals that all points $x \in C_4$ have very high silhouette values consistently surpassing 0.75.

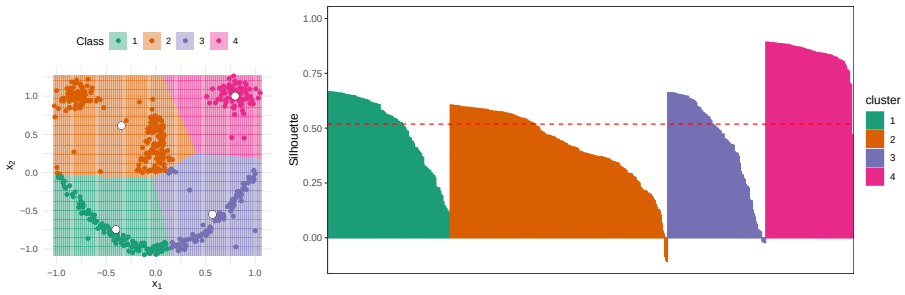


Figure 4.20: k-means clustering on the smiley data set (left) and respective silhouette plot (right).

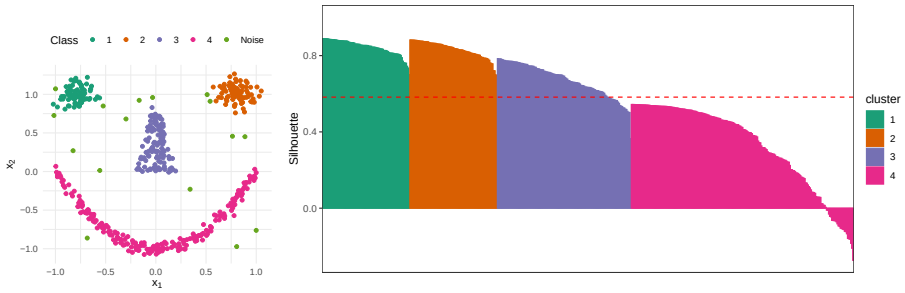


Figure 4.21: DBSCAN-clustering on the smiley data set (left) and respective silhouette plot (right).

For all other clusters we see many high values and a clear downward trend. This is perfectly in line with the clustering. Cluster C_4 is correctly identified while all other clusters are either split in two (the mouth) or merged (left eye and nose). Figure 4.22 plots the DBSCAN clustering ($\varepsilon = 0.01$ and `minPts`$= 4$) and the respective silhouette plot (without the consideration of the "noise cluster"). Here, the picture is different. Eyes and nose are identified correctly by the algorithm. Thus, the respective silhouette widths are consistently high. Even though the non-convex mouth is also identified correctly, the silhouette values are lower (all being below the average red dashed line) even with some negative values. How is this possible? Figure 4.22 shows the smiley data set with points coloured by their silhouette values. In par with our observations so far the convex shapes exhibit the highest silhouette values. In contrast, the silhouette values for the non-convex mouth cluster increases substantially if we move to the corners of the mouth finally turning negative. This makes absolute sense. Take the left-most point x in the mouth-cluster. Obviously, the closest different cluster is the left eye and it is easy to see that $a(x)$ will be lower than $b(x)$ since the points in x's cluster scatter much more in particular in x_1-direction. You see that the details matter. Nevertheless, comparing the silhouette plots in

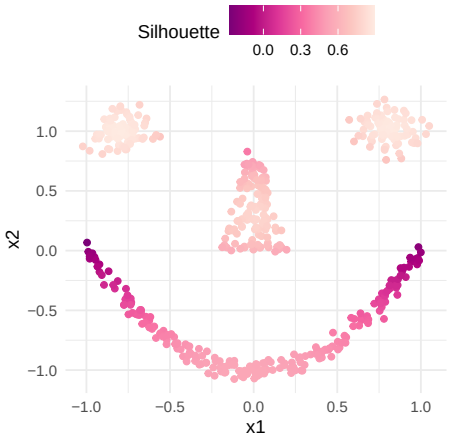


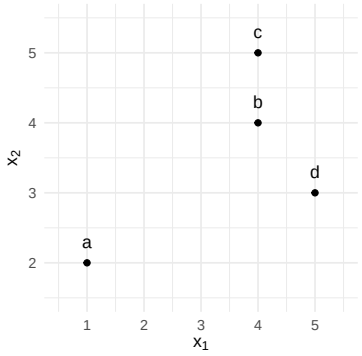
Figure 4.22: Smiley data coloured by silhouette values/widths of DBSCAN. The corners of the mouth tend towards lower (negative) silhouette values since the average distance to eye/nose get higher than the average distance to the cluster the points are assigned to (i.e., the mouth).

Figures 4.20 and 4.21, we can conclude that DBSCAN did a better job on this data set.

4.6 Exercises

Exercise 1 (Application: complete linkage)
Consider the four observations in the next figure and perform the following steps.

1. Use the Manhattan-Block distance (L_1-norm) to derive the respective distance matrix.
2. Apply agglomerative hierarchical clustering with complete linkage to derive a clustering. Which number of clusters seems adequate?



Exercise 2 (Hamming distance)
Consider observations $x_i \in \{0,1\}^p, i = 1,\dots,N$. I.e., for each variable the possible values are either 1 (yes/true) or 0 (no/false). This could be the answers of participants of a survey for yes/no answers (are you married? Do you have children? ...). A famous metric on such binary vectors is the *Hamming distance*

$$d_H : \{0,1\}^p \times \{0,1\}^p \to \{0,p\}, d_H(x,y) := |\{i \mid x_i \neq y_i\}|.$$

Describe what this definition actually means. Show that d_H is a distance function (see Definition 4.2).

Exercise 3 (Minkowski-metric ★)
The *Minkowski-metric* for two points $x, y \in \mathbb{R}^p$ and some parameter $\lambda > 0$ is defined as

$$d(x,y) = \sqrt[\lambda]{\sum_{i=1}^{p} |x_i - y_i|^\lambda} = \left(\sum_{i=1}^{p} |x_i - y_i|^\lambda\right)^{1/\lambda}.$$

1. What do we get if we set $\lambda = 1$ or $\lambda = 2$?
2. For two points $x, y \in \mathbb{R}$ visualize the Minkowski metric for different values of $\lambda \in \{1, 2, 2.5, 3\}$. I.e., draw a line-plot with the absolute difference $|x-y|$ on the x-axis and $d(x,y)$ on the y-axis. Describe your observations.

Exercise 4 (Similarity of sets)
The *Jaccard similarity* is a similarity measure on sets. For two sets A and B it is defined as

$$s_J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

1. For $A = \{1,4,7,2,6\}$ and $B = \{8,3,4,2,7,1\}$ calculate the Jaccard similarity.
2. Show that the Jaccard similarity satisfies all properties from Definition 4.3.

Exercise 5 (Quiz on k-means)

1. The within-cluster sum of squares values in the scree-plot are always monotonically decreasing.
2. The minimization of the WCSS is equivalent to the maximization of the TSS.

Exercise 6 (k-means cons)

In this exercise we are going to investigate another drawback of k-means. To this end we will generate some artificial data by generating data with two distinct spherical clusters. We consider $N = 200$ data points in $\mathbb{R}^2$ and the mean vectors $\mu_1 = c(5, 5)$ and $\mu_2 = c(10, 5)$. Now sample $n = 100, 90, 80, 70, \ldots, 10$ points from a $\mathcal{N}(\mu_1, \text{diag}(1, 1))$-distribution and the other $N - n$ points from $\mathcal{N}(\mu_2, \text{diag}(1, 1))$. Run k-means (R function `kmeans` from package `stats` with parameter `method="lloyd"` and multiple restarts) on each data set with $k = 2$ and plot the data coloured by the cluster assignment. Also add the cluster centres returned by k-means. Explain your observations and justify the results.

Exercise 7 (k-means implementation ★)
Implement the k-means clustering algorithm in R. For validation apply your implementation with $k = 2, 3$ on the `iris` data set and compare the results with the `kmeans` method implemented in the R package `stats`.

Exercise 8 (k-medioid implementation)
In k-means the cluster centres are usually no data points. A slight variation of k-means is the *k-medioid* algorithm. Here, the cluster center calculation is extended slightly. After calculating the center of mass, the cluster center is set to its closest data point. Modify your k-means implementation accordingly. I.e., add a Boolean/logical parameter `medioid` $\in$ {`TRUE`, `FALSE`}. If `FALSE`, the algorithm should fall-back to the usual k-means behaviour. If `TRUE`, the k-medioid approach should be used. Compare the results on the `iris` data set or some data set of your choice.

Exercise 9 (k-means: understanding the objective)
In k-means the objective is to find k clusters such that the within-cluster sum of squares (WCSS; see Eq. 4.2) is minimised. Show that this is equivalent to the maximisation of the *between-cluster sum of squares* (BCSS).

Exercise 10 (HCA fusion of clusters)
Argue that for agglomerative hierarchical clustering, if two single-element clusters fuse they do so at the same height regardless of the linkage function.

Exercise 11 (DBSCAN implementation ★)
Implement the DBSCAN algorithm in R. The function should return a named list with components `X` (the input data) and `cluster` (an integer vector of cluster assignment where noise points are encoded with zero entries). Verify empirically the correctness of your implementation by comparing the output with the established implementation of `dbscan::dbscan` on data sets of your choice.

Exercise 12 (DBSCAN application ★)
Consider the 2-dimensional **TODO** data set. Draw a scatter-plot of the data. How many clusters do you identify visually? Now consider the algorithms k-means and DBSCAN. Which algorithm is likely to correctly identify your ground truth and why / why not? Use R to run both algorithms on the data. Repeat the process multiple times for each algorithm toying around with different parameterisations (e.g., vary `minPts` and/or ε for DBSCAN). Visualise the clustering results and explain your observations.

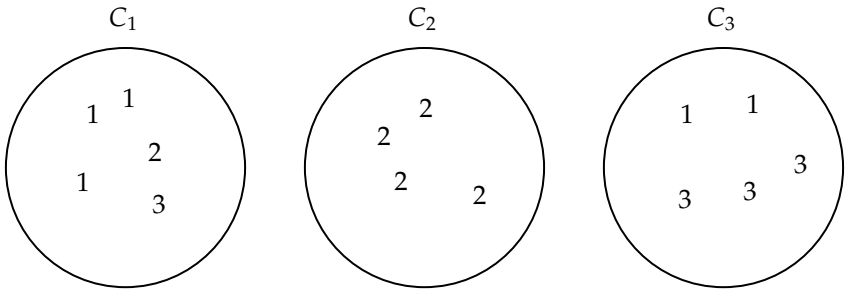
Exercise 13 (DBSCAN parameter "sensitivity" ★)
Consider the data set **TODO** and the clustering algorithm DBSCAN. Let n be the number of observations in the data. Let $d_{min} = \min_{x,y \in \mathcal{X}} d(x, y)$ and $d_{max} = \max_{x,y \in \mathcal{X}} d(x, y)$ be the minimum and maximum distance between data points in the data set. Consider an equidistant grid for the ε parameter. That is, let $\varepsilon = d_{min} + k \cdot (d_{max} - d_{min})/L$ for $L = 10$ and $k = 0, \ldots, L$. Moreover, set `minPts`$= 1, \ldots, n$. For each combination run DBSCAN on XY and log the number of clusters and the number of noise points. Visualise and discuss the results.TODO: does this make sense? Maybe implement the stuff, search for a fixed ε and let them vary `minPts` only?

Exercise 14 (Weirdo colleague)
A colleague of yours makes the following claim: For any data set $\mathcal{X}$ there is a value $\tilde{\varepsilon} \geq 0$ such that for any value of `minPts` ≥ 2 DBSCAN characterises all points $x \in \mathcal{X}$ as noise for all $\varepsilon \leq \tilde{\varepsilon}$. Is this claim true or false? Justify your answer.

Exercise 15 (k-means vs. DBSCAN on `DS3` ★)
Use the `DS3` data set from the `dbscan` R package (`data(DS3)`). Check the help package to get some information on the data. Apply k-means to obtain a clustering for different values of k. Afterwards, run DBSCAN experimenting with different values of ε and `minPts` and compare the results visually.

Exercise 16 (RAND index (exercise from DA1 exam in WT21/22))
The figure above shows three clustering results C_1, C_2, and C_3. The circles indicate which observations are assigned to the same cluster. The true labeling (i. e., the ground truth) is given by the respective numbers of the observations, e. g., all 1's belong to the same class. For all calculations provide a reasonable amount of steps and not just the answer.

1. Calculate the True Positives, False Positives, False Negatives, and True Negatives.



2. Calculate the RAND index.
3. You can add one observation (belonging to the class of your choosing) to any cluster. Which one would you choose, if you want to get the maximum increase in terms of RAND index. Justify your answer!

Literature

[Ran71] William M. Rand. "Objective Criteria for the Evaluation of Clustering Methods". In: *Journal of the American Statistical Association* 66.336 (1971), pp. 846–850. DOI: 10.1080/01621459.1971.10482356.

[Sib73] R. Sibson. "SLINK: An optimally efficient algorithm for the single-link cluster method". In: *The Computer Journal* 16.1 (Jan. 1973), pp. 30–34. DOI: 10.1093/comjnl/16.1.30.

[Dun74] J. C. Dunn. "Well-Separated Clusters and Optimal Fuzzy Partitions". In: *Journal of Cybernetics* 4.1 (1974), pp. 95–104. DOI: 10.1080/01969727408546059.

[Def77] D. Defays. "An efficient algorithm for a complete link method". In: *The Computer Journal* 20.4 (Jan. 1977), pp. 364–366. DOI: 10.1093/comjnl/20.4.364.

[DB79] David L. Davies and Donald W. Bouldin. "A Cluster Separation Measure". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2 (1979), pp. 224–227. DOI: 10.1109/TPAMI.1979.4766909.

[Rou87] Peter J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. DOI: https://doi.org/10.1016/0377-0427(87)90125-7.

[Est+96] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: AAAI Press, 1996, pp. 226–231.

[TGH01] Robert Tibshirani, Walther Guenther, and Trevor Hastie. “Estimating the Number of Clusters in a Data Set via the Gap Statistic”. In: *Journal of the Royal Statistical Society Series B* (2001).

[GT15] Junhao Gan and Yufei Tao. “DBSCAN Revisited: Mis-Claim, Un-Fixability, and Approximation”. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’15. New York, NY, USA: Association for Computing Machinery, 2015, 519–530. DOI: 10.1145/2723372.2737792.

[Sch+17] Erich Schubert et al. “DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN”. In: *ACM Trans. Database Syst.* 42.3 (July 2017). DOI: 10.1145/3068335.

Chapter 5

Dimensionality Reduction

Let us recall our general setup: we are given N observations (i.e., data points) each being a p-dimensional vector, that is a collection of p different values. We already learned how to visualise numerical data. For $p = 1$ histograms, density plots or box-plots are an adequate choice. In case $p = 2$ a scatter-plot is the visualisation method of choice which can also be extended to three dimensions. For $p = 4$ things already get complicated. We could draw two-dimensional scatter-plot and set the size and colour of points proportionally to the third and fourth dimension respectively. This is a valid approach and has certainly its benefits in certain analysis. Alternatively, a scatter-plot matrix is possible. However, you can certainly imagine where our journey heads to: for higher feature space dimension p there are

$$\binom{p}{2} = \frac{p(p+1)}{2} \approx p^2$$

pairs of features! For large p this approach is infeasible (for $p = 50$ we already need to inspect 1 275 scatter-plots) beside the obvious drawback that merely binary relationships can be identified. Another drawback of many features is that basically all multi-variate statistical analysis are more difficult both with respect to runtime and interpretation.

Here, so-called *dimensionality reduction* methods come into play. The goal is to find a mapping from the original p-dimensional feature space into a q-dimensional "surrogate" feature space with $q \ll p$ (i.e., (ideally) q is way lower than p). Ideally, these q "new" or "transformed" features will capture the essential structure(s) of the original quite well and allow for easier visualisation, interpretation and less complex machine learning tasks (e.g. subsequent clustering; see Section 4).

Let's collect some simple ideas. (1) We could simply select q random variables

from the whole set of p variables. You certainly agree that this is a quite bad idea, because the information value of the variables is ignored. How can we measure the information provided by a feature? One straight-forward way is to consider the variance: the higher the variance the more information the feature contributes. The lower extreme are constant features, i. e., features that do not vary at all (imagine all participants of a survey being males). Obviously, we can drop these features without any information loss. Hence, (2) we could sort the features in decreasing order of variance and select the first q. This approach is certainly better than (1). Its major shortcoming is that it does not consider relationships between variables, i.e., correlations. As a plain simple example assume that we are given three numerical features X_1, X_2, X_3 such that $X_2 = X_1$ and $X_3 = 1/4X_2 + 4$. I.e., features X_2 and X_3 are (deterministic) functions of feature X_1. As a consequence they do not contribute any additional value to the data. Nevertheless, $\text{Var}(X_1) \geq \text{Var}(X_2) \geq \text{Var}(X_3)$ and selecting two of these would neglect the variation. A third possibility is to combine all variables into a single one by averaging

$$Y = \frac{1}{p} \sum_{i=1}^{p} X_i.$$

This approach seems interesting, but the obvious drawback is that every single variable is weighted equally even though the respective information contributions will usually vary a lot (see above). Hence, it makes sense to introduce a weight vector $\gamma \in \mathbb{R}^p$ with $\gamma^T \cdot \gamma = 1$ and define

$$Y = \frac{1}{p} \sum_{i=1}^{p} \gamma_i \cdot X_i.$$

In this section we will learn about sophisticated dimensionality reduction methods. The first one, Principal Component Analysis (PCA), is a classical linear method from statistics which basically builds upon our weighted sum approach developed above. The second method is t-SNE, a very interesting non-linear approach which overcomes PCA's major shortcoming.

5.1 Principal Component Analysis (PCA)

The so-called *Principal Component Analysis* (PCA) is a sophisticated and often used statistical method for dimensionality reduction. The core idea is to replace the original p variables by p transformed uncorrelated variables – the so-called *principal components* (PCs) – which are ordered by the fraction of variance they

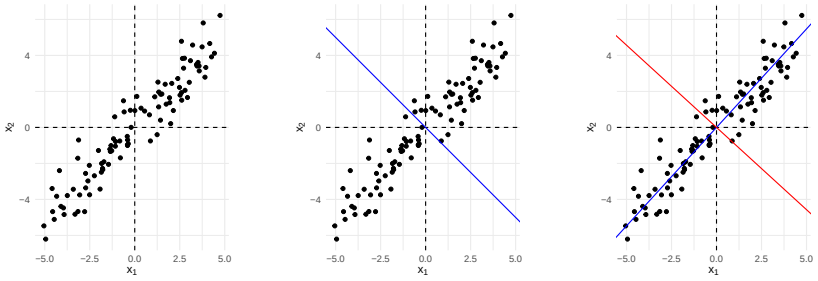


Figure 5.1: Example data (left) with bad (middle) and good (right) direction for the most variance explained with a blue line.

explain in the data (Wait!? Replace p variables with p variables does not reduce anything. We will see later that we select only a subset of those new features). The background here is that your real-world data is likely to share information throughout the variables. Our goal is to gather this common information in new variables where the variables explain the variation in the data in decreasing order of magnitude. Selecting the first $q \ll p$ (say 3 out of 43 PCs) of these PCs ideally explain the vast majority of variation within the data and vastly facilitates further multivariate analysis. In this section we will dive into the details of PCA, its calculation and geometric interpretation.

Let us start with some simple visualisation. Consider the leftmost plots in Figure 5.1. If the task would be to draw an arrow (a vector) starting in the origin $(0, 0)$ and pointing in the direction of the highest variance it would certainly point towards the top right corner or the bottom left corner; geometrically there is no difference. Hence, the blue line in the center plot does not explain the largest variation well while the blue line in the rightmost plot does. In a nutshell – leaving apart the details – PCA searches for these "principal" directions. I.e., we search for the direction with largest variation first (the first PC). Next, we search for the direction that explains the largest variation and is perpendicular/orthogonal to the first direction (the second PC) etc. The orthogonality assures that the new PC explains new variation. Figure 5.2 shows another way of how the PCs can be interpreted. The first direction/PC is selected such that the sum of orthogonal projections of the data points onto the PC is minimised.

5.1.1 Mathematical Formulation

Our starting point are the p original random variables stored in a random vector $X = (X_1, \ldots, X_p)^T$. Without loss of generality we assume that $E(X) = 0$ (i.e., $E(X_i) = 0$ for all $1 \leq i \leq p$), i.e., the data is centred. This is technically not

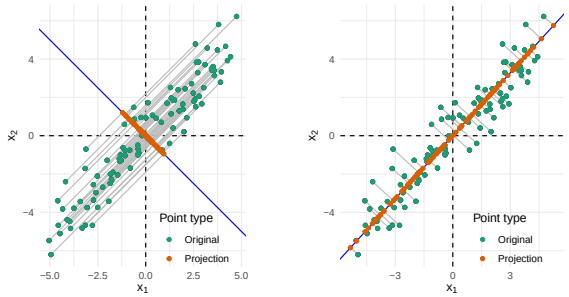


Figure 5.2: These plots show directions (blue lines) and orthogonal projections of the original data points (green dots) onto the directions (orange dots). Gray lines indicate the distance of the projection. PCA aims to minimise the sum of these orthogonal projection distances. Here, the direction in left plot certainly is no minimiser while the right one is.

necessary, but it makes the maths more accessible since

$$\text{Cov}(X) = E(XX^T) - \underbrace{E(X)E(X)^T}_{=0} = E(XX^T).$$

PCA generates a new set of p variables $Y_1, \ldots, Y_p$ where

$$Y_i = \sum_{j=1}^{p} \gamma_{ij} X_j = \gamma_{i1} X_1 + \gamma_{i2} X_2 + \ldots + \gamma_{ip} X_p. \tag{5.1}$$

I.e., the new variables are linear combinations of the original variables (cf. our weighted sum approach in the beginning of this chapter). In particular it holds that $\text{Cov}(Y_i, Y_j) = 0$ for $1 \leq i \neq j \leq p$; the PCs are uncorrelated in contrast to the original data that in general will have correlations between the variables. The construction of the PCs can be understood as an iterative process. The first PC $Y_1 = \sum_{j=1}^{p} \gamma_{1j} X_j$ is constructed such that

$$\max_{\gamma_1} \text{Var}(Y_1) \text{ s. t. } \gamma_1^T \cdot \gamma_1 \stackrel{!}{=} 1.$$

That means that the variance of the first PC should be maximised. The additional condition requires the weights $\gamma_1 = (\gamma_{11}, \ldots, \gamma_{1p})^T$ to be normalised. Why is this condition necessary? Well, without this condition we can increase the variance on and on by increasing/decreasing the weights in γ_1 without

bounds. For the 2nd, 3rd etc. components $Y_i = \sum_{j=1}^{p} \gamma_{ij} X_j, 2 \leq i \leq p$ we solve

$$\underbrace{\max_{\gamma_i} \text{Var}(Y_i)}_{\text{Maximise variation}} \quad \text{s.t.} \quad \underbrace{\gamma_i^T \cdot \gamma_i \stackrel{!}{=} 1}_{\text{Normalisation}} \quad \text{and} \quad \underbrace{\gamma_i^T \cdot \gamma_j = 0 \quad \forall j < i}_{\substack{\text{Require being uncorrelated} \\ \text{to all previous PCs}}}.$$

I.e., we maximise the variance under the condition that the weight vector γ_i is normalised and the additional constraints that Y_i is uncorrelated to all previous PCs $Y_j, 1 \leq j < i$. This last point is of utmost importance as it basically requires that the new PC explains "a new fraction" of the variation within the data.

5.1.2 Solving the Optimisation Problem

We will now proceed by solving the constrained optimisation problem

$$\max_{\gamma_1} \text{Var}(Y_1) \text{ s. t. } \gamma_1^T \cdot \gamma_1 \stackrel{!}{=} 1. \tag{5.2}$$

to obtain the first PC. To do this let us first reformulate the variance of the ith PC:

$$\begin{aligned}
\text{Var}(\gamma_i^T X) &= E((\gamma_i^T X)^2) \\
&= E((\gamma_i^T X)(\gamma_i^T X)) \\
&= E((\gamma_i^T X)(X^T \gamma_i)) \\
&= E(\gamma_i^T (X X^T) \gamma_i) \\
&= \gamma_i^T \underbrace{E(X X^T)}_{=\text{Cov}(X)} \gamma_i \\
&= \gamma_i^T \Sigma \gamma_i.
\end{aligned}$$

We see that the variance can be written as a function of the covariance matrix of X. With this we can state an equivalent formulation of Eq. 5.2 as

$$\max_{\gamma_1} \gamma_1^T \Sigma \gamma_1 \text{ s.t. } \gamma_1^T \cdot \gamma_1 \stackrel{!}{=} 1. \tag{5.3}$$

The problem is still constrained, but accessible to classical methods from calculus. Introducing a *Lagrange-multiplier* λ_1 we can transform the constrained optimization problem in Eq. 5.3 into

$$\max_{\gamma_1, \lambda_1} L(\gamma_1, \lambda_1) = \gamma_1^T \Sigma \gamma_1 - \lambda_1 (\gamma_1^T \gamma_1 - 1). \tag{5.4}$$

I. e., we encode the constraint into the objective function. The theory of Lagrange-multipliers assures that the optima with respect to γ_1 for both Eq. 5.3 and Eq. 5.4 are indeed the same. The partial derivatives of Eq. 5.4 are

$$(I) \quad \frac{\partial L}{\partial \gamma_1} = 2\Sigma\gamma_1 - 2\lambda_1\gamma_1 \stackrel{!}{=} 0 \qquad (II) \quad \frac{\partial L}{\partial \lambda_1} = \gamma_1^T\gamma_1 - 1 \stackrel{!}{=} 0.$$

Simple equivalence transformations yield

$$(I) \quad \Sigma\gamma_1 = \lambda_1\gamma_1 \qquad (II) \quad \gamma_1^T\gamma = 1.$$

This means that γ_1 is the (normalized) eigenvector to the largest eigenvalue λ_1 of the covariance matrix Σ (see Definition A.12). This is an astonishing result, isn't it? It basically links (or reduces) PCA to the problem of calculating eigenvalues and eigenvectors of the covariance matrix Σ. Let us briefly come back to the directions in the rightmost plot in Figure 5.1. Our result states that the first PC points in the direction of an eigenvector to the largest eigenvalue and that the latter describes the variation in this direction. Back to the mathematics we can now leverage the following essential result from linear algebra.

Theorem 5.1 (Eigenvalue decomposition (EVD)). *Every symmetric matrix $\Sigma \sim (p, p)$ can be decomposed into*

$$\Sigma = A \cdot D \cdot A^T$$

where

- *$D = diag(\lambda_1, \ldots, \lambda_p)$ contains the (non-negative) eigenvalues of Σ in descending order and*
- *$A \sim (p, p)$ is an orthogonal matrix[1] of the form*

$$A = [\gamma_1 \ldots \gamma_p] = \begin{bmatrix} \gamma_{11} & \gamma_{21} & \cdots & \gamma_{p1} \\ \gamma_{12} & \gamma_{22} & \cdots & \gamma_{p2} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{1p} & \gamma_{2p} & \cdots & \gamma_{pp} \end{bmatrix} \sim (p, p).$$

and contains the respective eigenvectors in the columns. I.e., the i-th column is γ_i.

[1] This means $A^TA = AA^T = I_p$, i.e., the column-vectors are pairwise orthogonal.

Given the EVD of Σ the principal components can be calculated via

$$Y = A^T X \Leftrightarrow \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_p \end{bmatrix} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1p} \\ \gamma_{21} & \gamma_{22} & \cdots & \gamma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{p1} & \gamma_{p2} & \cdots & \gamma_{pp} \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^p \gamma_{1i} X_i \\ \sum_{i=1}^p \gamma_{2i} X_i \\ \vdots \\ \sum_{i=1}^p \gamma_{pi} X_i \end{bmatrix}.$$

The PCs are indeed uncorrelated as the following calculations reveal:

$$\begin{aligned} \text{Cov}(Y) &= E[A^T X (A^T X)^T] && \text{(by definition)} \\ &= E[A^T X (X^T \underbrace{(A^T)^T}_{=A})] && \text{(properties of transposition)} \\ &= A^T \cdot \underbrace{E[X X^T]}_{=\Sigma} \cdot A && \text{(by linearity)} \\ &= A^T \Sigma A \\ &= D \\ &= \text{diag}(\lambda_1, \ldots, \lambda_p). \end{aligned}$$

Since D is a diagonal matrix we see that $\text{Cov}(Y_i, Y_j) = 0$ for $1 \leq i \neq j \leq p$ which proves the claim.

5.1.3 The Number of Principal Components

So far, we replaced p original variables with p principal components which – by their nature of being linear combinations – are more complex and harder to interpret than the original ones. Put differently, we did not solve the dimensionality reduction problem at all. Luckily, the PCs are uncorrelated and sorted by the variance explained, i. e.,

$$\text{Var}(Y_1) \geq \text{Var}(Y_2) \geq \text{Var}(Y_p).$$

In addition the total variance explained by <u>all</u> p PCs equals the total variance of the original data since

$$\sum_{i=1}^p \text{Var}(Y_i) = \text{tr}(D) = \text{tr}(A^T \cdot \Sigma \cdot A) = \text{tr}(\underbrace{A^T \cdot A}_{=I_p} \cdot \Sigma) = \text{tr}(\Sigma) = \sum_{i=1}^p \text{Var}(X_i).$$

Here we used (1) a nice property of the trace of a matrix product, namely that even though matrix multiplication is not commutative in general, we can rearrange within the trace operation, and (2) $A^T A = A A^T = I_p$ for an orthogonal

$(p \times p)$-matrix A where I_p is the $(p \times p)$-unit-matrix.
With these insights we can now naturally describe the *proportion of variance explained by the i-th PC* as

$$P_i = \frac{\text{Var}(Y_i)}{\sum_{j=1}^{p} \text{Var}(Y_j)} = \frac{\lambda_i}{\text{tr}(D)} \in [0, 1].$$

The total variance explained by the first $1 \leq k \leq p$ PCs is simply the cumulative sum of the P_i, namely

$$P^{(k)} = \sum_{i=1}^{k} P_i = \frac{\sum_{i=1}^{k} \lambda_i}{\text{tr}(D)} \in [0, 1].$$

So, starting with the first PC, the addition of any further PC will add a new fraction of variation. In practice, the goal is to select q such that at least 70% to 90% of the variation in the data is covered. Ideally, this can be achieved with $q \in \{2, 3\}$ components. Unfortunately, this is rarely the case (see Moore's Law). One simple possibility to decide on the number of PCs is to keep all PCs $j \in \{1, \ldots, p\}$ where $P_i > \frac{1}{p} \sum_{i=1}^{p} \lambda_i$. Another, more appealing approach is to draw a line-plot / bar-plot of q against the variation explained P_q or – even better – q versus $P^{(q)}$, the total variance explained. In this plot we search for an elbow / a knee in the plot very much alike the elbow-method to determine the optimal number of clusters for k-means (cf. Chapter 4).

5.1.4 PCA Example and Visualisation

We consider a data set on crime statistics in the USA. For each state ($N = 50$) it records $p = 7$ features (number of cases with respect to different crimes: murder, rape etc.). Figure 5.3 shows pairwise scatter-plots alongside correlation values. Take a moment and think of how well PCA would perform on this data set before you continue reading!
Obviously and unsurprisingly, there is a lot of redundancy in the data. Correlations are consistently in high positive regions. Recall that highly correlated data is an indicator for one variable being well explained through (a combination of multiple other) variable(s). Hence, PCA is likely to perform well in the sense that way less than seven PCs will be enough to represent the majority of variation in the data. In Figure 5.4 we see the resulting scree-plot. The first two PCs account for $\approx 78.6\%$ of the total variability; this is lot. With $q = 2$ we proceed by drawing a so-called *bi-plot* in Figure 5.5. A bi-plot shows two PCs (here the first two). Points represent the projections – so-called *scores* – of the original data into the transformed space. Vectors originating in $(0, 0)^T$

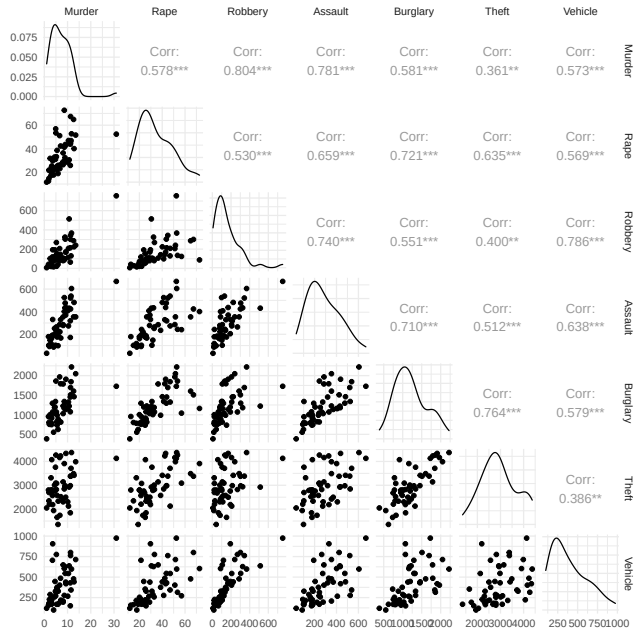


Figure 5.3: Pairwise scatter-plots for crime data. Correlations are very high. This is an indicator for presumably promising high potential of PCA for dimensionality reduction.

represent the weights λ_{ij} of the PCs (here $j = 1, 2$), the so-called *loadings*. How do we interpret scores, loadings and their interrelations? For the points/score the relative position is interesting. Points which are grouped close together in the bi-plot have similar scores for the displayed PCs. This means that they are similar with respect to the original variables. Likewise, distant points are rather dissimilar in the original space. The loadings can be interpreted with respect to three properties [Ros17]:

1. The *orientation* of the vectors to the PC-axis (ordinate and abscissa). The more parallel a vector is to an axis, the stronger its contribution to solely this principal components.
2. *Angles* between variables indicate correlations between original variables and can be interpreted as (dis)similar response pattern. A high positive correlation is indicated by small angles while negative correlations make their presence felt by opposite angles. (Almost) right angles emphasise low or no correlation. In our example we see many positive correlations (cf. pairwise scatter-plots in Figure 5.3). There is an almost right angle, e. g., between theft and murder which is supported by a relatively low correlation of 0.361. Contrary, murder and robbery have a very high correlation of 0.804 and indeed the angle between the loadings is minuscule.

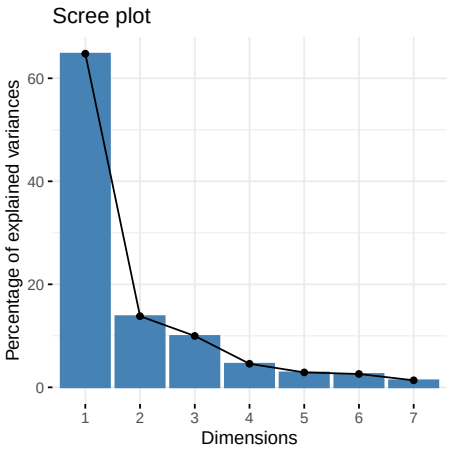


Figure 5.4: Scree-plot for the crime data. The first two components explain over 75% of the total variation in the data.

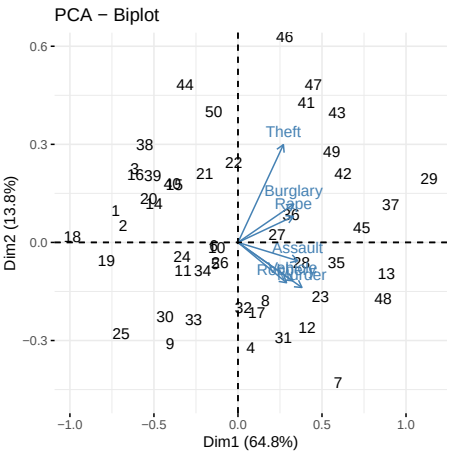


Figure 5.5: Bi-plot of the first two PCs for the crime data.

3. Finally, the *length* of the vector represents its importance/variability in the two displayed components. Variables with relatively short loadings are better explained by other PCs. In our crime example, theft contributes heavily to the displayed PCs, in particular to the 2nd PC (0.310 for the first PC and 0.697 for the second) while murder contributes way less to both components (0.379 for the 1st and -0.253 for the 2nd component, but 0.602 for the 3rd).

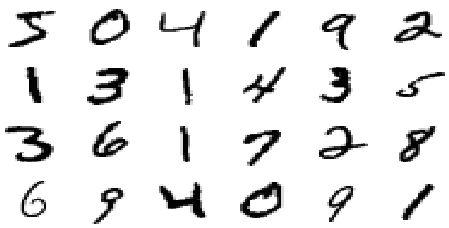


Figure 5.6: Example digits from MNIST data set [Lec+98].

5.1.5 Runtime ★

The calculation of the covariance matrix requires $O(p^2N)$ operations. The Eigenanalysis, i.e., the calculation of the eigenvalue decomposition of the covariance matrix Σ needs $O(p^3)$. This is actually very fast if p is small. If $p > N$ we can take advantage of the result that the matrices $XX^T \sim (p, p)$ and $X^TX \sim (N, N)$ share the same $\min\{p, N\}$ positive eigenvalues. Hence, for $p > N$ we need $O(N^2p)$ for the matrix product and $O(N^3)$ for the Eigenanalysis. Combining both bounds we get a runtime of $O(\min\{N^3, p^3\})$ given the covariance matrix is available. Note however, that the matrix product calculation can dominate if $N \gg p$ or $p >> N$ respectively. Take as an example $p = \Theta(N^2)$. In this case we can compute the EVD in time $O(N^3)$, but the calculation of XX^T takes time $O(N^4)$.

5.2 t-Distributed Stochastic Neighbor Embedding (t-SNE)

PCA is an established method. It has its drawbacks however. A major weakness of PCA is that it mainly aims to preserve the structure of dissimilar points, but firmly ignores similar points. This property is a direct consequence of the PCA calculations: the distance of dissimilar points is large and has therefore a huge impact on the variance (squared distances from the mean value) which is to be maximised in PCA. In contrast, the distance of similar points is small. This results in infinitesimal[2] contributions to the variance. Thus, PCAs' focus is on the global structure and not the local structure. In particular for large-scale data PCA might therefore perform poorly.

To illustrate this issue we consider the MNIST image data set [Lec+98].[3] It contains images (60 000 training and another 10 000 test examples) of handwritten digits (see Figure 5.6 for examples). Each image has a resolution of 28×28

[2] Infinitesimal means "extremely small".

[3] http://yann.lecun.com/exdb/mnist/

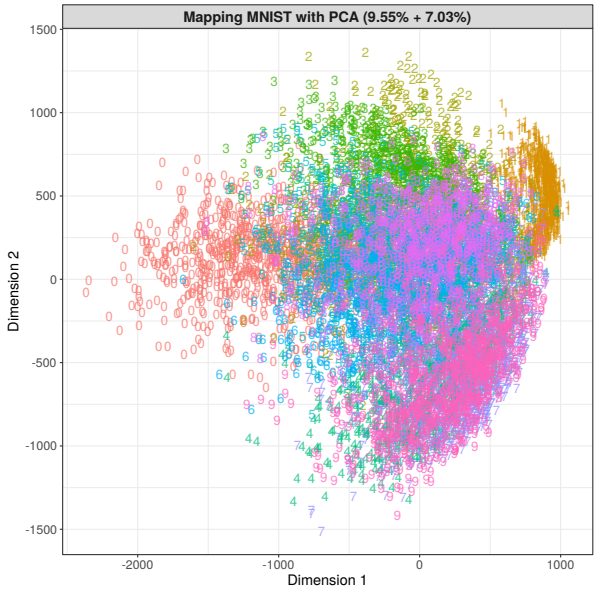


Figure 5.7: Scatter-plot of the first two principal components on a sub-set of the MNIST data set. While, e.g., zeros and ones are well separated, many letters strongly overlap.

pixels which results in a high-dimensional feature space of $p = 28^2 = 784$ features. The data is a standard benchmark for supervised learning since the actual labels (the true digits) are known for each image. Nevertheless it serves well to illustrate the different working principles of PCA and t-SNE. Figure 5.7 shows the a scatter-plot of the first two PCs coloured by the labels. Surprisingly, 16.58% of the total variation is encoded in only two dimensions. Nevertheless, even though written digits are obviously very distinct (e.g., one versus seven), there is a huge overlap in the first PCs as PCA is not able to capture the small local differences well. We will come back to this example later.

In this section we introduce t-SNE, short for **t**-distributed **S**tochastic **N**eighbor **E**mbedding, a non-linear dimensionality reduction method. t-SNE's focus is to model the large-scale high-dimensional space in two or three dimensions such that (1) similar points in the original space are mapped to nearby points in the mapped space (preserving local proximity) and likewise (2) dissimilar points are mapped to distant points (preserving global dissimilarities).

t-SNE was introduced by Van der Maaten and Hinton [MH08]. It is based on *Stochastic Neighbor Embedding* (SNE) proposed by Roweis and Hinton [HR03] in 2002 with some modifications that increase the method's robustness and computational complexity. For simplicity, we will not cover the differences and instead focus on t-SNE only. The interested student is referred to [MH08, p.3ff]. The starting point is: we are given a set of N observations $x_1, \ldots, x_N$ in our

original large-scale space where $x_i = (x_{i1}, \ldots, x_{ip})^T$ is a p-dimensional feature vector.

The key idea of t-SNE in a nutshell: first calculate the pairwise (Euclidean) distances $d(x_i, x_j) = ||x_i - x_j||^2, 1 \leq i, j \leq n$ between data points in the original space and derive a probability distribution over the respective pairs of points yielding probabilities p_{ij}.[4] Next, place target points $y_1, \ldots, y_n$ in the target space (usually $\mathbb{R}^2$ or $\mathbb{R}^3$) randomly and derive another probability distribution q_{ij} on basis of the points' respective distances in target space. The probabilities p_{ij} and q_{ij} can be interpreted as similarity values. Finally optimize the placement of the mapped points $y_1, \ldots y_n$ such that $p_{ij} \approx q_{ij}$ for all pairs (i, j), i.e., the mapping mimics the actual similarities in the low-dimensional space as good as possible. Actually the core idea is intriguing in its simplicity, isn't it?

Now let's dive into the details. p_{ij} can be interpreted as follows: it is the probability for data point x_i to select data point x_j as its neighbor (or the other way around) if neighbors were sampled randomly. Now (t-)SNE defines these probabilities as functions of the distances $d(x_i, x_j)$ between the points [HR03]. The rationale behind this is that nearby points should have a fairly high probability while points being located far off each other should be assigned a quite low probability. In any case, the probability should decrease with increasing distance in the original feature space. To this end we first define *conditional probabilities* $p_{j|i}$ with $p_{j|i} = 0$ for $i = j$ (zero probability to select the point itself as its neighbor) and for $i \neq j$ we set

$$p_{j|i} = \frac{\exp\left(-||x_i - x_j||^2/\sigma_i^2\right)}{\sum_{k=1,k\neq i}^{n} \exp\left(-||x_i - x_k||^2/\sigma_i^2\right)}.$$

Actually, the denominator – if Euclidean distance is used – corresponds to the density function of a Gaussian distribution centered at the mean value x_i. In particular the numerator approaches zero as $||x_i - x_j||^2 = d(x_i, x_j)$ approaches infinity (distance increases) which is desirable since the probability/similarity should decrease with growing distance in the original space. The denominator is the sum of all Gaussians and serves for standardization such that $p_{j|i}$ in fact defines a probability distribution. Convince yourself that in fact $\sum_{j=1}^{n} p_{j|i} = 1$ holds. In a last step Van der Maaten and Hinton define the *joint* probabilities

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} = p_{ji}.$$

This step actually assures that (i) the probabilities are symmetric ($p_{j|i} \neq p_{i|j}$ in general) and (ii) the probability to select a neighbor is at least $1/2N$ to avoid

[4] We will use Euclidean distance here, but t-SNE is perfectly fine with any distance measure.

arbitrarily low probabilities.

In the low-dimensional target space probabilities are defined slightly different:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)-1}{\sum_{1 \leq k \neq i \leq N}(1 + ||y_i - y_k||^2)^{-1}} = q_{ji}.$$

Here, the denominator again serves for standardization. The numerator follows a Student's t-distribution with one degree of freedom; this corresponds to a so-called Cauchy-distribution. Why don't we use Gaussian distributions again like we did for the original feature space? The reason is that the density function of a t-distribution is *heavy-tailed*. This means that the probability for large deviations from the mean value are much more likely than for Gaussian distribution. In fact, for a $\mathcal{N}(\mu, \sigma^2)$-distributed RV X we get $\Pr(|X| \geq 3\sigma) = 0.0027$. In contrast, for $X \sim t_1$ (t-distributed with one degree of freedom), the probability is $\Pr(|X| \geq 3\sigma) = 0.2048$, higher by orders of magnitude. This design choice is done to allow far apart points to be more likely far apart in target space.

Eventually, t-SNE aims to perturb $y_1, \ldots, y_N$ such that $p_{ij} \approx q_{ij}$ for all pairs $1 \leq i, j, \leq N$. To reduce these $2 \cdot \binom{N}{2}$ probabilities into a single measure of quality the *Kullback-Leibler divergence* (KLD) is adopted. KLD measures the "distance" between probability distributions and is defined via the following formula

$$C = \sum_{i=1}^{N} \sum_{j=1}^{N} p_{ij} \cdot \log\left(\frac{p_{ij}}{q_{ij}}\right) \tag{5.5}$$

which is to be minimized. If p_{ij} is very close to q_{ij} we get $p_{ij}/q_{ij} \approx 1$ and hence $\log(p_{ij}/q_{ij}) \approx 0$. I.e., the summand does not play a big role. If points are close in original space ($p_{ij} \approx 1$), but falsely distant in target space (q_{ij} close to zero) this is heavily penalized by KLB since $\log(p_{ij}/q_{ij}) > 1$. If the other way around holds (x_i and x_j far off, but y_i and y_j are placed close together) $\log(p_{ij}/q_{ij})$ will be large, but p_{ij} will be close to zero which results in a low contribution of $p_{ij} \cdot \log(p_{ij}/q_{ij})$ to the whole summation. Got it? ☺

t-SNE approximates the optimum via a simple gradient descent approach. To this end Algorithm 5 calculates the partial derivatives of the target function in Eq. 5.5

$$\frac{\partial C}{\partial y_i} = 4 \cdot \sum_{j=1}^{N} (y_i - y_j) \cdot (p_{ij} - q_{ij}) \cdot \left(1 + ||y_i - y_j||^2\right)^{-1}$$

and iteratively realigns the target points with respect to the steepest descent direction. For more details on the rationale behind this approach and the vari-

ous unexplained parameters the interested student is urged to take a glance at the original publications [MH08; HR03]. Note that gradient descent is deterministic. Nevertheless, Algorithm 5 is of stochastic nature due to the random placement of the initial points. Consequently, results over multiple runs may vary and thus the algorithm should be re-run multiple times.

Algorithm 5 t-Distributed Stochastic Neighbor Embedding (t-SNE)

Require: Data set $\mathcal{X} = \{x_1, \ldots, x_n\}$, $x_i \in \mathbb{R}^p$, target dimension q, perplexity parameter σ, max. number of iterations $T_{\max}$, learning rate η and momentum $\alpha(\cdot)$

1: Compute all pairwise distances $d(x_i, x_j), 1 \leq i, j \leq N$
2: Calculate joint probabilities p_{ij}
3: Sample initial targets $\mathcal{Y}^{(0)} = \{y_1^{(0)}, \ldots, y_N^{(0)}\}$ using $\mathcal{N}(0, 10^{-4} \cdot I_q)$
4: **for** $t \leftarrow 1$ to $T_{\max}$ **do**
5: Compute $d(y_i, y_j)$ and q_{ij}
6: Calculate gradients $\frac{\partial C}{\partial y_i}$
7: Set $y_i^{(t)} = y_i^{(t-1)} + \eta \cdot \frac{\partial C}{\partial y_i} + \alpha(t) \cdot \left(y_i^{(t-1)} - y_i^{(t-2)}\right)$
8: **end for**
9: **return** $\mathcal{Y}^{(T_{\max})}$

Figure 5.8 shows the result of t-SNE applied to the MNIST data with the goal to represent the data in just two dimensions. The results are downright impressive. Coloring the points with the true class labels which t-SNE as an unsupervised learning method did not use at all shows that the 2D-embedding in fact is capable of discriminating all 10 classes (digits zero, one, ..., nine) very well; we can clearly identify ten widely disjoint clusters (for comparison, look at the PCA result in Figure 5.7 again). There is a handful of "false cluster assignments". However, this can be explained in multiple ways. (1) t-SNE as a heuristic may be trapped in a local optimum (even if re-run multiple times). (2) occasionally it may be really hard – even for humans – to distinguish poorly hand-written nines and fours or eights and sixes. In fact, this can be seen in the plot. Eights and nines are among the most often wrongly assigned digits. Besides such artifacts, a modified version of Figure 5.8 shows even more impressive results if the respective images are drawn at the coordinates of the embedded points (see the Google Techtalk by Geoffrey Hinton[5]). For instance, the modified plot reveals that the cluster for the digit seven is further subdivided into different variants of handwritten seven: the usual seven as it is used, e.g., in the US, and a 7 with an additional line in the middle (sometimes called *continental seven*). The latter is done to make it easier to distinguish seven and one for certain handwriting styles. This small detail illustrates the ability

[5] http://lvdmaaten.github.io/tsne/

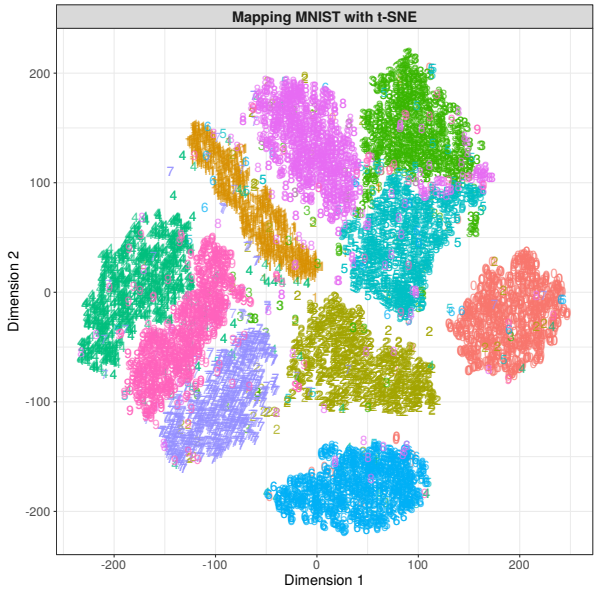


Figure 5.8: Two-dimensional embedding produced by t-SNE on a subset of the MNIST dataset. The algorithm – even though being an unsupervised method (i.e., t-SNE does not use any knowledge on the true class labels) – does a remarkable job in separating the ten classes in 784-dimensional original space in just two dimensions.

of the procedure to detect even the smallest similarities.

5.3 Exercises

Exercise 1 (PCA: scaled data and correlation matrix)
Let $x_i, y_i, i = 1, \ldots, N$ be samples of two random variables X, Y. The sample covariance and correlation is given by

$$s_{xy} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x}) \cdot (y_i - \bar{y}) \text{ and } r_{xy} = \frac{s_{xy}}{s_x \cdot s_y}$$

where s_x and s_y is the respective standard deviations.

1. Show that for linear transformations $x_i' = a \cdot x_i + b$ and $y_i' = c \cdot y_i + d$ with $a, b, c, d \in \mathbb{R}$ it holds that $s_{x'y'} = a \cdot c \cdot s_{xy}$.
2. Show that for $x_i' = \frac{\bar{x}}{s_x}$ and $y_i' = \frac{\bar{y}}{s_y}$ it holds that $s_{x'y'} = r_{xy}$.
3. From the above draw conclusions for the relationship of PCA with scaled data and the correlation matrix.

Exercise 2 (t-SNE joint probabilities)

Show that for the joint probabilities $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2N}$ defined in t-SNE it holds that (a) $p_{ii} = 0$, (b) $p_{ij} = p_{ji}$ and (c) $\sum_{i,j} p_{ij} = 1$. Do all these properties also hold true for the conditional probabilities $p_{j|i}$?

Exercise 3 (PCA and t-SNE on MNIST)
Reproduce the dimensionality reduction conducted in this section for three target dimensions. To this end:

1. Download the `MNIST` data set (see package `snedata`).
2. Select 6 000 random data points.
3. Perform a PCA with the data using `prcomp`. Visualize the first three PCs in a 3D scatter-plot (use either `scatterplot3d`, `plot3D` or – for interactive plotting – `plotly`). How much variance is explained by the first three PCs? How many PCs would be necessary to consider if one wants to explain at least 70% of the data?
4. Run t-SNE on the data with three target dimensions. Compare the results with PCA.
5. Now we want to compare the computational time required to perform PCA and t-SNE. Consider random subsets of $n \in \{5\,000, 10\,000, 15\,000$ and $20\,000\}$ data points from `MNIST`. For each n calculate the PCA and t-SNE embeddings (with default parameters) each 10 times independently to account for stochasticity and the varying workload of your machine. Measure the time (in seconds) for each run and visualize the running time distributions, e.g., via box-plots. Describe your observations.

Literature

[Lec+98] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

Chapter 6

Working with Missing Data

Unfortunately data is often *incomplete*. I. e., in contrast to complete observations a part of the features is *missing*. R has the reserved special value `NA` for not *available* observations. We will use both terms here interchangeably. What are potential reasons for missing data? Well, there are several reasons. In (sociological) studies data is often acquired by representative samples of people filling out questionnaires. Here, the questionnaire may require too much time. In such cases participants often submit the survey incomplete in the sense that they omit the last questions because they no longer feel like it or are annoyed. In addition, participants may get offended by questions(s). Imagine questions regarding sexual preferences, body-weight, eating habits etc. Such questions then are left unanswered; in the worst-case participants are so offended that they also skip any subsequent questions (even the non-offending ones). As another example imagine yourself jumping on the self-optimisation train. You would certainly start to track your body-weight and/or muscle circumferences on a regular basis, say once a week every Sunday morning. This way you could nicely visualise muscle gain or fat loss over time by plotting the time series data. Missingness occurs if you forget to track the respective values occasionally. Finally, in computational experiments missing data may come up if certain algorithms fail on certain problems and output nothing, aka `NA`, if no further information is being logged.

6.1 Missing Data Model

Rubin [Rub87] defined a formal model of missing data. To this end for each individual observation let X_{obs} be the observed part of the data and likewise X_{mis} the missing part. Let R be an indicator random variable which takes the value 0 if data is missing and 1 otherwise. Rubin's missing data model

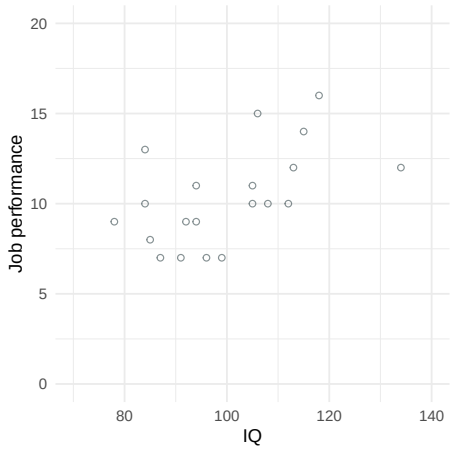


Figure 6.1: Complete hypothetical employee selection data set. Example taken from Enders [End10].

considers the probability distribution of the random variable R. He basically distinguishes three categories of missingness. For illustration of the latter we consider a small data set from a hypothetical employee selection scenario borrowed from Enders [End10]. The scenario is as follows: a company wants to hire. To this end prospective employees complete an IQ test. The company then hires applicants from the upper half of the IQ distribution. After a 6-month trial period a supervisor rates the job-performance (JP) on an integer scale. Figure 6.1 shows the hypothetical complete data. In the following we will see different modifications of the data with occasional missing JP values.

6.1.1 Missing Completely at Random

The first of Rubin's three categories is *Missing Completely at Random* (MCAR). As the name suggests here, the probability $\Pr(R = 0 \mid \theta)$ does not depend on the observed data X_{obs} or the missing variables X_{mis} at all. In the employee scenario this means that missing values for JP neither depend on JP itself nor IQ; they are completely random. What remains is to clarify what the parameter θ means. θ stands representative for distribution parameters. E. g., we could model R as being Bernoulli-distributed[1] with "success" probability θ, i. e., $\Pr(R = 0 \mid \theta) = \theta$. Figure 6.2 shows the hypothetical data again. Red crosses represent missing JP values under MCAR.

[1] A Bernoulli trial is a sequence of experiments with binary outcome which can be interpreted as *success* and *failure*. A Bernoulli RV models a single such experiment.

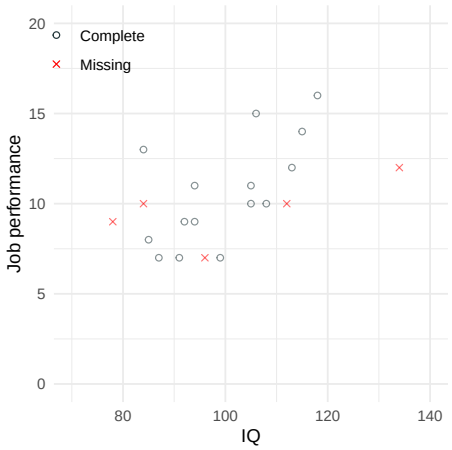


Figure 6.2: Employee selection data under MCAR. The missing job performance is independent of the observed and missing data.

6.1.2 Missing at Random

We can augment the probability distribution by additionally conditioning on the observed data X_{obs}: $\Pr(R = 0 \mid X_{\text{obs}}, \theta)$. Let's again take our example. Figure 6.3 shows the respective scatter-plot with missing JP-scores marked by red crosses. Obviously, the missing data is related to the IQ variable (which is observed in all cases) solely, but not to the missing data itself. This would make much sense in this scenario since only applicants from the upper half of the IQ distribution were hired. A logical consequence is that we cannot rate the job performance of non-hired applicants.

6.1.3 Missing not at Random

Finally, the missingness may be both due to X_{obs}, X_{mis} and further unobserved data: $\Pr(R = 0 \mid X_{\text{obs}}, X_{\text{mis}}, \theta)$. What does this mean? Take a glance at Figure 6.4. Here, five JP scores are missing in the lower end of the JP distribution; the missingness hence depends on the performance itself. These employees might have been fired prior to the evaluation time or may have left for different reasons (found a better job; got severely sick). However, we do not know since the data does not contain any information on the job status.
How do we know in applied settings which category missing data belongs to? Well, actually we do not know and most often cannot know. We could make an educated guess like we did in the MNAR example. Nevertheless there is no definite way and it gets particularly hard if missingness is a combination of multiple categories.
So why did we put so much effort into the theory? The theory supports the anal-

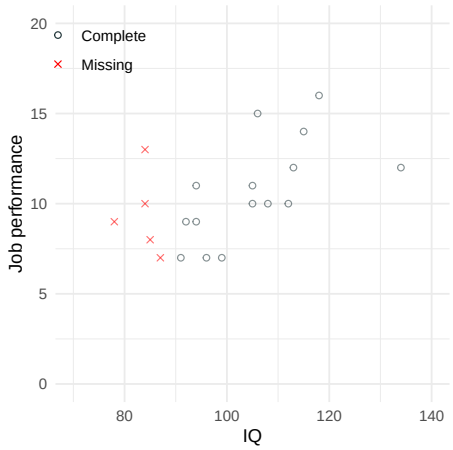


Figure 6.3: Employee selection data under MAR. Missing job performance values depend on the the observed values of the IQ feature:low IQ candidates were not hired.

ysis and aids the development of method suitable to deal with missingness. In the remainder of this chapter we will learn about different methods designed to tackle the problem of missing data, their properties and missingness conditions they rely on to perform reasonably well.

6.2 Deletion Methods

The solution to the missing data problem seems obvious: just delete all records where at least one feature is missing and work with the complete cases only. This approach is known as *complete case analysis* or *list deletion*. It is intriguingly simple and has the benefit that a common set of cases is used for all subsequent analysis. In fact this is the de facto standard in most statistical analysis tools. In R, the common functions for calculating summary statistics (e. g., `mean()`, `sd` etc.) expose a logical parameter `na.rm`. If set to `TRUE` – luckily not the default – missing data will be deleted prior to calculations. However, the method has at least two severe drawbacks. The first one is quite obvious: the method is extremely wasteful in particular if much data is missing. Assume we have p variables and for each variable MCAR holds with a probability of a missing value being $q \in (0, 1)$. Then, due to independence, the probability for each observation to have no missing values at all is $(1 - q)^p$ and thus the probability for at least one missing feature per observations is $1 - (1 - q)^p$. For $p = 3$ and $q = 0.05$ (very low) this is already 0.1426. I. e., about 14% of all observations would be dropped by complete case analysis. For $p = 3$ and $q = 0.2$ the value grows to 48.86%. Keeping $q = 0.05$ and increasing the number of features to

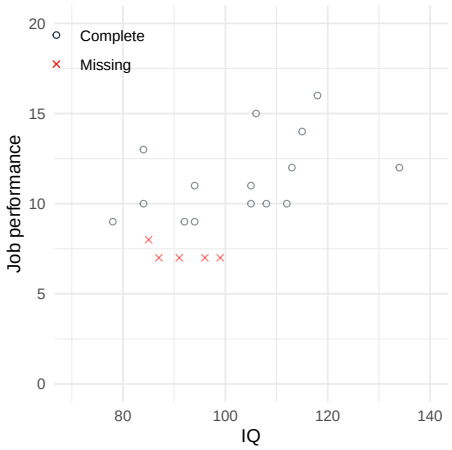


Figure 6.4: Employee selection data under MNAR. The missing job performance depends on data not recorded at all (likely the employment status).

$p = 20$ results in neglecting 64.15% of the observations. The second problem is related to the quality of estimations. Biased estimates will be the rule rather than the exception if MCAR does not hold. Figure 6.5 shows the employee setting data under MCAR and MAR. The black vertical lines represent the mean IQ if all the available IQ-scores where used. Red vertical lines show the mean value calculated on the complete cases only. Under MAR we see a massive shift towards higher IQ values. The reason is that we dropped all IQ-scores of applicants with missing job performance. The systematic cut-off of low IQ observations leads to the complete cases not being a representative sample of all applicants anymore. Such systematic distortions are very likely under MAR (or MNAR) if list deletion is adopted.

There seems to be an easy way out: perform the dropping process on an analysis-by-analysis basis. This is called *available-case analysis* or *pairwise deletion*. Here, for each statistical analysis we use all available data. E. g., to calculate the mean IQ we use all available IQ-scores (even those of observations with missing job performance). To calculate the mean JP, we would drop all observations with missing JP. For a multi-variate clustering based on IQ and JP we would need to drop all observations with missing JP values; i. e., available-case analysis would correspond to complete-case analysis for this type of investigation. This method is less wasteful, but nevertheless requires the missingness to adhere to MCAR. In addition, available-case analysis may introduce non-sense results. The standard example here is the calculation of correlations with the familiar

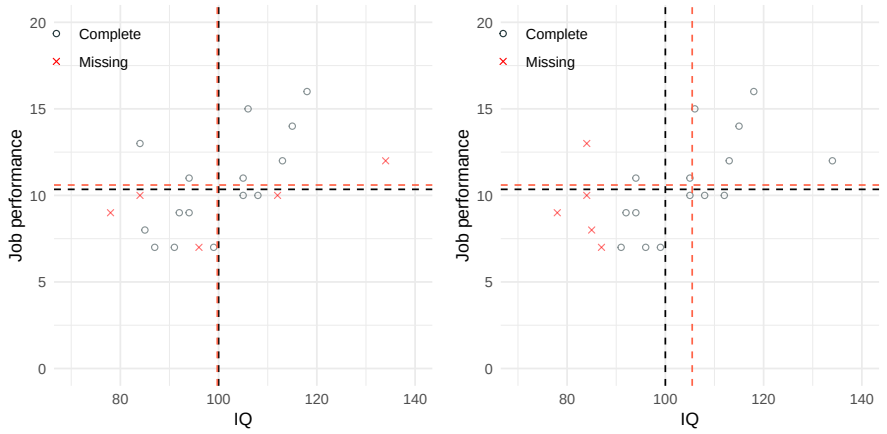


Figure 6.5: Employee selection data. Left: average mean split based on all cases and complete cases only for both features under MCAR. Right: the same estimates under MAR. With MCAR as the underlying missingness category, the IQ estimate is heavily biased towards higher values.

formula

$$\frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{N \cdot \sqrt{{s_x}^2 \cdot {s_y}^2}}.$$

If the respective mean values and standard deviations are all calculated on a different subset of the data – the available cases – correlations values outside $[-1, 1]$ are perfectly possible. Even the variance, i. e., the correlation of feature with itself, can turn to be not one.

In conclusion, deletion techniques are plain simple and thus tempting. The disadvantages (waste of data, reduction of statistical power, biased estimates if MCAR does not hold) by far surpass the benefits. These methods should thus be used with caution only if the amount of missing data is very very low. We close this section with a quote from the Wilkinson & Task Force on Statistical Inference, 1999, p. 598: "The two popular methods for dealing with missing data that are found in basic statistical packages – list-wise and pairwise deletion of missing values – are among the worst methods available for practical applications".

6.3 Imputation

If deletion is not an option, we need to replace the missing values somehow. This approach is called *imputation* which – according to the Cambridge dictionary[2]

[2] https://www.dictionary.cambridge.org

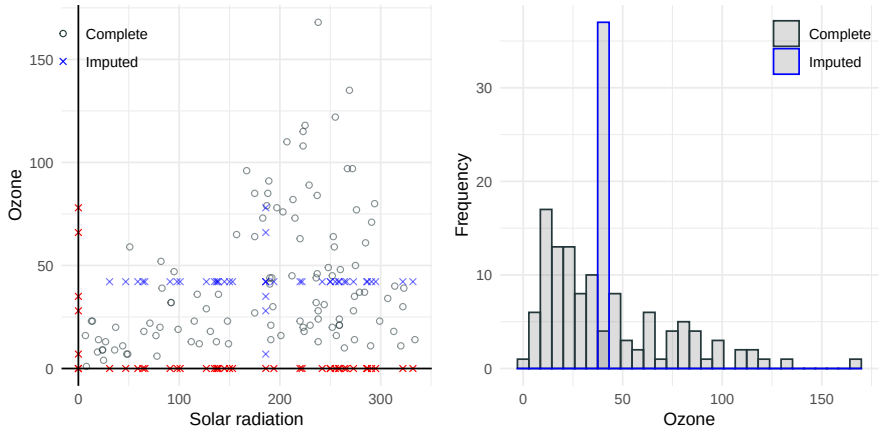


Figure 6.6: Results on airquality with mean imputation. Left: scatter-plot of solar versus ozone with point shapes indicating the type. Right: distribution of ozone coloured by set membership (complete or imputed).

– is "a way of calculating something when you do not have the full or correct data". This may seem weird at first sight. How can we replace values if they are not reported? Random guess? No. Actually, the core idea is to use the information of the available/observed data of the variable and/or other variables to find adequate fill-ins. In this section we will learn about multiple methods with increasing level of sophistication.

For illustration we will employ the small size real-world data set `airquality` on daily air quality measurements in New York, USA, from May to September 1973 [Cha+83].[3]. The data set consists of measurements of mean ozone (feature `Ozone` measured in parts per billion; ppb) with 35 missing values and solar radiation (feature `Solor.R`) with 5 missing values. Additionally, the data reports wind and temperature which are not of interest here.

6.3.1 Mean Imputation

The idea of *mean imputation* is to replace the missing value for a feature with the arithmetic mean of the available cases. Thus, in some sense imputation methods build upon complete/available-case analysis. This is done for each feature independently. Figure 6.6 shows a scatter-plot of solar radiation versus ozone. Black circles represent the complete cases while red crosses represent observations with at least one missing value. I. e., for all red crosses on the ordinate/abscissa the ozone / solar radiation value is missing. For the point in the origin both values are not available. Blue crosses represent the imputed observations, i. e., missing values replaced by the feature-wise mean value.

[3] Available in R via `data(airquality)`.

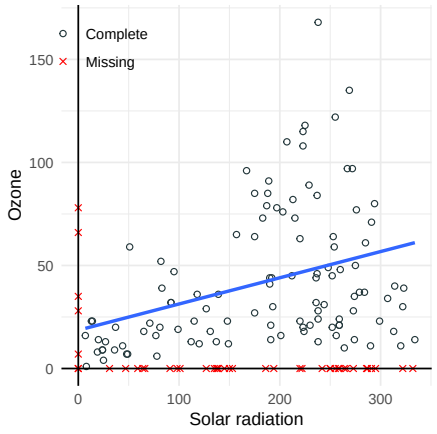


Figure 6.7: Airquality data with the liner regression model $\texttt{Ozone} = \beta_0 + \beta_1 \cdot \texttt{Solar.R} + \varepsilon$.

The right plot is a histogram of the ozone values coloured by complete cases and imputed cases.[4] Unsurprisingly, the histogram of the imputed ozone data has a single peak and no variation at all since we replaced the missing values with the constant mean value. Note that this colour encoding here is used for illustration purposes. In applications we would use the observed and imputed data without distinction (treat the imputed values as real data). The additional peak in the ozone distribution distorts the original data distribution heavily. Another drawback is that the standard deviation of the imputed data is reduced. Why is this? Note that in the formula for standard deviation we average the squared distances of values from the mean value. However, for the subset of imputed data the data values are equal to the mean value and thus contribute zero to the summation; since there are more observations considered now (observed plus imputed), the denominator grows. Another issue is that results are obviously very unconvincing if there are strong correlations or general functional dependencies in the data.

6.3.2 Regression Imputation

The problem of ignored correlations is tackled by *regression imputation*. Assume that X_i is a variable with missing values. Then regression imputation trains a

[4] The histogram(s) for solar radiation would show similar patterns. However, due to the low number of just 5 missing values for solar radiation the latter is less suited for demonstration purpose.

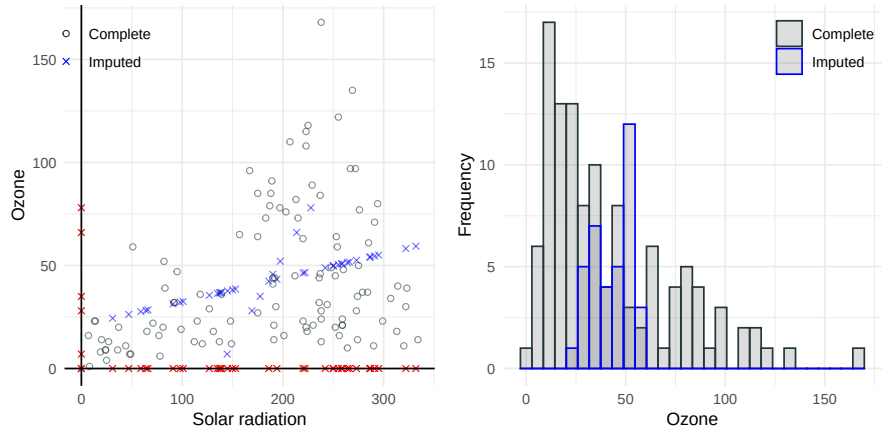


Figure 6.8: Results on airquality with regression imputation. Left: scatter-plot of solar versus ozone with point shapes indicating the type. Right: distribution of ozone coloured by set membership (complete or imputed).

(linear) regression model

$$X_i = \sum_{\substack{j=0 \\ j \neq i}}^{p} \beta_j \cdot X_j + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

with X_i being the target variable and the remaining variables serve as predictors. The training is performed on complete cases only. The idea is to use the predictions of the model for imputation. Figure 6.7 shows the least squares estimate for the regression $\texttt{Ozone} = \beta_0 + \beta_1 \cdot \texttt{Solar.R} + \varepsilon$. In order to obtain surrogates for missing ozone values, we plug in the predictor solar radiation into the formula and take the predicted values for imputation. Figure 6.8 shows imputed values for both ozone and solar radiation (i. e., two linear models trained). This approach seems interesting as it uses information encoded in other variables to obtain replacements. In fact, if the model is very accurate, i. e., the linear function models the data well (put differently if the data is strongly correlated), realistic imputation values are possible. Nevertheless, the method inherits most shortcomings of mean imputation.

6.3.3 Stochastic Regression Imputation

This method builds upon regression imputation. Basically, it does exactly the same. However, after having obtained the model predictions additional random noise with respect to the error distribution of the linear model is added. Thus, the method is called *Stochastic Regression Imputation*. This step adds a portion of stochasticity on top and makes the replacement values less "artificial".

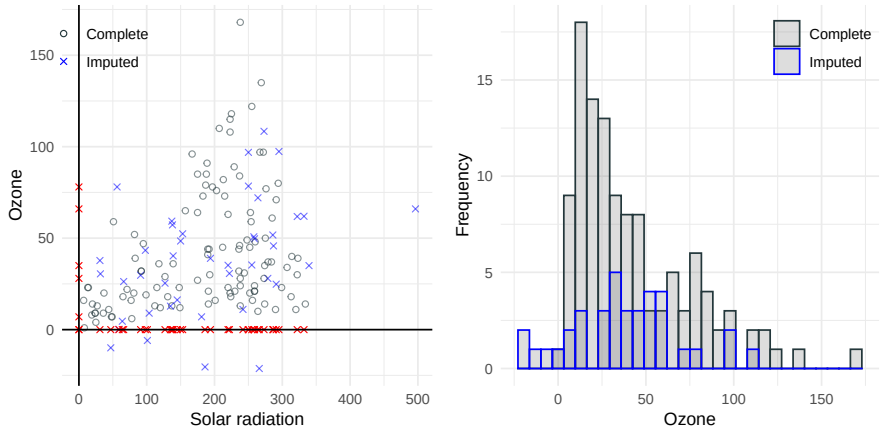


Figure 6.9: Results on airquality with stochastic regression imputation. Left: scatterplot of solar versus ozone with point shapes indicating the type. Right: distribution of ozone coloured by set membership (complete or imputed).

Figure 6.9 shows the method by example. The imputed values are scattered much more naturally in the space in direct comparison to RI in Figure 6.8. Also the ozone distribution appears to match the complete case distribution much better. However, there is one severe issue! A couple of imputed ozone values are negative. This does not make sense at all! This is major weakness of SRI. The random noise may introduce infeasible values in many applications domains. Nevertheless, the method serves as a basis for more sophisticated approaches.

6.3.4 Predictive Mean Matching

The method builds upon regression imputation, but avoids the problem of random noise of SRI by selecting true observed values for imputation. The method works as follows: first, the regression model is trained. Next, the prediction for the missing value is obtained. Furthermore, for each complete observation also predictions are calculated. Given an integer parameter $k \geq 1$ the method then calculates the k nearest predictions with respect to the missing feature. The fill-in value is the true observed value of a random *donor* from these k neighbours. Note that the distance is based on predictions only while the imputed value corresponds to the true observed value. It is done this way to ensure, that the range of the data is covered well. There exist various modifications of this method [SB08; ST96]. Since true observations are used for imputation the method produces realistic and plausible replacement values. As a side effect the method also works for integer features and even in the discrete case (given a suitable distance measure). In Figure 6.10 the PMM predictions are visualised. No infeasible values are produced here and the scattering of the

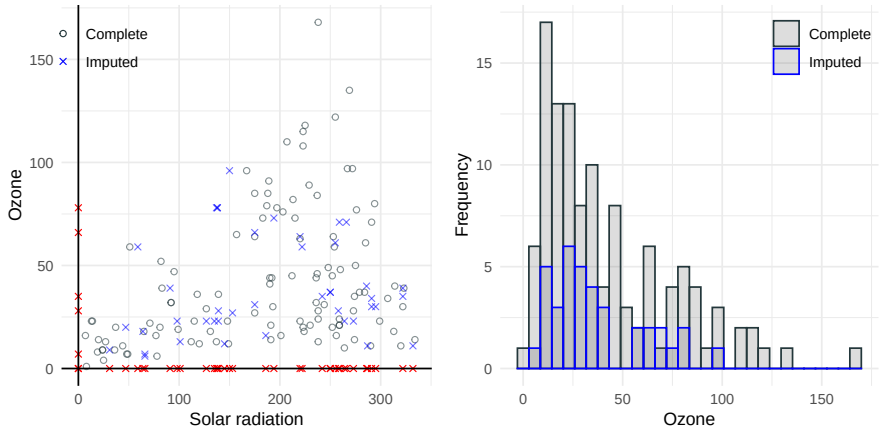


Figure 6.10: Results on airquality with predictive mean matching imputation. Left: scatter-plot of solar versus ozone with point shapes indicating the type. Right: distribution of ozone coloured by set membership (complete or imputed).

imputed values with respect to ozone very much mimics the complete cases. This observation is supported by the ozone density in the histogram. Finally, the distribution of the imputed values and the complete case is very much alike with respect to shape and characteristics. PMM also tackles another drawback of SRI which is that the latter performs poorly if the imputed feature is heteroscedastic (varying variance). Figure 6.11 illustrates this issue with an artificial data set. Due to sampling in SRI relying on the standard error of the regression model,[5] the narrow nature of the data for low x values is mimicked badly and so is the wide spread for large x. In contrast PMM produces imputations that look quite convincing. So, are we done now? PMM seems to be an incredibly well-designed method. The answer is no. PMM also has some shortcomings. It performs badly on small data sets and cannot extrapolate beyond the observed data range.[6] In addition, alongside a distance measure (we assume Euclidean distance here) the parameter k needs to be specified; low values increase the probability to choose the same donor repeatedly.[7] Nevertheless it is one of the most recommended methods in combination with multiple imputation (see next section).

[5]The standard error in regression analysis is the mean distance of the observations to their projections on the regression function.

[6]It is questionable though whether the infeasible extrapolation is much of a problem since extrapolation in general is tough.

[7]In fact the default is $k = 1$ in SAS and SPSS. This does not make sense if multiple imputation is adopted as it makes the method deterministic.

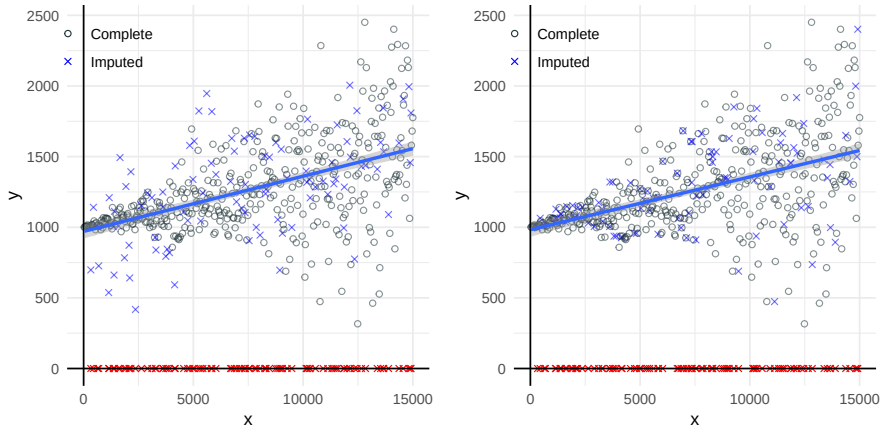


Figure 6.11: Exemplary result of SRI (left) and PMM (right) on artificial heteroscedastic data. PMM outperforms SRI clearly by generating imputation values that are much more plausible.

6.4 Multiple Imputation

The stochastic nature of the two best classical imputation methods SRI and PMM gives rise to yet another source of variation: imputing values with these methods multiple times will yield different estimates of subsequently calculated statistical measures. Since the analysis part treats imputed values as real data, one iteration of classic imputation will usually underestimate standard errors [End10, p. 222]. In order to account for this additional uncertainty *multiple imputation* is the gold-standard for missing data. Here, given a basic stochastic imputation method, the process is three-fold (see also Figure 6.12)

1. **Imputation**: Generate $m > 1$ copies of the original data and apply the imputation method of choice independently to each copy. This yields imputed data sets $X_1, \ldots, X_m$.
2. **Analysis**: Apply the statistical analysis of interest (just think the arithmetic mean for simplicity) to each imputed data set. The result of this phase is a series of m estimates $\theta_1, \ldots, \theta_m$ for the parameter of interest.
3. **Pooling**: combine the m estimates into a single estimate.

Note that deterministic imputation, e. g., regression imputation, does not make sense for step (2). This would obviously lead to $X_1 = X_2 = \ldots = X_m$ which implies $\theta_1 = \theta_2 = \ldots = \theta_m$. Thus we would gain nothing more than wasted computing power. We see that multiple imputation is just a fancy name for independent repetitions. This simple idea however alleviates the usual problem of too small standard errors. How do we calculate the final estimate? The usual

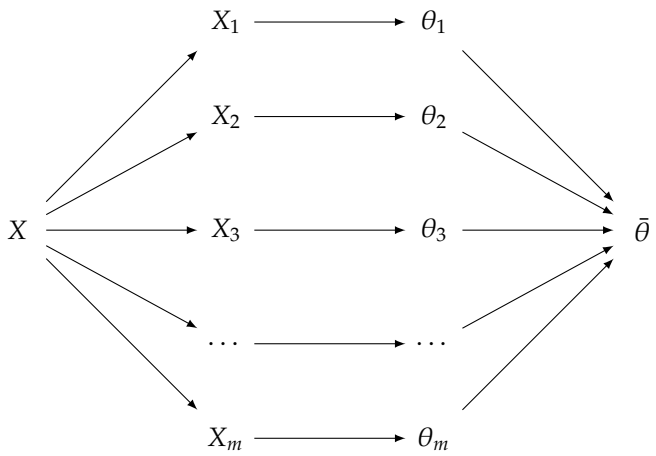


Figure 6.12: Scheme of multiple imputation phases. The incomplete data is subject to m independent imputation runs. Subsequently, the estimates are calculated and eventually combined into a single pooled estimate.

approach proposed by Rubin [Rub87] is to average

$$\bar{\theta} = \frac{1}{m} \sum_{i=1}^{m} \theta_i.$$

The total variation is slightly more involved since there are multiple sources of variation: (1) within-imputation variance: the variance that stems from the fact that only a random sample of the data was observed (our original prior-imputation data) measures the variability of the input data sets.

$$V_W = \frac{1}{m} \sum_{i=1}^{m} \operatorname{Var} X_i.$$

(2) between-imputation variance: there is additional variation introduced by the stochastic imputation technique. I.e., there is variation in the sample $\theta_1, \ldots, \theta_m$. The contribution of this source of uncertainty can be calculated by the sample variation, i.e., the mean squared distances from the pooled estimate as follows

$$V_B = \operatorname{Var}(\theta_, \ldots, \theta_m) = \frac{1}{m-1} \sum_{i=1}^{m} \left(\theta_i - \bar{\theta}\right)^2.$$

(3) The last source relates to the fact that the pooled estimate is calculated for finite m. This issue is usually addressed by adding another correction term

$$\frac{V_B}{m} \xrightarrow{m\to\infty} 0$$

which tends to zero for a high repetition number. Altogether the total variation is

$$V = V_W + V_B + \frac{V_B}{m}.$$

6.5 Exercises

Exercise 1 (Theory: MCAR and complete-case analysis)

We aim to study what percentage of data we can expect to throw away in the complete-case analysis under certain assumptions. Assume there are N observations of p variables. Assume MCAR holds for every single variable and that a value is missing with probability $\theta \in (0, 1)$ independently of all other values. The probability of at least one missing value per observations is thus

$$L(\theta, p) = 1 - (1 - \theta)^p.$$

Thus, the expected number of dropped observation in the complete-case analysis is $N \cdot (1 - (1 - \theta))^p$. Visualize $L(\theta, p)$ for different combinations of p and θ. E.g., select different values for $\theta \in \{0.001, 0.002, 0.05, 0.1, \ldots\} =: \Theta$. For each $\theta \in \Theta$ select a range for p, say between 1 and 1 000, and draw a line-plot. Alternatively, draw a three-dimensional function. Interpret the results.

Exercise 2 (Mean value in mean imputation ★)

Assume we have a single variable X and we have a sample of realizations $x_1, \ldots, x_N$ where $1 \leq k < N$ realizations are missing. Without loss of generality we can assume that the last k realizations are missing (we can always rearrange the data to fit this assumption). The mean value of the complete cases is thus

$$\bar{x} = \frac{1}{N-k} \sum_{i=1}^{N-k} x_i.$$

Likewise, the variance[8] is

$$s = \frac{1}{N} \sum_{i=1}^{N-k} (x_i - \bar{x})^2.$$

Next, we perform regression imputation and replace all k missing values with $\bar{x}$. Let $x'_1, \ldots x'_N$ be the completed data set after imputation. Let $\bar{x}' = \frac{1}{N} \sum_{i=1}^{N} x'_i$ and s' analogously.

1. Show that $\bar{x}' = \bar{x}$ holds.
2. Show that $s' = \frac{N-k}{N} s$. What does this statement mean?

[8] Actually, in the sample variance we would divide by $N - 1$ rather than N, but calculations and results are equivalent. The calculation path only becomes slightly more complicated for $N - 1$.

Literature

[Tuk77] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[Cha+83] J.M. Chambers et al. "Graphical Methods for Data Analysis". In: *The Wadsworth Statistics/Probability Series. Boston, MA: Duxury* (1983).

[Rub87] D. B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. Wiley, 1987, p. 258.

[ST96] Nathaniel Schenker and Jeremy M.G. Taylor. "Partially parametric techniques for multiple imputation". In: *Computational Statistics & Data Analysis* 22.4 (1996), pp. 425–446. DOI: https://doi.org/10.1016/0167-9473(95)00057-7.

[SB08] Juned Siddique and Thomas R. Belin. "Multiple imputation using an iterative hot-deck with distance-based donor selection." In: *Statistics in medicine* 27 1 (2008), pp. 83–102.

[End10] C.K. Enders. *Applied Missing Data Analysis*. Methodology in the social sciences. Guilford Publications, 2010.

Chapter 7

Single-Objective Optimisation

So far we discussed different topics of a modern data analysis pipeline: handling data with a sophisticated statistical programming language, dealing with missing values, detecting outliers, identifying groups with similar properties and even how to project high-dimensional data into a low-dimensional space preserving most of the information. Recall that most methods at some point of the solution finding process solved some kind of optimisation problem. E. g., in k-means clustering the algorithm opts to find a k-partition of the data (out of a plethora of possible k-partitions) such that the within-cluster function sum of squares is minimised (see Eq. 4.2). In dimensionality reduction, in PCA, we formulated calculated directions of the data where the variance is maximised subject to some additional constraints (see, e. g., Eq. 5.2). Likewise, the algorithm behind t-SNE aims to minimise the mismatch between probabilities p_{ij} in the original input space and counterparts q_{ij} in the two- or three-dimensional target space.

In this chapter we will learn about different approaches to solve (complex) optimisation problems. However, let us first formally define what we mean be. optimisation problems.[1]

We are given a *function* $f : \mathcal{X} \to \mathbb{R}$ which maps from a space $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_d$, the *decision space*, to the real-numbers, the *objective space*. The function f is

[1] The general notions and first approaches to solve such problems, e. g., gradient-descent, should be familiar from basic courses like operations research.

termed the *objective function*. The standard form of an *optimisation problem* is

$$\min_{x \in \mathcal{X}} f(x) \tag{7.1}$$

$$\text{subject to} \quad g_i(x) \leq 0 \quad i = 1, \ldots, L \tag{7.2}$$

$$h_j(x) = 0 \quad j = 1, \ldots, K. \tag{7.3}$$

$$\tag{7.4}$$

I. e., we search for a parameter $x^* \in \mathcal{X}$ which takes the minimal function value among all parameters:

$$f(x^*) = \min_{x \in \mathcal{X}} f(x).$$

subject to optional inequality and equality constraints which pose additional restrictions on the parameters chosen from the decision space. Sometimes, we instead write

$$x^* = \arg\min_{x \in \mathcal{X}} f(x)$$

if we are interested in the actual parameter x^* rather than its function value $f(x^*)$. Note that analogously, we can search for a parameter that maximises the objective function. However, to keep the definition clean we assume that we minimise. This is no restriction at all since maximising f is equivalent to minimising $-f$ (see exercises for a simple formal proof). For illustration consider the *continuous* function

$$f : \mathbb{R} \to \mathbb{R}, f(x) = x \cdot \cos(2x) + 4$$
$$\text{s. t. } x \in [-4, 4].$$

Continuous means, that the decision space is a subset of the real-valued vector space. In this example simply $\mathcal{X} = \mathbb{R}$. The decision space is subject to so-called *box-constraints* which basically restrict the valid real numbers for input to be within certain bounds. Figure 7.1 shows the function graph. The blue point represents the value $x^* \in \mathbb{R}$ which minimises the function while the three red points illustrate so-called local optima. Formally, a point $x' \in \mathcal{X}$ is a *local optimum*, if $f(x) \geq f(x')$ for all points $x \in B_\varepsilon(x') := \{x \in \mathcal{X} \mid d(x, x') \leq \varepsilon\}$. I. e., for a local optimum all points within its ε-neighbourbood have higher objective values.

Let us now consider another example: $f : \{0, 1\}^n \to \mathbb{R}$.

How do we solve optimisation problems? I.e., how to we locate a global optimum x^*? For our continuous sample function it is pretty straight-forward: we

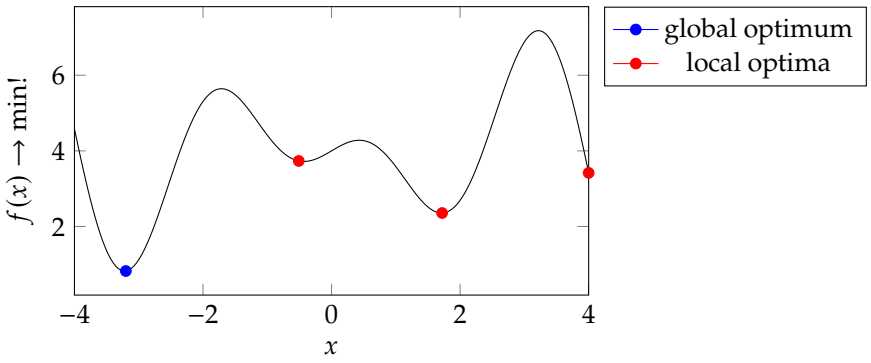


Figure 7.1: A simple single-objective function $f : \mathbb{R} \to \mathbb{R}, f(x) = x \cdot \cos(2x) + 4$.

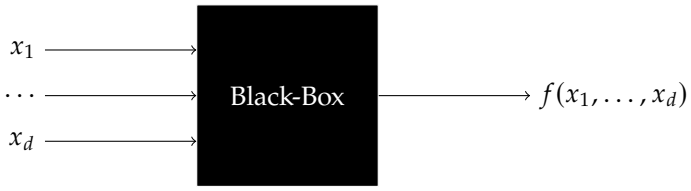


Figure 7.2: Illustration of the black-box setting. We have no possibility to obtain information from the objective function besides simple function evaluations.

apply basic calculus to solve the problem analytically and obtain an exact solution, i. e., the true global optimum given with a very high precision.[2] However, for combinatorial optimisation problems there is no method like this. In addition, even in continuous optimisation basic calculus hits its limits quite fast for several reasons.

- The function f may not be known in closed form, i.e., we do not have a nice mathematical formula we can work with. Instead we can get information on f only via function evaluations by passing values and checking the output. This setting is known as *black-box optimisation* as we can interpret f as a "black-box" that cannot look into (see Fig. 7.2 for an illustration).
- The problem can be of combinatorial nature, i.e., the goal is to find a combination out of a plethora of possibilities. Again, there is no differentiable function we can work with. In addition, many combinatorial problems of high practical relevance are inherently difficult to solve.[3] In fact, the k-means problem of finding a partition such that the sum of inner-cluster pairwise distances is maximised is an example of an $\mathcal{NP}$-hard combina-

[2] If the solution is an irrational number or has very many decimal places we in general need to sacrifice most of these decimal places.

[3] From a computer science / maths perspective these problems are called $\mathcal{NP}$-hard. Essentially this means that we do not know any algorithm that can calculate an optimal solution within acceptable time limits. We cannot go any deeper here since this is out of scope.

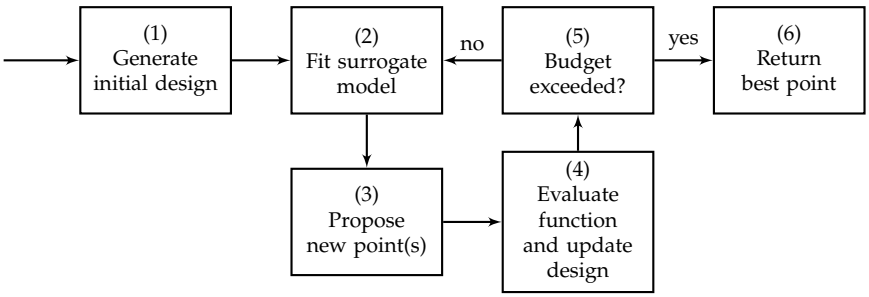


Figure 7.3: General SMBO approach.

torial optimisation problem.

- The function evaluation may be very expensive in terms of evaluation time or sheer financial costs. For example if a function evaluation requires a lab experiment or a computationally expensive simulation. This is often an additional challenge in the black-box setting.

In all these cases we therefore need to rely on so-called heuristic optimisation algorithms or short heuristics. These are algorithms that in general cannot guarantee to find a global optimum, but often find sufficiently good solutions within reasonable time. In this lecture we will learn about a special class of heuristics which take inspiration from nature: bio(logically)-inspired algorithm and in particular Evolutionary Algorithms (EAs) which mimic principles from Darwinian evolution theory like recombination, mutation and (natural) selection. This class of algorithms originates in engineering applications and has proven to be very powerful and competitive.

7.1 Sequential Model-Based Optimization (SMBO)

Algorithm 6 Generic SMBO procedure

Require: Budget, e.g., maximum number of FEs, number of initial design points, infill criterion, method to learn model etc.,

1: Generate *initial design* of n points $x^{(1)}, \ldots, x^{(n)}$ and calculate $y^{(i)} = f(x^{(i)})$.
2: Fit model $\hat{f}$ based on $(x^{(i)}, y^{(i)}), i = 1, \ldots, n$.
3: Point proposal: let n be the number of design points. Determine $m \geq 1$ points $x^{(n+j)}, j = 1, \ldots, m$ by means of the infill-criterion.
4: Evaluate the new points and add $(x^{(n+j)}, y^{(n+j)}), j = 1, \ldots, m$ to the design.
5: Go to step (2), if budget is not depleted. Otherwise return best found solution.

We now introduce a very general, but nevertheless in practical applications of the expensive black-box very successful approach. The core idea is to replace the

costly objective function f with a cheap surrogate model $\hat{f}$ which is way easier to evaluate and ideally approximates the true objective fun well. This surrogate model is than sequentially refined to fit the true function better and better. The general framework of *Sequential Model-Based Optimisation* (SMBO) [Hor+15; Bis+17; JSW98; Jon01] is illustrated in Figure 7.3 (see Algorithm 6 for generic pseudo-code). Basically all steps of SMBO offer different modules to work with. Here, we sketch the basic ideas only and mention a handful of established components. The first step is to draw an initial sample of points which are to be evaluated with f, i. e., the true objective. We call these points the *initial design (points)*. Given these points an initial function approximation $\hat{f}$ is fitted (step 2). Here, any function approximation model can be used, e. g., multiple polynomial regression, splines, regression trees or Gaussian Processes (GPs). The latter is the de-facto standard due to a huge advantage over alternative methods: GPs provide an uncertainty sample $s(x)$ for each $x \in \mathcal{X}$ besides the actual function approximation $\hat{f}(x) = \mu(x)$.[4] This is a helpful property, since in step 3 (see Fig. **??**), we need to choose at least one new point which is than evaluated with f, added to the design (step 4) and serves as the foundation for the function approximation in the next loop of the SMBO-process. This iterative procedure is repeated until some termination condition is met (step 5). Usually, in expensive black-box settings, this is either a maximal time or a maximal number of function evaluations (FEs), i. e., a maximum number of times SMBO is allowed to query the black-box f. Once the algorithm stops, the best design point is returned as the approximation of the global optimum. Now let us dive into possible choices for steps 1 and 3.

7.1.1 Initial Design

In this step the goal is to scatter n points $x^{(1)}, \ldots, x^{(n)} \in \mathcal{X}$ such that the initial function approximation $\hat{f}$ resembles the true function as good as possible. Hence, it is certainly no good idea to cluster all these points densely together as this will likely lead to a very bad approximation.[5] Thus, the goal is to scatter the points nicely in the decision space such that the global approximation quality of the initial surrogate is high. Here, the most frequently used methods are either uniform at random placement or a more sophisticated method based on Latin-Hypersquare-Sampling (LHS) where the goal is place points in such a way that the minimum distance between points is maximised.

[4] The approximation in this context is often termed $\mu(x)$ instead of $\hat{f}(x)$ since Gaussian Processes rely on Gaussian distributions and their prediction is basically a mean value of a normal distribution.

[5] An exception would be if our clustered points actually fall into the region of a global optimum of f, but as you can imagine this is a very unlikely event to happen in particular in applications with high decision space dimension.

7.1.2 Infill Criterion

The infill criterion is a crucial component of the SMBO approach. It is responsible for the proposal of new points which are to be evaluated by the expensive true objective function f in order to improve $\hat{f}$ sequentially and improve its approximation quality to resemble f better and better. Basically, all infill criteria available out there calculate a trade-off between two baseline extremes: *exploitation* and *exploration*. Pure exploitation takes full advantage of the surrogate model $\hat{f}$. It basically proposes the minimum of $\hat{f}$ in every single iteration. I. e., the proposed point is x' with $\mu(x') = \min_{x \in \mathcal{X}} \mu(x)$. This is certainly a natural and straight-forward choice, but can quickly lead to convergence of the approach in a local optimum of f. The alternative in the other direction is to purely consider the model-uncertainty $s(x)$. Proposing x' with $s(x') = \max_{x \in \mathcal{X}} s(x)$ helps to improve the global model quality. This approach however has its own drawbacks as it might propose the "wrong" points to often and do not lead to substantial improvement prior to termination. The *Lower Confidence Bound (LCB)* infill criterion aggregates both criteria discussed so far via

$$\text{LCB}(x; \lambda) = \mu(x) - \lambda \cdot s(x)$$

which is to be minimised. Here, $\lambda > 0$ is a parameter which makes the method adaptive. For λ close to zero, LCB deteriorates to pure exploitation while for large values of $\lambda \gg 0$, the model-uncertainty becomes the leading term and LCB behaves like pure model uncertainty maximisation. One of the most popular infill criteria is *Expected Improvement (EI)*. It is based on the following assumptions: Let $y_{min} = \min\{f(x^{(1)}), \ldots, f(x^{(n)})\}$ be the minimum function value among the design points in some iteration. Assume $f(x)$ to be the realisation of a normally distributed random variable Y. Now define the *improvement* as $I = \max\{y_{min} - Y, 0\}$ and maximise the *expected improvement*

$$\begin{aligned} EI(x) &= E(I(\mathbf{x})) \\ &= E\left[\max\{y_{min} - Y, 0\}\right] \\ &= (y_{min} - \hat{y}) \cdot \Phi\left(\frac{y_{min} - \hat{y}}{s}\right) + s \cdot \phi\left(\frac{y_{min} - \hat{y}}{s}\right). \end{aligned}$$

where Φ is the distribution function and ϕ is the density function of Y. The nice thing about the EI is that despite its theoretical formulation a closed formula can be derived which can be implemented directly.

Figures 7.4 and 7.5 show an example of SMBO on a simple continuous function (solid line in top plots). The dashed line is the approximation $\hat{f}$ in the respective

iteration while shaded regions illustrate the model's uncertainty; the latter being zero at the design points (no uncertainty at all) and quite in the regions in-between design points in particular if these are located far of each other. In both figures, the optimisation process starts with the same initial points. The only difference is the infill-criterion (mean, i. e., pure exploitation in 7.4 vs. EI in 7.5). We can observe a very similar behaviour in the first iterations. However, while the pure exploitation approach stagnates by proposing infill points close to the local optimum of f, the EI-based run manages to improve the global approximation quality after three iterations. This in particular improves the model quality close to the global optimum significantly. This (would) finally lead to finding a very good approximation of the global optimum within the next few SMBO iterations.

7.1.3 Closing Remarks

Sequential Model-Based Optimisation is a neat approach to attack expensive optimisation problems of black-box style. It is very flexible due to many interchangeable components. However, even though the idea is to replace the costly function f with a cheap surrogate, SMBO runs can – depending on the field of applications – take a long time. There are several reasons for this: (1) we obviously cannot avoid to query the expensive function f as we need data for the initial and reined approximations $\hat{f}$. (2) Even besides this bottleneck the approach is likely to be rather slow in comparison to classical optimisation algorithms, e. g., line-search, Nelder-Mead simplex algorithm on cheap functions, due to the many components involved. Note that (almost) all components involve some kind of optimisation on their own (LHS requires min-max optimisation, the model fitting of the approximation $\hat{f}$ requires the minimisation of some loss-function, e. g., mean squared error, and finding the minimum or maximum of the infill-criterion is usually a highly non-trivial task which can also be solved only by heuristics). Nevertheless, in expensive optimisation tasks, SMBO methods actually shine (see, e. g., the benchmarks in [Bis+17]).

Since this is no purely optimisation-related lecture, we just scratched the surface. The interested reader should read the taxonomy paper by Horn et al. [Hor+15] or the paper by Jones et al. [Jon01]. Another great source to dive into the topic is the book by Forrester, Sobester and Keane [FSK08] or, with focus on R, the paper for our R package `mlrMBO` [Bis+17].

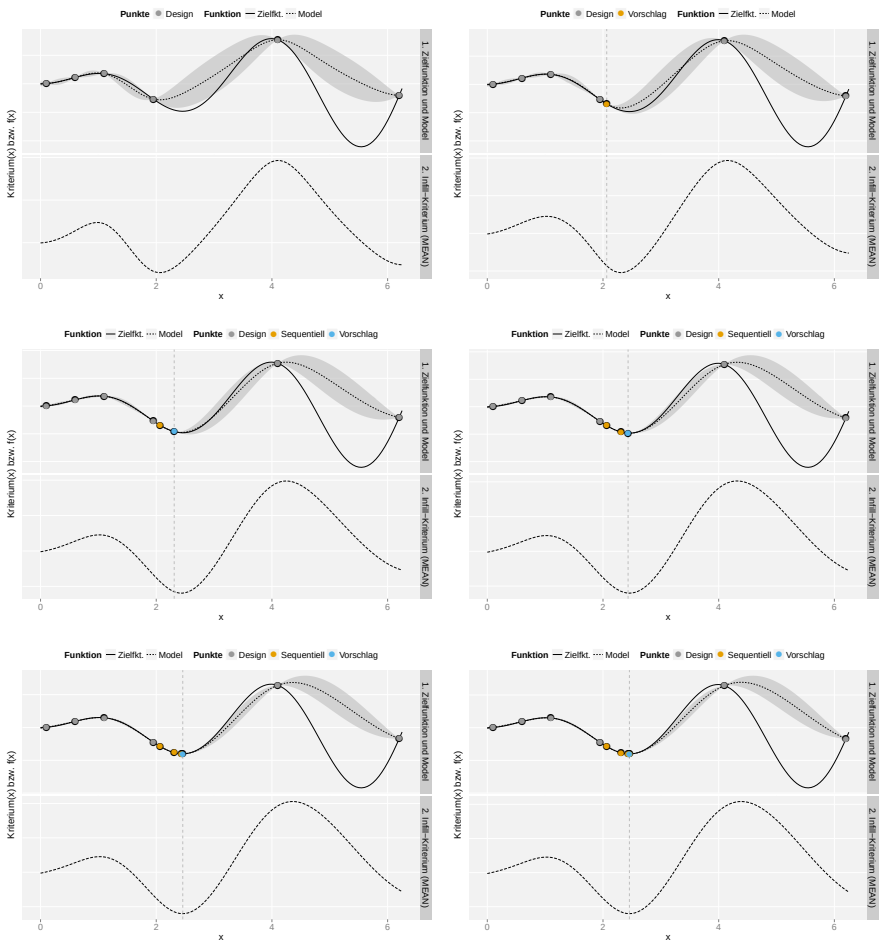


Figure 7.4: Exemplary SMBO run for five iterations with pure exploitation of the model as infill criterion. We observe a rapid convergence of the process as the function approximation changes minimally.

7.2 Evolutionary Algorithms

In the field of (applied) optimisation, engineers in the late 1960s and early 1970s got inspired by natural processes. *Evolutionary algorithms* (EAs) describe a broad class of biologically inspired heuristic algorithms. EAs mimic the principles that drive natural evolution in the Darwinian sense. In a nutshell, Charles Darwin back in the 18th century came up with the nowadays accepted theory of evolution in his book *On the Origin of Species* [Dar59]: variation is added to a population of individuals by means of sexual reproduction of the parents genes and/or asexual variation via small random changes to the genes (mutation). Individuals which are subject to a lucky combination of genes are adapted better to their respective environmental niche and thus have a higher probability to

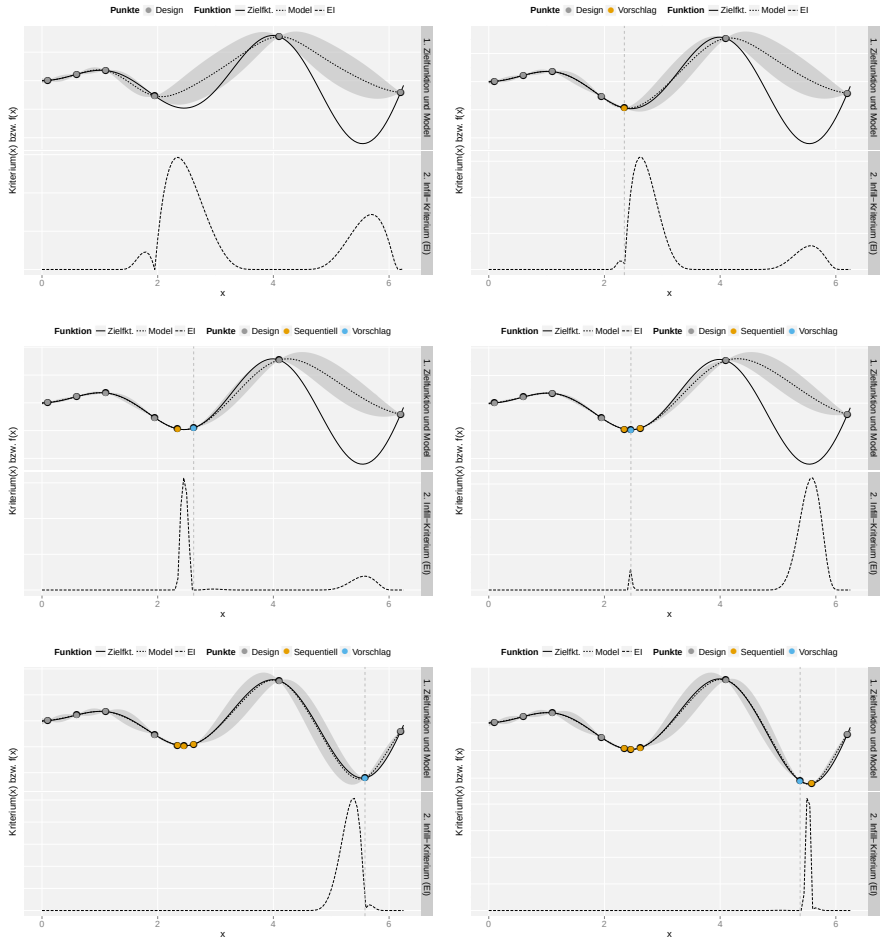


Figure 7.5: Exemplary SMBO run for five iterations adopting expected improvement (EI) for infill-point proposal. In the beginning the process behaves very similar to the one with mean infill strategy in Figure 7.4. In iteration four however, game changing point is proposed very close the global optimum of the true objective function.

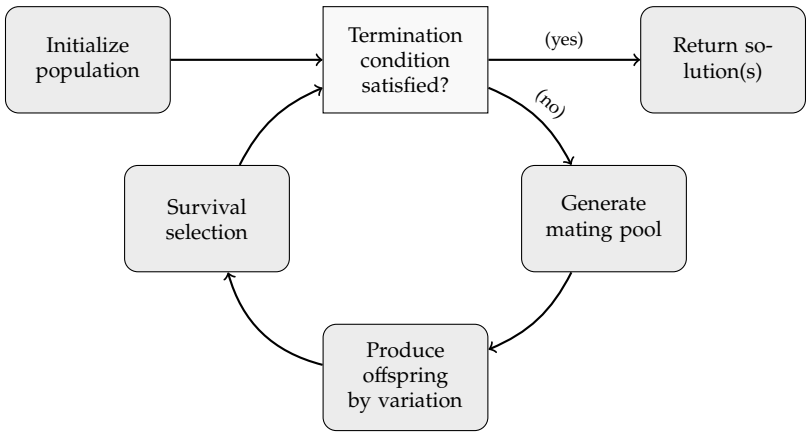


Figure 7.6: Generic cycle of an evolutionary algorithm.

Algorithm 7 (1 + 1) Evolution Strategy

Require: Fitness function $f : \mathbb{R}^d \to \mathbb{R}$, step-size σ
1: Initialise solution x uniformly at random within the box-constraints of f.
2: **while** Stopping condition not met **do**
3: $x' = x + \mathcal{N}(0, \text{diag}(\sigma, \ldots, \sigma))$
4: **if** $f(x') \leq f(x)$ **then**
5: $x = x'$
6: **end if**
7: **end while**
8: **return** x

survive in comparison to less adapted individuals. Engineers adopted this process to the optimisation domain: generate a random set of candidate solutions and assign them a "fitness" which evaluates its quality (in the most straightforward setting the fitness is the objective function value of the solution). Next, iteratively combine (reproduce) and slightly change (mutate) random individuals where the selection of parents is usually biased towards higher selection probabilities for fitter individuals (i.e., with a better/lower fitness to fit our assumption that the goal is to minimise an objective function). Eventually, let a subset of the parents and/or offspring individuals survive. Iterate this process until some stopping condition is met, e. g., maximum number of objective function evaluations, and return the best so far solution. Figure 7.6 sketches this generic "evolutionary loop". Note that – similar to SMBO – this process so far describes a very generic procedure with lots of modules: representation of solutions, initialisation of candidate solutions, recombination/mutation, parent- and survival-selection etc.

Let us first consider a very simple evolutionary algorithm for unconstrained continuous optimisation problems. The so-called (1 + 1) *Evolution Strategy (ES)*

Algorithm 8 Generic Evolutionary Algorithm

Require: Fitness function f, population size μ, number of offspring λ, number of parents ρ, further parameter, e. g., probability of mutation or specific parameter of evolutionary operators.

```
Create initial population P_0 = {x_1, ..., x_μ} at random
t ← 0                                                  // Iteration counter
while Stopping condition not met do
    O_t ← ∅
    for i = 1 → λ do
        Select ρ parents p_1, ..., p_ρ from P_t
        Recombine p_1, ..., p_ρ to generate a new individual x
        Mutate x
        Add x to O_t
    end for
    Select μ solutions from P_t and/or O_t to form P_{t+1}
    t ← t + 1
end while
return P_t
```

(see Algorithm 7) is plain simple. The algorithm initialises a population of size one, the candidate solution x, at random within the boundaries stated by the objective functions box-constraints. After initialisation the algorithm performs the following steps over an over again until some termination criterion is met which is usually a maximum number of function evaluations: a new candidate solution x' is generated by so-called *Gaussian mutation* from the sole parent x. I. e., each component x_i is changed by adding the realisation of a normally distributed random variable with zero mean and variance σ (termed the *step-size* in this context). More formally, $x'_i = x_i + \mathcal{N}(0, \sigma), i = 1, \ldots, d$. Finally, x is replaced with x' if x' is no worse than x with respect to the objective function f; otherwise x' dies immediately and x is kept. That means that we accept solutions with an at least as good objective value. That's it! If you are already familiar with classical (deterministic) gradient descent algorithms you certainly see that the (1 + 1) ES is a randomised version of gradient descent where the direction is sampled at random rather than being stoically fixed. Note that the realisation of the normally distributed variables can be quite large if σ is large, but even for small step-sizes there is a non-zero to sample a large value. Such steps may help the algorithm to escape local optima where classical non-stochastic algorithms would fail with probability one.

7.2.1 Components of Evolutionary Algorithms

Take another look at the evolutionary cycle in Figure 7.6 and the generic pseudo-code of an evolutionary algorithm 8. Obviously, everything is kept very vague

here. This is because EAs are *general purpose optimisers* and for every single component there exists a plethora of concrete realisations. All components can be adapted to the problem domain at hand! Let is briefly go through the different components.

Representation

One of the most important components is the *representation* or *encoding* of candidate solutions. This aspect is not part of Figure 7.6 neither Algorithm 8. The reason is that the representation is kind of under the hood. The representation defines how candidate solutions / individuals are presented encoded in the algorithm. Here, we usually distinguish between the *genotype*, i. e., the internal representation of a solution and its representation in the problem in the application domain termed the *phenotype*. Both representation are mapped to each other by a *genotype-phenotype mapping*. Example: the (1+1) ES (see Algorithm 7) is an algorithm designed for continuous optimisation tasks where the goal is to minimise $f : \mathbb{R}^d \to \mathbb{R}$. Here, the genotype representation is very naturally given by real-valued vectors in $\mathbb{R}^d$ and is equal to the phenotype. As a second example imagine a problem where the decision space is $\mathcal{X} \subseteq \mathbb{N}$, i. e., a subset of the integer numbers. For sake of simplicity think of $\mathcal{X}$ being the set of integers in the range $0, 1, \ldots, 2^L - 1$ for some $L > 0$. E. g., for $L = 8$ the integer numbers in the set $\{0, 1, \ldots, 2^8 - 1 = 255\}$ would be valid. In an EA we could represent these numbers as plain integer values (genotype equals phenotype). An alternative is to use a *binary encoding / sequence of L bits*.[6] Here, e. g., the for $L = 8$, the number 5 (the phenotype) would have the genotype $(0, 0, 0, 0, 0, 1, 1, 0) \in \{0, 1\}^8$ whereas the number 255 would have the genotype $(1, 1, 1, 1, 1, 1, 1, 1) \in \{0, 1\}^8$. The genotype to phenotype mapping would be

$$\mathrm{GtP}(x) := \sum_{i=1}^{L} x_i \cdot 2^{i-1}, x \in \{0, 1\}^L.$$

Common representations are real-valued string, integer string, bit-strings (in particular for subset-selection problems) or permutations (e. g., for problems in vehicle routing where the goal is to find a shortest path route through a subset of a set of cities).

Variation Operators

Variation operators are functions which map solutions to solutions. We usually distinguish variation operators by their so-called *arity*. This is the number of

[6] To encode all integers between 0 and $2^L - 1$ exactly L bits are needed.

solutions, the so-called *parents*, the operators expects as input. *Recombination operators* have arity ≥ 2. They take the information of at least two solutions and combine this information into a new solution;[7] this process is also called *mating*. The idea is – in analogy to Darwin's theory – that two good solutions are likely to produce a good or even better *offspring* solution. It should be obvious that recombination operators need to be designed genotype-specific. Classical recombination operators for real-valued representation is the intermediate recombination: given $\rho \geq 2$ parents $p_1, \ldots, p_\rho \in \mathbb{R}^d$ an offspring x is generated as

$$x = \frac{1}{\rho} \sum_{i=1}^{\rho} p_i.$$

I. e., the offspring's ith component is the arithmetic mean of the ith components of all parents. This operator can be generalised to the *weighted recombination*

$$x = \sum_{i=1}^{\rho} \alpha_i \cdot p_i$$

where $\alpha_1, \ldots, \alpha_\rho$ are (randomly sampled) weights with $\sum_{i=1}^{\rho} \alpha_i = 1$ which allow to put more or less emphasis on certain parents. Intermediate recombination is obtained for fixed $\alpha_i = 1/\rho, i = 1, \ldots, \rho$. Now imagine the two bit-string $(1, 1)$ and $(0, 1)$ as parents. Intermediate recombination would yield $(0.5, 1)$; apparently no bit-string at all. Classic recombination operators for the binary encoding are *uniform crossover* where for two parents p and q for the offspring $x_i = p_i$ with probability $1/2$ and $x_i = q_i$ with probability $1/2$ for each bit independently. Another famous operator is the 1-*point crossover*. The name says it all. For two bit-strings of length d a cut-point L between 1 and $d - 1$ is sampled at random. Subsequently, two offspring solutions x, y are generated. For the first one the first L bits are taken from p and the the last $d - L$ bits are taken from q. Offspring y is generated analogously flipping the meaning of p and q.

While recombination aims to *exploit* the "knowledge" that is already given in the *gene-pool* of the population, so-called *mutation operators* or short *mutators* have arity one. They take a single solution and apply (usually small) random changes to it in order to *explore* the decision space further. We already know the *Gaussian mutation* from the introductory $(1 + 1)$ ES. It adds some Gaussian noise to the components of a real-valued vectors. For bit-strings the *independent bit-flip mutation* is the de-facto standard. Here, the mutant x is generated from

[7] At least two is no typo! In fact, we can compile more than two solutions into a new one.

the parent x by setting $x_i = 1 - p_i$ with probability $\theta \in (0, 1)$ and keeping $x_i = p_i$ with inverse probability $(1 - \theta)$ for each position $i = 1, \ldots, d$ independently. Here, usually θ, the *probability to flip*, is chosen to be small, e. g., $\theta = 1/d$, which results in an expected number of just one flip per mutation.

Selection

There are two types of selection in typical EA. *Parent-selection* is the process of selecting solutions from the current population P_t for reproduction. In contrast, *survival-selection* decides which individuals from the current population P_t of μ solutions and the set of λ offspring O_t should survive and be transferred into the next population, i. e., the population in the next generation/iteration of the evolutionary loop. Luckily, selection operators are independent of the representation.

Parent Selection Parent-selection is usually performed at random with a bias towards choosing solution with better fitness/objective values (recombination of good genes likely to produce good offspring). If the goal is to maximise the fitness/objective a common choice is *fitness-proportional selection*. Here, the probability $p(x)$ to select a solution $x \in P_t$ as a parent is proportional to its fitness, i. e.,

$$p(x) = \frac{f(x)}{\sum_{x \in P_t} f(x)}. \tag{7.5}$$

The numerator assures that $\sum_{x \in P_t} p(x) = 1$. This approach seems valid. However, it has many drawbacks. If there is one very fit individual x in the population and many bad individuals (imagine an extreme example where $f(x) = 10\,000$ and $f(y) = 1$ for $y \in P_t \setminus \{x\}$ and $\mu = 10$) x is likely to take-over the entire population within few iterations. Why is this? $p(x)$ is close to one and fitness-proportional selection is highly unlikely to select any other solution $y \in P_t \setminus \{x\}$ for reproduction. In consequence, reproduction will yield solutions which are very similar to x or even clones of x if x is recombined with itself and/or mutation does not change anything. Since x is very fit, the offspring will be too and thus other – less fit – individuals will extinct quickly while being replaced by clones of x. If x is is global optimum this is no problem at all. On highly multi-modal problems with many local-optima x being located close to local optimum is much more likely. This take-over leads to so-called *premature convergence* which is highly undesirable since a population of clones does not give any be benefit to the search process. One solution to this problem is in replacing the fitness values $f(x)$ in Eq. 7.5 with *fitness ranks* $\mathrm{r}(x) \in \{1, \ldots, \mu\}$

which is the position of $f(x)$ in the sorted sequence off of all fitness values (in increasing order). An alternative parent-selection method is *k-tournament selection* for $k \geq 3$. Here, in order to select ρ parents the following two steps are performed ρ times:

1. Sample k out of μ solutions from P uniformly at random (i. e., without any bias) with replacement.
2. In the tournament phase select the best of these k solutions with respect to fitness.

This approach is far less biased and works for minimisation problems with ease while classic fitness-proportional selection requires adaptation to work properly in the minimisation setting (think why).

Survival Selection The task of survival selection is to take – using the notation from Algorithm **??** – the sets P_t (the current population of size μ) and the evolved set of offspring O_t of size λ and to produce another set P_{t+1} which shall serve as the population for the next iteration of the EA. Apparently, we can use any parent-selection procedure to complete this task. However, while parent selection is usually stochastic, survival selection is designed in a deterministic way. The two main strategies are $(\mu + \lambda)$-strategy (*plus*-strategy) and (μ, λ)-strategy (*comma*-strategy). In the former μ out of all $\mu + \lambda$ individuals are selection for survival whereas in the comma-strategy all individuals from P_t die, and P_{t+1} contains the best μ of the λ offspring individuals (this requires $\lambda \geq \mu$ obviously and often $\lambda \gg \mu$). The plus-strategy has a strong *selection pressure*: less fit individuals are killed rigorously and in particular the best-so-far solution always survives (so-called *elitism* behaviour). This can also lead to premature convergence. In contrast, in a comma-strategy the best-so-far individual is not guaranteed to survive which it turn creates its own drawbacks. Of course in an implementation nobody hinders us to keep track of the best-so-far solutions nevertheless. Also one can think of taking the $\mu/2$ best individuals from P_t and the other $\mu/2$ individuals from O_t. Other approaches consider the age of individuals measured as the number of iterations they persist they survived. The possibilities are countless. However, plus-strategies are the de-facto standard methods in combination with so-called *diversity-preservation* which ensure that the population does not degenerate to quickly into a multi-set of very similar solutions.

Initial Population

How do we generate initial solutions? Well, the usual and (in most cases) recommended way is to generate it at complete at random without any bias to-

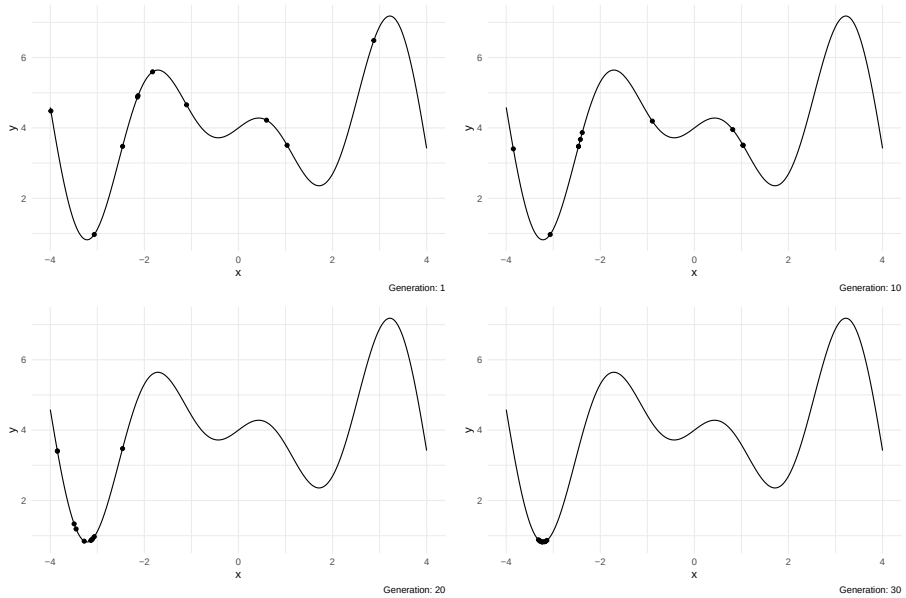


Figure 7.7: Progress of a (10 + 1) EA with intermediate recombination, Gaussian mutation and fitness-proportional parent selection on the continuous single-objective function $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x \cdot \cos(2x) + 4$. The initial population is widely scattered in the decision space. After 10 iterations the search process already shifts towards the region of the global optimum. 10 iterations later the entire population is already very concentrated in the basin of attraction of the global optimum. Finally, after 30 generations a take-over effect can be observed: the population degenerated into a set of very similar solutions.

wards specific regions of the search/decision space. The exact implementation depends on the encoding of solutions. For real-valued vectors each component would be sampled from a uniform distribution in the range of the respective lower and upper box constraints or within the feasible numerical range if the problem is unbounded. For bit-string the initialisation is even simpler. Set each bit to 1 or 0 with probability 1/2 independently of the other bits.[8] The alternative is to seed the initial population with solutions that stem from short runs of other heuristics (e. g,, local search algorithms or approximation algorithms). However, there is a high risk to fill up the initial population with many very similar solutions. This is not desirable at all; in the worst-case all initial solutions are close to each other and the algorithm is trapped in a premature convergence setting from the beginning on!

The initialisation problem becomes much more challenging if the problem is restricted/bounded by several constraints. Random uniform initialisation in these case may yield a population where all solutions are *infeasible*, i. e., all solu-

[8] Independently means that the outcome for the ith bit is not affected by the random choices of bit positions $j \neq i$.

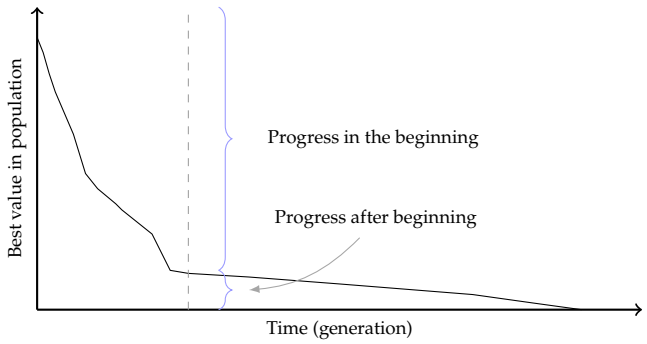


Figure 7.8: Illustration of a typical progress of an evolutionary algorithm (inspired by Figures 3.5, 3.6 and 3.7 in [ES15]).

tions violate at least one constraint. In these cases, an adaptation of the fitness function (e. g., via penalisation of infeasible solutions), repair mechanisms or a more directed initialisation may be necessary. These topics are out of scope however; I suggest to dive into later chapters of [ES15] to learn more about these topics.

Termination Criteria

Stopping conditions are rather straight forward. One can either use a maximum number of function evaluations or a time limit (see SMBO). Quite common are also termination criteria which measure the progress over time and quit the process once in the last L iteration either no progress – i. e., improvement in the fitness value of the best-so-far solution, the average solution quality etc. – at all was made or the progress was just infinitesimally small. This is also called *stagnation detection*. This is highly relevant as in the case of early premature convergence the algorithm is unlikely to make any more progress. In such cases it makes sense to not waste more function evaluations and instead restart the algorithm from scratch (see also the following section on anytime behaviour) . Final remark: a mixture of stopping criteria is of course perfectly possible. Here, the EA stops once at least one stopping condition is true.

7.3 Stochastic Nature and Anytime Behaviour

Evolutionary algorithms are inherently stochastic: initialisation, parent-selection and variation introduce a remarkable number of random decisions which all-together decide on the solution quality with respect to the fitness/objective function of the best solution in the final population. Thus, usually EAs are restarted multiple times. I. e., multiple independent runs are performed on the

same problem and the final solution is the best over all solutions gathered in all runs.

EAs are also so-called *anytime algorithms*. I. e., EAs can be interrupted at any time returning valid solutions. This is in strong contrast to many classical (exact) algorithms or so-called construction heuristics which cannot be stopped prematurely since no solution is build in-between. Figure 7.8 shows a typical progress of an anytime algorithm. This progress gives insights into several important aspects worth consideration:

- Typically EAs make very significant progress in the beginning. At some point in time stagnation begins where the algorithm makes little progress / small step improvements. This is plausible since random initial solutions are usually far from optimal such that variation operators in combination with survival of the fittest yields many improvements. Once the algorithm converges into a (local) optimum, small steps (and the right steps) are necessary in order to improve. Since the probability for improving steps decreases the waiting time until such an event takes place increases. From this we can learn that very long runs, i. e., a way to high budget, are usually not that advantageous. It is rather advised to stop the algorithm once convergence occurs (e. g., via some kind of stagnation detection) and save the budget for further independent restarts.
- The strong progress in the very beginning is an indicator that rather complex initialisation of the population may not be worth the effort.

7.4 Exercises

Exercise 1 (Elementary relation)

Show formally, that for a function $f : \mathcal{X} \to \mathbb{R}$ it holds that minimising f is equivalent to maximising $-f$.

Exercise 2 (Dynamic step-size adaptation)

Algorithm 9 (1 + 1) Evolution Strategy

Require: Fitness function $f : \mathbb{R}^d \to \mathbb{R}$, initial step-size σ

1: Initialise solution x uniformly at random within the box-constraints of f.
2: **while** Stopping condition not met **do**
3: $x' = x + \mathcal{N}(0, \text{diag}(\sigma, \ldots, \sigma))$
4: **if** $f(x') \leq f(x)$ **then**
5: $x = x'$
6: **end if**
7: Modify σ according to Rechenberg's 1/5-rule (optional).
8: **end while**
9: **return** x

Consider the function $f : \mathbb{R}^{10} \to \mathbb{R}$ with $f(x) := \sum_{i=1}^{10} i \cdot x_i^2 \to \min!$

- What is the global optimum of f?
- Implement a simple (1+1) ES with fixed step-size σ (see Algorithm 9) and another version where the step-size is adapted by means of the so-called 1/5-*rule by Rechenberg*. The latter rule tracks the ratio P_s of successful mutations (i.e., mutations that replace the current-best solution) given a measurement period M. More precisely, the number N_s of successful mutations is tracked for M consecutive iterations; then $P_s = N_s/M$. After the measurement period the step-size is dynamically adapted as follows:

$$\sigma = \begin{cases} \sigma \cdot a, & \text{if } P_s > 1/5 \\ \sigma / a, & \text{if } P_s < 1/5 \\ \sigma, & \text{if } P_s = 1/5. \end{cases}$$

  where a is the *step-size multiplier* and usually set to a value in $[1.1, 1.5]$.
- Run both versions with random initialisation multiple times on f; choose an adequate termination condition (e. g., gap to known optimum is smaller then some threshold). For evaluation, track the best solutions over the course of optimisation and plot the distance to the optimum (it might be beneficial to use a logarithmic scale of the distance). In addition, track the development of the step-size σ and plot it. Describe your observations.

Exercise 3 ((1 + 1) EA progress on `OneMax`)
The so-called `OneMax` problem is defined as $\texttt{OneMax}(x) = \sum_{i=1}^{n} x_i$ with $x \in \{0,1\}^n$, i. e., the decision space consists of all binary strings of length n. Implement a simple (1 + 1) EA to solve the problem. The algorithm should generate

the initial solution uniformly at random and use mutation only. In the mutation step, the offspring x' should be generated from the current solution x by flipping each bit of x independently with probability $p = 1/n$. Run the algorithm 30 times to solve `OneMax` for $n = 50$ until the optimum – the all-ones string – is found. In each iteration, log the current function value. After having collected the data, for each run log the average fitness increase in the first half of the iterations needed and likewise for the second half. It also makes sense to plot convergence-plots (iteration on x-axis and fitness of best-so-far solution on y-axis). Interpret your results. Do you think the observations will carry over to other domains/problems and other (more sophisticated) EAs? Justify your answer(s).

Literature

[Dar59] Charles Darwin. *On the Origin of Species by Means of Natural Selection*. or the Preservation of Favored Races in the Struggle for Life. London: Murray, 1859.

[JSW98] Donald R. Jones, Matthias Schonlau, and William J. Welch. "Efficient Global Optimization of Expensive Black-Box Functions". In: *Journal of Global Optimization* 13.4 (1998), pp. 455–492.

[Jon01] D. R. Jones. "A taxonomy of global optimization methods based on response surfaces". In: *Journal of Global Optimization* 21.4 (2001), pp. 345–383.

[ES15] Ágoston E Eiben and James E Smith. *Introduction to Evolutionary Computing*. Springer, 2015. DOI: 10.1007/978-3-662-44874-8.

Chapter 8

Multi-Objective Optimisation

Recall the generic optimisation problem in Eq. 7.1 in Chapter 7. The goal is to find a parameter $x^* \in \mathcal{X}$ such that $f(x^*) = \min_{x \in \mathcal{X}} f(x)$. I. e., the function value of x^* is minimal. In multi-objective optimisation we need to deal with multiple objective functions at once that all need to be optimised simultaneously. Formally, we are given a vector-valued function

$$f : \mathcal{X} \to \mathbb{R}^m, x \mapsto \begin{pmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{pmatrix} \in \mathbb{R}^m$$

where $f_i : \mathcal{X} \to \mathbb{R}$ for $i \in [m] := \{1, \ldots, m\}$ are single-objective functions. The goal is to identify solutions which minimise all objectives simultaneously.

8.1 Basic Definitions and Vocabulary

How do we define optimality if there are no scalar but vector-valued objective values? In single-objective optimisation things are pretty easy: for $x, x' \in \mathcal{X}$ either $f(x) \leq f(x')$, $f(x) \geq f(x')$ or $f(x) = f(x')$. In any case we can compare arbitrary numbers with these elementary binary relations. The reason is that there is a *total order* in $\mathbb{R}$. In $\mathbb{R}^m$ things are different. Clearly,

$$\begin{pmatrix} 1 \\ 6 \end{pmatrix} \leq \begin{pmatrix} 2 \\ 8 \end{pmatrix} \text{ and } \begin{pmatrix} 4 \\ 9 \end{pmatrix} \leq \begin{pmatrix} 5 \\ 9 \end{pmatrix},$$

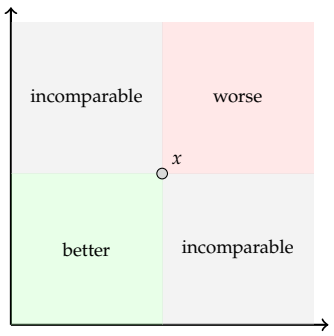


Figure 8.1: Illustration of the dominance concept. All points in the red area are dominated by x, all points in the green area dominate x. Points located in the gray areas are incomparable.

but what is the relation between the following two vectors?

$$\begin{pmatrix}1\\4\end{pmatrix} \overset{?}{\sim} \begin{pmatrix}2\\3\end{pmatrix}$$

Obviously, the first vector is better with respect to the first component, but worse than the second with respect to the second components. Here, we adopt the concept of dominance to first introduce a notion for the comparison of vectors. You may want to glance at Figure 8.1 to support the understanding.

Definition 8.1 (Dominance)**.** Let $x, y \in \mathbb{R}^m$ for $m \geq 2$ be two vectors. We say

1. x *weakly dominates* y, denoted as $x \preceq y$, iff

$$\forall i \in [m] : x_i \leq y_i.$$

2. x *dominates* y, denoted as $x \prec y$, iff

$$\forall i \in [m] : x_i \leq y_i \text{ and } \exists i \in [m] : x_i < y_i.$$

3. If neither $x \preceq y$ nor $y \preceq x$ we say that x and y are *incomparable/incommensurable*, denoted as $x \parallel y$.

Sometimes it is easier to understand the following equivalent definition of dominance: for $x, y \in \mathbb{R}^m$ it holds that $x \prec y$ if and only if $x \preceq y$ and $x \neq y$ where the less-or-equal relation is to be interpreted component-wise. This is just a less formal notation, but less precise. We can transfer the dominance concepts to sets of points X and Y. $X \preceq Y$ if for any $y \in X$ there is a points $x \in X$ which weakly dominates y. Next, we transfer the dominance concept to the optimisation domain:

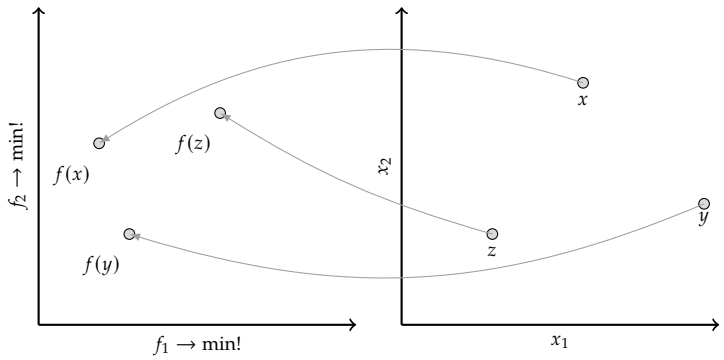


Figure 8.2: Illustration of Pareto-dominance concept. The decision space (right plot) is mapped to the 2-dimensional objective space (left plot) by f. Here, x dominates z and likewise y dominates z while x and y are incomparable. Note that the Pareto-dominance concept relies on the dominance relation in the objective space and not in the decision space.

Definition 8.2 (Pareto-dominance)**.** Given $f : \mathcal{X} \to \mathbb{R}^m$ a point $x \in \mathcal{X}$ *(Pareto-)dominates* another point $y \in \mathcal{X}$ if

$$\forall i \in [m] : f_i(x) \leq f_i(y) \text{ and } \exists i \in [m] : f_i(x) < f_i(y).$$

In this case we also say that $f(x)$ (Pareto-)dominates $f(y)$.

Note that Definition 8.2 speaks about domination of points in the decision space (see Figure 8.2). The next definition finally introduces what we consider to be optimal points in multi-objective optimisation.

Definition 8.3 (Pareto-optimality)**.** For $f : \mathcal{X} \to \mathbb{R}^m$ with $m \geq 2$, a solution $x \in \mathcal{X}$ is *Pareto-optimal*, if there is no solution $y \in \mathcal{X}$ such that $f(y) \prec f(x)$.

In words: a solution is Pareto-optimal if it is not (Pareto-)dominated by any other solution. A Pareto-optimal solution thus is an optimal compromise of the objective function values. This is why we call Pareto-optimal points sometimes *optimal trade-offs*, *optimal trade-off solutions* or *optimal compromises*.
Finally, the task in multi-objective optimisation is to find the *Pareto-set*, i. e., the set of all non-dominated solutions

$$\text{PS} = \{x \in \mathcal{X} \mid !\exists y \in \mathcal{X} : f(y) \prec f(x)\} \subset \mathcal{X}$$

and/or the image of the Pareto-set in the objective-space, the so-called *Pareto-front*

$$\text{PF} = f(\text{PS}) = \{f(x) \mid x \in \text{PS}\} \subset \mathbb{R}^m.$$

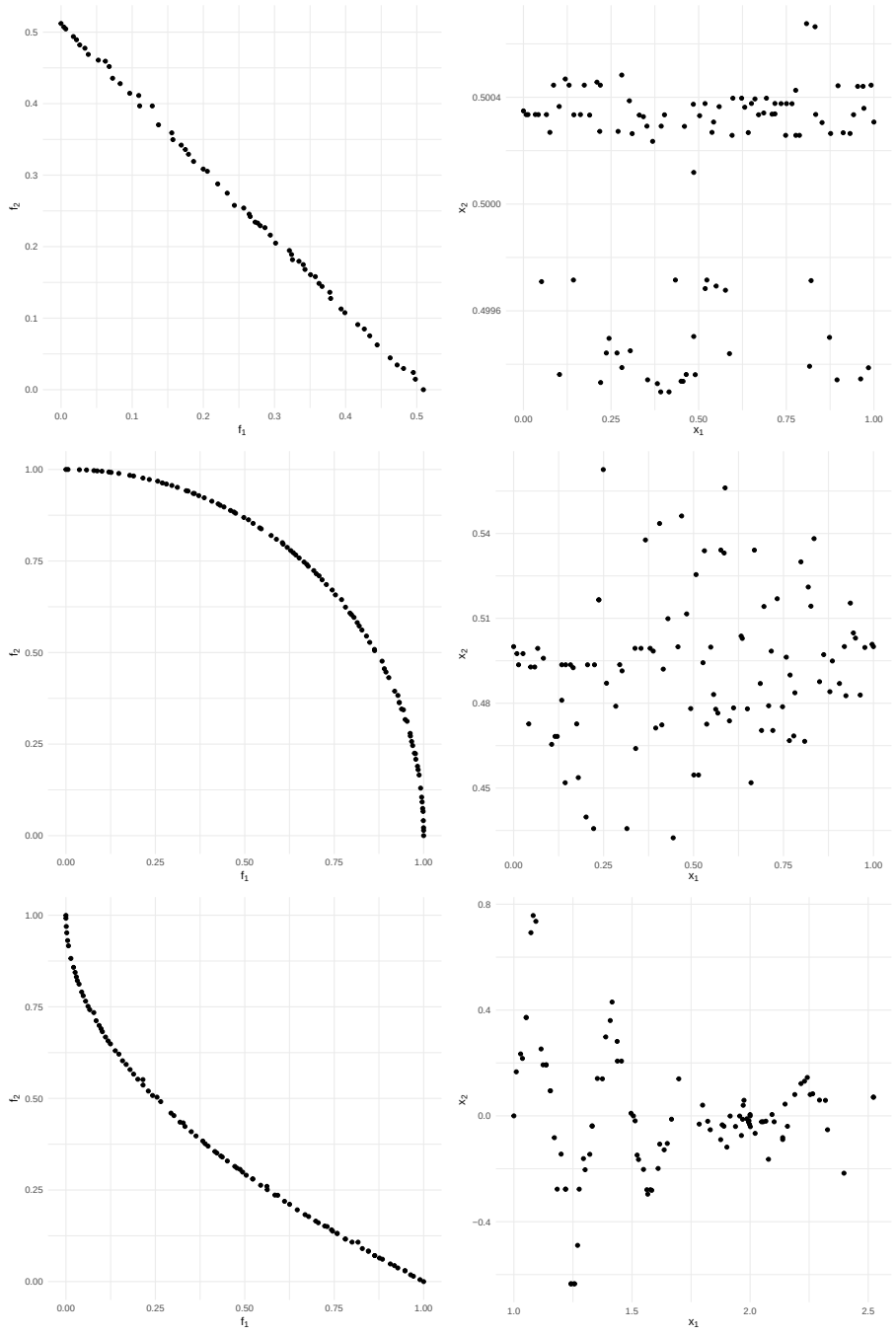


Figure 8.3: Decision space (right column) and objective space (left column) for three exemplary benchmark functions $f : \mathbb{R}^2 \to \mathbb{R}^2$ where both objectives are to be minimised. The functions are DTLZ1, DTLZ2 and MMF7 (from top two bottom). Each point depicted in the decision space is mapped to one of the points on the Pareto-front approximation (see also Figure 8.2. These examples nicely illustrate different "shapes" of the Pareto-front and the spread of Pareto-optimal solutions in decision space.

Figure 8.3 shows examples of the approximated PSs and PFs for three different benchmark functions from the literature of the type $f : \mathbb{R}^2 \to \mathbb{R}^2$ (i. e., 2d real-valued decision space dimension and two objectives). You can imagine that the task of finding PS and PF is not trivial. The reasons are manifold. First of all we learned that even single-objective problems can be very challenging through various reasons, e. g., the objective function is expensive or a black box. Obviously, these problems directly carry over to multi-objective optimisation. Another reason is that often it is impossible to calculate the whole Pareto-set since it contains infinitely many solutions (usually the case in continuous optimisation) or exponentially many (in combinatorial optimisation); in combinatorial optimisation such problems are called *intractable*. In consequence once again heuristic algorithms are the solution. One straightforward way to find Pareto-optimal solutions is to reduce the multi-objective problem to an auxiliary single-objective problem; a process called *scalarisation*. *Weighted-sum scalarisation* is the simplest of these approaches. Here, a set of m positive weights $\lambda_1, \ldots, \lambda_m$ that sum up to one is chosen and used to weight the components of the multi-objective problems by summing up the weighted components as follows

$$f_s : \mathcal{X} \to \mathbb{R}, x \mapsto \sum_{i=1}^{m} \lambda_i \cdot f_i(x) \in \mathbb{R}.$$

Now we can proceed and solve $f_s \to \min!$ with the single-objective method of our choice. If we solve the problem exactly, the resulting solution x^* with $f_s(x^*) = \min x \in \mathcal{X} f_s(x)$ is in fact guaranteed to be Pareto-optimal. However, there are several issues with this approach: (1) the single-objective problems will usually remain hard to solve, (2) we obtain only a single solution only and thus need to repeat the process multiple times for different weights, (3) it is rather unclear how to choose the weights and (4) the weighted sum approach cannot guarantee to find all Pareto-optimal solutions. In fact, this is guaranteed only for solutions which map to the convex part of the Pareto-front. There is a large pool of different approaches proposed in the literature. Later in this chapter we will introduce evolutionary algorithms designed to solve multi-objective problems.

8.2 Performance Evaluation

Before we dive into some evolutionary methods let us speak about performance evaluation. Why do we need this? First of all one of the methods, the algorithm SMS-EMOA, uses an performance indicators internally. The second reason is

that performance measurement in multi-objective optimisation is far less trivial than it is in single-objective optimisation. In single-objective optimisation, given a problem and multiple stochastic algorithms, we can simply compare the distribution of the solution quality over multiple runs of each algorithm. If we know the true global optima, we can even leverage this and instead analyse the distribution of the distance to the known optimum.[1] However, in multi-objective optimisation each run of an algorithm returns an approximation of the Pareto-front! What is a reasonable approach to compare sets of points? A common idea is to define a *(performance) measure / (performance) indicator / (quality) indicator* $I : \mathbb{R}^m \to \mathbb{R}$ which maps the complex, semi-ordered vector-valued objective space to the real numbers. This way a total order on sets is established. I. e., given two sets A and B, we can quantify whether (and to which extent) A is better/worse than B considering $I(A)$ and $I(B)$. Performance indicators usually focus on one or multiple of the following categories:

- **Cardinality**: Simply count the number of non-dominated solutions in the produced approximation sets.
- **Convergence**: How "close" is the approximation set to the true Pareto-front? Such measures obviously lack practical applications since we do not known the true Pareto-front; otherwise it would be pointless to run optimisation algorithm. Many of the measures in this category rely on a so-called reference set which is either a (subset of) the true Pareto-front or an approximation thereof. The reference set can be approximated by filtering the non-dominated points of the the union set of all approximations of all algorithms applied to a problem.
- **Spread**: Ideally the Pareto-front approximation should cover the objective space nicely instead of offering solutions limited to certain regions only.

Figure 8.4 gives a simple example. Here, the Pareto-front consists of 9 points (grey circles) and we are given three approximations A_1, A_2 and A_3. We have $|A_1| = 7$, $A_2 = 5$ and $|A_3| = 6$. Hence, with respect to cardinality A_1 is better than A_3 which itself is better than A_2. However, clearly A_3 is "farthest away" from the Pareto-front. A_1 and A_2 in contrast are both quite converged to the Pareto-front. With respect to spread or diversity of solutions again A_1 is the winner. A_2 is not bad at all, but there is "gap" with no solution between the solution in the bottom right corner and its closest neighbour in the center of the plot. Without having introduced any indicator so far this simple example together with the high-level concepts of of cardinality, convergence and spread already shows the need for multiple performance indicators in order to compare

[1] This approach is of course limited to empirical studies on well-known artificial benchmark function

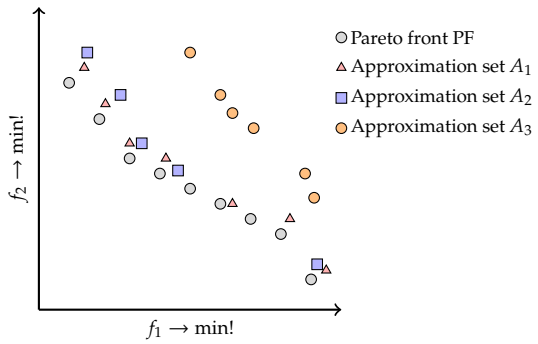


Figure 8.4: Illustration of the distance calculations in GD (left) and IGD (right): each point in A is assigned to the closest point in the reference set R or the vice versa.

Pareto-front approximations of different algorithms.

Let us now consider a first – quite bad – example of an indicator.

8.2.1 Error ratio

Given an approximation set A and the Pareto-front PF the *error ratio* is the percentage of points in A which are not Pareto-optimal

$$e(A) = \frac{|\{x \in A \mid x \notin \text{PF}\}|}{|A|}.$$

This measure is appealing due to its simplicity. However, it has some serious drawbacks. First of all the Pareto-front must be known (see above). Second, if $|A| = 1$ and it contains a single Pareto-optimal point we obtain $e(A) = 0$ (the minimum value) while for many close-to-optimal (i. e., not Pareto-optimal) points we would obtain $e(A) = 1$ (the maximal value). In addition – even though we did not introduce any multi-objective algorithms so far – it seems not adequate to "guide" an algorithm towards the Pareto-front: I am sure you can come up with an artificial example of two sets $A, B \subseteq \mathbb{R}^2$ of points and a Pareto-front PF $\subset \mathbb{R}^2$ such that $e(A) = e(B) = 1$, but A is way better than B with both respect to cardinality, convergence and spread.

8.2.2 Desirable Properties of Quality Indicators

The above insights call for some formal requirements on quality indicators. Which desirable properties can we imagine? In the following definition we concisely capture a set of requirements an discuss the properties later on

Definition 8.4 (Requirements for quality indicators)**.** Let $I : \mathbb{R}^m \to \mathbb{R}$ be a quality indicator which is to be maximized without loss of generalization.[2] Let $A, B \subseteq \mathbb{R}^d$ be two multi-sets.

Compatibility with the dominance relation I is *weakly compatible with respect to the dominance relation* iff $\forall A, B$:

$$I(A) \geq I(B) \Rightarrow A \preceq B.$$

Soundness with respect to the dominance relation I is *sound* iff $\forall A, B$:

$$A \preceq B \Rightarrow I(A) \geq I(B).$$

Monotonicity I is *monotonic* if $I(A \cup \{x\}) > I(A)$ for a non-dominated point $x \notin A$. I is *weakly monotonic* if the addition of a at least does not worsen the indicator value.

Relativity I is called *relative* if each approximation A of the Pareto-front has a worse value than PF, i. e., $I(A) < I(\text{PF})$. I is *weakly relative* if each approximation has no better value than the Pareto-front.

The first two properties relate indicator values to the dominance relation. The *compatibility with the dominance relation* requires that if the indicator value for approximation set A is as least as high as $I(B)$ then for any point in $y \in B$ there must be (at least) one point $x \in A$ that dominates. Soundness requires the other way around. Monotonicity and relativity are self-exploratory. The error-ratio violates almost all these properties! It is by no means compatible or sound neither is it (weakly) monotonic. However, it is at least weakly relative since the best value zero is taken if $A \subseteq \text{PF}$.
Let us know consider some more advanced measures. Let A be an approximation set and R be a reference set.

8.2.3 (Inverted) Generational Distance

Then *generational distance* GD is defined as the arithmetic mean of (Euclidean) distances of the approximation to the reference set:

$$\text{GD}_p(A, R) = \frac{1}{|A|}\left(\sum_{i=1}^{|A|} d_i^p\right)^{1/p}$$

[2] Some indicators, e. g., the error ratio, require minimization. However, in order to avoid the need for tedious case distinctions (minimize vs. maximize) we can assume maximization without biasing the general statement.

where d_i is the (Euclidean) distance of point $x_i \in X$ to its closest neighbour in R and p is usually set to 1. It should be clear that this measure is to be minimised. Figure 8.5 illustrates the closest-neighbour mapping. Obviously, this measure requires a reference set again (ideally the Pareto-front). Moreover, it is not weakly monotonic since non-dominated not Pareto-optimal points worsen the measure. At least it is weakly relative, i. e., $\mathrm{GD}(A) = 0$ if $A \subseteq \mathrm{PF}$. Instead of mapping each points from A to its closest neighbour another valid approach is to change the roles and calculate the distance of each reference point to the closest neighbour in the approximation set (see the right plot in Figure 8.5). This is known as the *inverted generational distance* (IGD)

$$\mathrm{IGD}_p(A, R) = \frac{1}{|R|}\left(\sum_{i=1}^{|R|} d_i^p\right)^{1/p}.$$

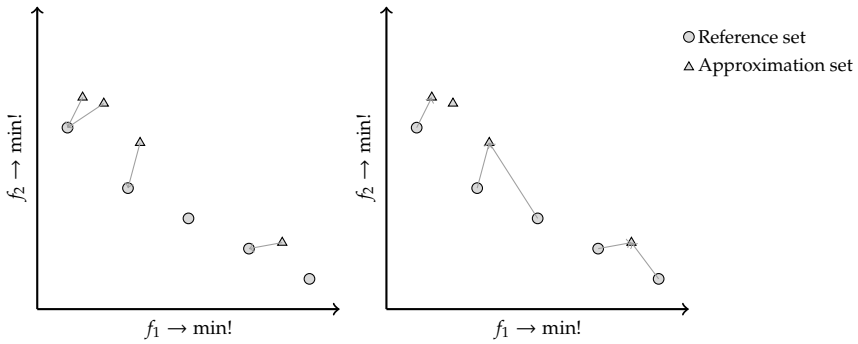


Figure 8.5: Illustration of the distance calculations in GD (left) and IGD (right): each point in A is assigned to the closest point in the reference set R or the vice versa.

8.2.4 Dominated Hypervolume (HV)

The *dominated hypervolume* (HV) is a measures with very nice properties. It does not require the knowledge of the Pareto-front. Instead a single *anti-optimal reference point* $r \in \mathbb{R}^m$ is required. Anti-optimal means that the point must be chosen such that it is dominated by all points in the approximation set. HV is than the dominated space enclosed by the points in A and r. This can be illustrated nicely in two dimensions (see Figure 8.6 left hand side) whereas the formal mathematical formulation looks quite odd to to its generality to arbitrary

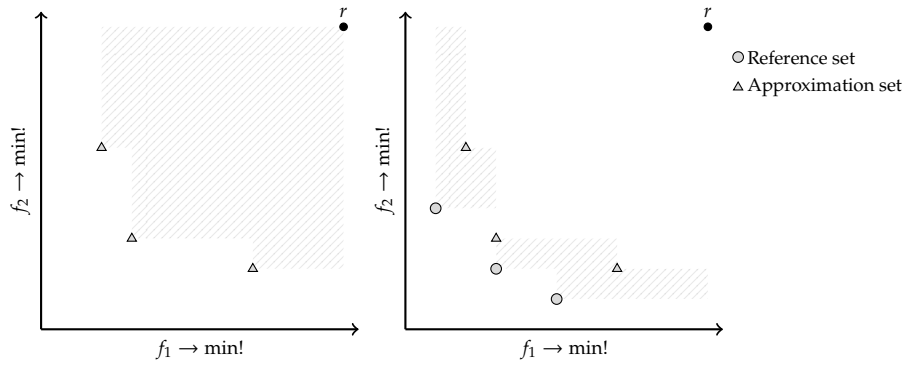


Figure 8.6: Example of the dominated hypervolume (left) and the dominated hypervolume difference to a reference set R (right).

dimensions:

$$\mathrm{HV}(A, r) = \lambda_m \left(\bigcup_{x \in A} [x \preceq x' \preceq r] \right).$$

where λ_m is the *m-dimensional Lebesgue measure*. The higher the dominated HV (given a fixed reference point), the better is both the convergence and the diversity/spread of the solutions. HV satisfies most of the nice-to-have conditions introduced earlier in this chapter: monotonicity, relativity, soundness and compatibility (see Definition 8.4). The only critical aspect is its sensitivity of the choice of the reference point. Nevertheless it is one of the de-facto standard measures in (evolutionary) multi-objective optimisation. If the Pareto-front is known (for example in benchmark studies) the *hypervolume difference* (see right hand side plot in Figure 8.6) makes sense

$$\mathrm{HVD}(A, R, r) = \mathrm{HV}(R, r) - \mathrm{HV}(A, r).$$

It is – as the name suggests – the difference between the hypervolume of the reference set (and the reference points) and the approximation set (and the reference set). Since $R \preceq A$, $\mathrm{HVD}(A, R, r) \geq 0$ and is zero if and only if $R = A$.

8.3 Evolutionary Multi-Objective Algorithms

Now it is time to design some algorithms which approximate the set of Pareto-optimal solutions. Evolutionary algorithms quite naturally fit into the multi-objective optimisation domain due to their population-based approach. A population is a multi-set of candidate solutions and it seems natural to extend EAs to evolve multi-sets of non-dominated solutions. Such algorithms are

Algorithm 10 Non-dominated sorting genetic algorithm II [Deb+02]

```
1: Initialise P_0
2: t ← 0
3: while Termination condition not met do
4:    O_t ← variation(P_t)
5:    F_1, F_2, ... ← NDS(P_t ∪ O_t)                    // Sort by NDS
6:    P_{t+1} ← ∅
7:    i ← 1
8:    while |P_{t+1}| + |F_i| ≤ μ do
9:       P_{t+1} ← P_{t+1} ∪ F_i
10:      i ← i + 1
11:   end while
12:   Calculate CD for F_i and sort in descending order
13:   Add first (μ − |P_{t+1}|) elements of F_i to P_{t+1}
14:   t ← t + 1
15: end while
16: return P_t
```

termed **E**volutionary **M**ulti-**O**bjective **A**lgorithms (EMOAs) and many of these are state-of-the-art in industry. Which modules of EAs do require modification to turn the algorithm into an EMOA? Well, basically it is about survival-selection. Recall that survival selection is usually a deterministic operation. In a $(\mu + \lambda)$-strategy we select the μ "best" out of $\mu + \lambda$ individuals. Likewise, in an EMOA we could simply select all non-dominated solutions among the $\mu + \lambda$ candidates. However, let N be the set of non-dominated solutions. It is likely that $|N|$ is smaller or larger than μ. Thus, this criterion is not enough. This is why many EMOAs adopt a two-phase procedure.

8.3.1 Non-Dominated Sorting Genetic Algorithm II

The first such algorithm we are going to learn about is the **N**on-Dominated **S**orting **G**enetic **A**lgorithm (NSGA-II) [Deb+02]. NSGA-II (see Algorithm 10) adopts the eponymous *non-dominated sorting* (NDS) procedure as a sub-routine in the first phase of the selection process. Pseudo-code of NDS is given in Algorithm 11. The core idea is to sort the points of a set into layers by a very simple approach: let X be the original set of points. NDS first determines the set of non-dominated points in X. These points are assigned to the first layer F_1 and removed from X. Now NDS again calculates the non-dominated points of the remaining points in X, assigns them to layer F_2 and subsequently removes them from X. This process continues until all points are assigned to a non-domination level. Given this sorting $F_1, \ldots, F_k$ we can assume that F_1 is better than F_2, F_2 is better than F_3 and so on. Therefore, it is desirable to keep points from the first few layers and this is exactly what NSGA-II does. It fills

Algorithm 11 Non-dominated sorting (NDS)

Require: Set of points $X \subset \mathbb{R}^m$.
1: $i \leftarrow 1$ // Layer counter
2: **while** $X \neq \emptyset$ **do**
3: $F_i \leftarrow$ set of non-dominated points in X
4: $X \leftarrow X \setminus F_i$ // Remove non-dom. points
5: $i \leftarrow i + 1$
6: **end while**
7: **return** $F_1, F_2, \ldots, F_{i-1}$

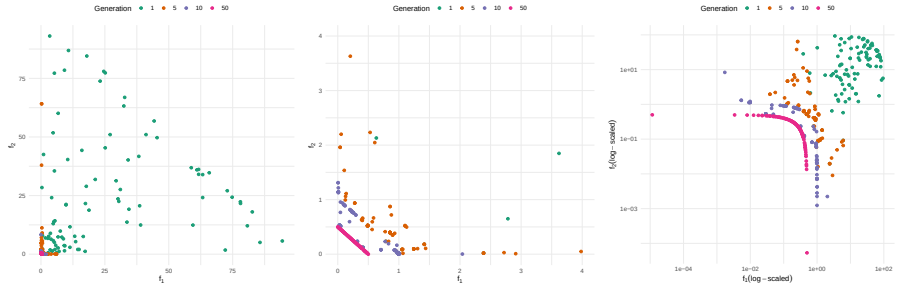


Figure 8.7: Visualization of the progress of NSGA-II in objective space only on benchmark function DTLZ1. Show is the entire population in a couple of generations. The plots show the overall image (left), a zoomed area in $[0, 4] \times [0, 4]$ (allows to see the fine-grained process in later iterations and the convergence to the linear Pareto-front in the last depicted iteration (middle); see also Figure 8.3 and the left-most plot with logarithmically transformed axis (allow to see the full picture in a distorted/transformed space (right).

the next population layer by layer until the addition of a complete layer, say F_i, would surpass the population size μ. Then, a second criterion is adopted to decide which subset of the points in F_i to keep. The *crowding distance* (CD) is calculated for each point in F_i individually via

$$\mathrm{CD}(x_{(i)}) = \sum_{j=1}^{m} \frac{|f_j(x_{(i+1)}) - f_j(x_{(i-1)})|}{\max_i f_j(x_i) - \min_i f_j(x_i)}.$$

Here, $x_{(i)}$ is ... CD measures the space around individual solutions. The points are sorted in decreasing order of the crowding distance and the first $\mu - |P_{t+1}|$ survive

8.3.2 S-Metric Selection EMOA

The S-Metric Selection Evolutionary Multi-Objective Algorithm (SMS-EMOA) is another famous algorithm in the field. It was proposed back in 2005 [EBN05] in a first version. The journal extension of the paper introduced another slight

variation of the algorithm. Like in NSGA-II the algorithms major contribution is the combination of two criteria for survival selection. SMS-EMOA imple-

Algorithm 12 SMS-EMOA (variant A) [EBN05; BNE07]

1: Initialise P_0
2: $t \leftarrow 0$
3: **while** Termination condition not met **do**
4: $o \leftarrow$ variation(P_t) // Generate one new solution
5: Let D be the set of dominated individuals from $P_t \cup \{o\}$
6: **if** $D \neq \emptyset$ **then**
7: Let x^* be the individual that is dominated by most individuals in $P_t \cup \{o\}$
8: **else**
9: Let x^* be the individual with the lowest HV-contribution.
10: **end if**
11: $P_{t+1} \leftarrow (P_t \cup \{o\}) \setminus \{x^*\}$
12: $t \leftarrow t + 1$
13: **end while**
14: **return** P_t

ments a steady-state $(\mu + 1)$-strategy, i. e., in each iteration a single offspring is generated by variation. Consequently, selection needs to select μ out of $(\mu + 1)$ individuals. The algorithm starts by calculating the set D of dominated individuals. If D is not empty SMS-EMOA (see Algorithm 12) adopts a simple domination count argument: drop the individual that is dominated by most other individuals (and break ties at random. I. e., if there are several individuals in D with maximal domination count throw a fair coin and throw away one of these individuals). If $D = \emptyset$, this approach does not make sense since the domination count is zero for all points. Here, SMS-EMOA calculates the *dominated Hypervolume contribution* for each of the $(\mu + 1)$ solutions and drops the one that contributes least to the overall dominated Hypervolume. The dominated hypervolume contribution of point x is is exclusive area that is added to the total dominated hypervolume by x (see Figure 8.8 for an illustration). This is a striking idea since the Hypervolume favours both convergence and spread and has very desirable properties (see Section on performance measurement).
Note that in a typical run of SMS-EMOA, in the first iterations, D most likely will be non-empty. The Hypervolume plays a role once the algorithm made some progress towards to Pareto-set/front. The second variant of the algorithm (see Algorithm 13) pretty much replaces crowding distance in favor of Hypervolume contribution in NSGA-II. In combination with the $(\mu + 1)$-approach this results in an elegant algorithm 11). Note that since only one offspring is generated per iteration and the last non-domination layer F_k, $k \geq 1$ is non-empty the NSGA-II loop to "fill up" the next population (lines 8-11 in Algorithm 10) is obsolete.

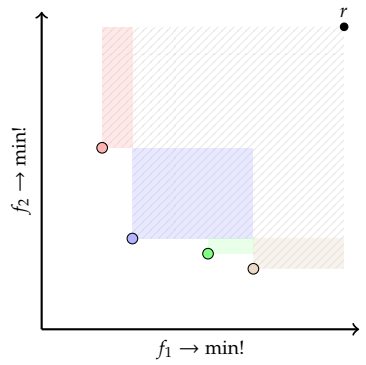


Figure 8.8: Dominated hypervolume spanned by a reference point r and three points. The colored regions are correspond to the exclusive hypervolume contributions of the correspondingly colored points. SMS-EMOA 12 drops the point with the smallest contribution (here, the green point).

Algorithm 13 SMS-EMOA (variant B) [BNE07]

```
1: Initialize P_0
2: t ← 0
3: while Termination condition not met do
4:     o ← variation(P_t)                    // Generate one new solution
5:     F_1, ..., F_k ←NDS(P_t ∪ {o})
6:     Let x* be the individual with the lowest HV-contribution in F_k
7:     P_{t+1} ← (P_t ∪ {o}) \ {a*}
8:     t ← t + 1
9: end while
10: return P_t
```

Instead, the algorithm just calculates the Hypervolume contributions for the points in F_k and deletes the individual with the smallest contribution. SMS-EMOA is a highly effective algorithm. For large m however, the Hypervolume calculation becomes the crucial aspect as it grows in the order of $O(n^{m/2}\log(n))$ for n points and m objectives. For $m = 2$ this is $O(n \log n)$ which is perfectly fine. For, say $m = 6$, we end up with super-cubic runtime which may become a bottleneck if n is large.

8.4 Exercises

Exercise 1 (Non-domination)
Consider $n = 10$ points in $\mathbb{R}^2$ and assume we aim to minimise both objectives. Sketch a set of points such that there is

- exactly one non-domination level,

- two non-domination levels with one point on the first and $n-1$ points on the second level,
- exactly n non-domination levels.

Exercise 2 (Non-dominated sorting implementation)
Implement a function `nds(x)`. The function expects a numeric matrix `x` (each column is a point in objective space). The output is an integer vector whose length equals the number of columns of `x`. The ith position is the non-domination level of the ith point (we assume minimisation of all objectives). To check you implementation for correctness visually, plot the `mtcars` variables hp and gas and colour the point by their respective level.

Exercise 3 (Optimisation by enumeration ★)
For $\mathcal{X} = \{0, \dots, 50\}^2 \subseteq \mathbb{N}^2$ let

$$f : \mathcal{X} \to \mathbb{R}^2, f(x) := \left(1 - \sqrt{\frac{x_1}{50+9x_2}} - \left(\frac{x_1}{50+9x_2}\right)^{\frac{x_1}{50}} \cdot \sin\left(\frac{\pi \cdot x_1}{5}\right)\right) \to \text{min!}.$$

Calculate the Pareto-set numerically, i.e., write a program which calculates $f(x)$ for each $x \in \mathcal{X}$ and checks for dominance. Plot the whole decision and objective space; in both plots highlight the non-dominated points.

Exercise 4 (Weighted-sum scalarisation: Pareto-optimality ★)
Let $f : \mathcal{X} \to \mathbb{R}^m, m \geq 2$ be a multi-objective function with all objectives to be minimised. Assume that $\mathcal{X}$ is n-dimensional. Let $\lambda_1, \dots, \lambda_n \in [0, 1]$ such that $\sum_{i=1}^n \lambda_i = 1$. Consider the weighted-sum function $f_\lambda(x) = \sum_{i=1}^n \lambda_i \cdot f_i(x)$ and let x^* be a global optimum of f_λ. Show that x^* is Pareto-optimal for f.
Hint: prove the statement by contradiction.

Exercise 5 (Weighted-sum scalarization ★)
Consider the bi-objective problem $f : [-4, 4]^2 \to \mathbb{R}^2$ with $f(x) = (f_1(x), f_2(x)) \to$ min! with

$$f_1(x) := 1 - \exp(-(x_1 - 1)^2 - (y - 1)^2)$$
$$f_2(x) := 1 - \exp(-(x_1 + 1)^2 - (y + 1)^2).$$

Use the weighted sum approach with $\lambda_1 \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$, $\lambda_2 = 1 - \lambda_2$ to convert the bi-objective problem into single-objective problems. For each λ_1, plot the single-objective problem in 3D plot or by means of a contour plot (or both). In addition, solve the single-objective problem with the Nelder-Mead algorithm (in R you can use the `optim` function setting `method="Nelder-Mead"`;

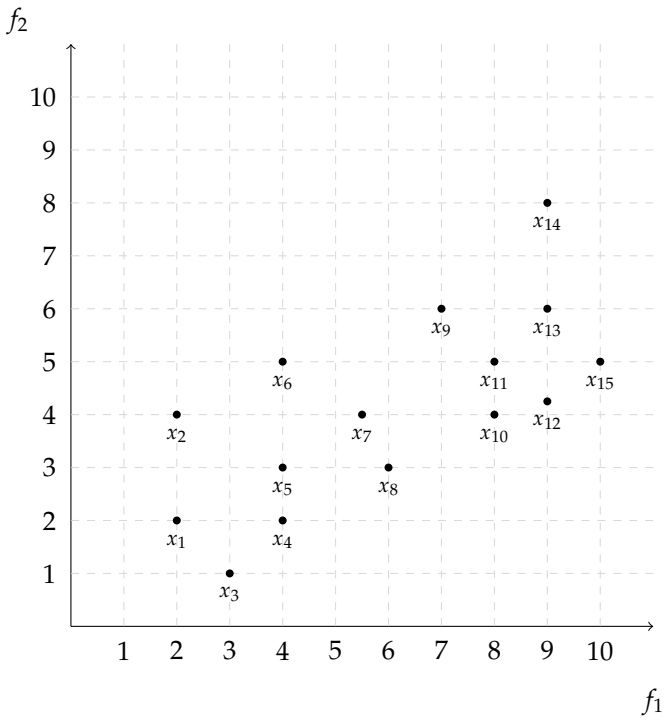


Figure 8.9: Set of 15 points in two-dimensional objective space.

see help page for further details). Take a look at the results and come up with an idea which condition Pareto-optimal points need to satisfy.

Exercise 6 (SMS-EMOA (exercise from DA1 exam in WT21/22))
Figure 8.9 above shows a set of $n = 15$ points in the two-dimensional objective space. Assume that goal is to maximize objective function f_1 and to simultaneously minimize f_2.

1. Apply *non-dominated sorting* (NDS 11) to the points and split them into domination fronts/layers. Explain your actions.
2. Assume that the points in the figure are a snapshot of a (14 + 1) SMS-EMOA (variant B from the lecture notes; see Algorithm 13) and we need to select 14 out of the 15 points for survival. Perform the necessary HV-contribution calculations by hand using $r = (1, 10)^T$ as reference point. Note that reporting just the results is not enough! In addition, sketch the HV-contributions for the points in a new scatter-plot (i. e., draw your own scatter-plot). Which point will be dropped? Explain your actions.
3. Assuming minimization of both objectives sketch a set of 6 points and a reference point r in a scatter-plot such that the HV-contribution is the same for all 6 points.

Exercise 7 (I) n the lecture we learned about different multi-objective performance measures. Consider the following measure: given an approximation set $A \subseteq \mathbb{R}^p$ and a reference set $R \subseteq \mathbb{R}^p$ it is defined as

$$m(A, R) := \min_{x \in A} d(x, R)$$

where $d(x, R)$ is the distance of x to its nearest-neighbour in R. Describe what the measure does in your own words. Furthermore discuss whether this is a good or bad measure.
Hint: you may also add visualisations.

Literature

[Deb+02] K. Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017.

[EBN05] Michael Emmerich, Nicola Beume, and Boris Naujoks. "An EMO Algorithm Using the Hypervolume Measure as Selection Criterion". In: *Evolutionary Multi-Criterion Optimization*. Ed. by Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 62–76.

[BNE07] Nicola Beume, Boris Naujoks, and Michael Emmerich. "SMS-EMOA: Multiobjective selection based on dominated hypervolume". In: *European Journal of Operational Research* 181.3 (2007), pp. 1653–1669. DOI: https://doi.org/10.1016/j.ejor.2006.08.008.

[ES15] Ágoston E Eiben and James E Smith. *Introduction to Evolutionary Computing*. Springer, 2015. DOI: 10.1007/978-3-662-44874-8.

[Hor+15] Daniel Horn et al. "Model-Based Multi-objective Optimization: Taxonomy, Multi-Point Proposal, Toolbox and Benchmark". In: *Evolutionary Multi-Criterion Optimization*. Ed. by António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos Coello Coello. Vol. 9018. Lecture Notes in Computer Science. Springer, 2015, pp. 64–78.

Appendix A

Mathematical Foundations

Experience shows that students attending Data Analytics 1 are very heterogeneous and bring very different levels of mathematical background. Even though we will not need much mathematics it is a good idea to introduce very basic concepts here. We will cover sets, summations, basic concepts of linear algebra and statistics. Even if you are familiar with all this it is a good idea to quickly read the following just to familiarise yourself with the notation used throughout this manuscript.

A.1 Sets

Definition A.1 (Set[1])**.** A *set* is a *collection of distinguishable objects*. We call these objects the *elements* of the set.

Objects of a set are enumerated within curly braces: $A = \{5, 3, 6\}$. The term "distinguishable" implies that duplicates are not allowed (i.e., there are no copies of an element). We write $x \in A$ if x is an element of A and $x \notin A$ otherwise. The *cardinality* of a set A is simply the number of elements of A and is denoted $|A|$ or $\#A$. The empty set $\emptyset$ or sometimes $\{\}$ has no elements at all, thus $|\emptyset| = 0$ while $|\{5, 3, 6\}| = 3$.
There are different ways to fill a set with elements. *Explicit construction* just exhaustively enumerates all elements of a set, say $A = \{1, 2, 5\}$. For obvious reasons this is not practicable for large sets. *Implicit construction* allows to drop elements if it is clear enough for the reader how the elements look like:

$$\mathbb{N} = \{1, 2, 3, \ldots\} \quad \text{(natural numbers)}$$
$$\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\} \quad \text{(integer numbers)}.$$

[1] Intuitive definition of a set according to George Cantor.

We can go even a step further and describe the properties of the set:

$$\begin{aligned}\mathbb{N}_{2k} &= \{n \in \mathbb{N} \mid n \text{ is even}\} = \{2n \mid n \in \mathbb{N}\} \quad \text{(even natural numbers)}\\ \mathbb{Q} &= \{p/q \mid p \in \mathbb{Z}, q \in \mathbb{N}\} \quad \text{(rational numbers)}\\ A &= \{x \in \mathbb{N} \mid x \text{ is even and a multiple of } 8\}.\end{aligned}$$

It is useful to inspect relations between sets. For two sets A and B we say that A *is a subset of* B, $A \subseteq B$, if $x \in A$ implies $x \in B$. In this case B is called a *super-set of* A. This definition allows for equality of the sets. A is a *strict* subset of B, denoted as $A \subset B$, if $A \subseteq B$, but there is at least one element in B that is not in A. A and B are equal, if $A \subseteq B$ and $B \subseteq A$.

Definition A.2 (Set operations)**.** Let A and B be two sets. Then

$$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$$

is the *intersection of A and B*,

$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$$

is the *union of A and B* and

$$A \setminus B = \{x \mid x \in A \text{ and } x \notin B\}$$

is the *set difference of A and B*.

With set operations we can build new sets by intermingling two or more other sets. Intersection and union can be naturally extended to more than two sets, e.g., for n sets $A_1, \ldots, A_n$ we define

$$\bigcap_{i=1}^{n} A_i := A_1 \cap A_2 \cap \ldots \cap A_n \text{ and } \bigcup_{i=1}^{n} A_i := A_1 \cup A_2 \cup \ldots \cup A_n.$$

If $A \cap B = \emptyset$ we call A and B *disjoint*.
Basic operations can be visualised by means of Venn-diagrams. Here, for n sets we draw n circles such that all circles have a common overlap. This way we can nicely visualise certain relation and operations (see Fig. A.1
Oftentimes all sets that we consider in the context of problem are subsets of a larger set denoted the *ground set* or *universe* U. Given U and a set A we can define the complement via the set difference.

Definition A.3 (Complement)**.** Let U be a universe and $A \subseteq U$. Then, the

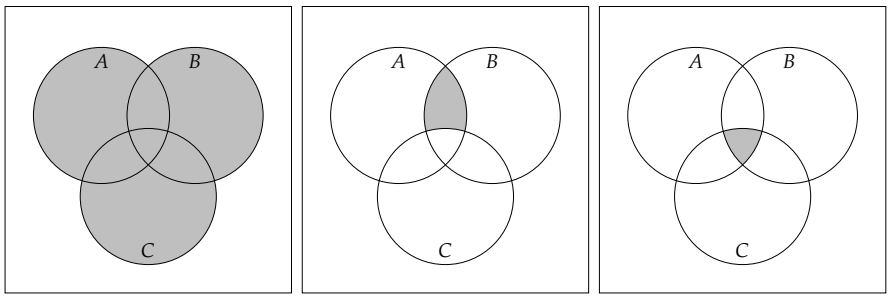


Figure A.1: From left to right: $A \cup B \cup C$, $(A \cap B) \setminus C$ and $A \cap B \cap C$

complement of A (with respect to U) is defined as

$$\bar{A} := U \setminus A = \{x \mid x \in U \text{ and } x \notin A\}.$$

So, given $U = \mathbb{Z}$ the complement of $\mathbb{N}$ with respect to $\mathbb{Z}$ is all integer numbers smaller than or equal to zero.
The power set is the set of all sub-sets and will be important in discrete optimisation:

Definition A.4 (Power set)**.** Let A be a set. The *power set* 2^A is *the set of all subsets of A*, i.e.

$$2^A := \{X \mid X \subseteq A\}.$$

It holds that $|2^A| = 2^{|}A|$.

For example: the power-set of $A = \{1, 2, 3\}$ is

$$2^A = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, A\}$$

. The cardinality of is $|2^A| = 8$.

Definition A.5 (k-partition)**.** A *k-partition* of a set A is a decomposition of A into $k > 0$ *non-empty* subsets $A_1, \ldots, A_k$ such that the following holds:

1. $A_i \cap A_j = \emptyset$ for $1 \leq i \neq j \leq k$, i.e., the subsets are pairwise disjoint.
2. $\bigcup_{i=1}^{k} A_i = A$, i.e., the union of all the subsets is A itself. We say that the partition *covers* A.

E.g., for $A = \{1, 4, 6, 7, 18, 19, 25\}$ the subsets $A_1 = \{1, 25\}$ and $A_2 = \{4, 6, 7, 18, 19\}$ define a 2-partition while $B_1 = \{1\}, B_2 = \{25, 19\}$, $B_3 = \{4, 6, 7, 18\}$ is a 3-partition of A. However, $C_1 = \{1, 4, 6, 7\}, C_2 = \{19, 25\}$ is no 2-partition since the element 18 is neither in C_1 nor in C_2 and as a consequence C_1, C_2 does not

cover A. Obviously, a k-partition defines a decomposition of elements into disjoint groups/clusters. Unsurprisingly, this concept is of interest in the context of clustering in Section 4.

A.2 Summations

Mathematicians are lazy and so are computer scientists as applied mathematicians. Hence, if we have a set/sequence $a_1, a_2, \ldots$ of numbers and we aim to sum up all the elements in the sequence we avoid to write down all elements a la

$$a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + \ldots.$$

Imagine to do so for a set with thousands of elements! Instead there are symbols that make life easier and look absolutely scary for people who do not know the meaning.

Definition A.6 ((In)finite summations)**.** Let $a_1, a_2, \ldots$ be a sequence of numbers. Then

$$\sum_{i=1}^{\infty} a_i := a_1 + a_2 + \ldots$$

is an *infinite summation*. For each $n \in \mathbb{N}$

$$\sum_{i=1}^{n} a_i := a_1 + a_2 + \ldots + a_{n-1} + a_n$$

is a *finite summation*.

We will also make use of some "shortcut"-notation if we want to sum up elements from a finite set. To illustrate how this can look like consider a set $A = \{a_1, \ldots, a_n\}$ of real numbers. Adding up all elements in A can be written as

$$\sum_{i=1}^{|A|} a_i = \sum_{i=1}^{n} a_i = \sum_{x \in A} x.$$

If we want to add up all elements expect for the j^{th} one we can write

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} a_i = \sum_{x \in A \setminus \{a_j\}} x$$

In order to sum up all pairwise absolute differences of elements in A we could use the following notation:

$$\sum_{a \in A} \sum_{b \in A} |a - b| = \sum_{a,b \in A} |a - b|.$$

You see that there are most often multiple ways to describe something. We will use the notation that fits our needs best.

A.3 Linear algebra

The elementary structure in linear algebra is a vector. In data analysis vectors are a welcome structure to gather either observations (rows of a data table) or the columns itself, i.e., sequences of realizations of some variable or feature. In the following we introduce real-valued vectors. However, vectors are certainly not limited to real numbers.

Definition A.7 (Vector)**.** A vector $x \in \mathbb{R}^n$ is an ordered collection of n real values. We can write it as a column vector

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n.$$

Having two vectors $x, y \in \mathbb{R}^n$ and a scalar $c \in \mathbb{R}$ we can create new vectors. Multiplication with a scalar multiplies each component of the vector with c.

$$c \cdot x = \begin{pmatrix} c \cdot x_1 \\ \vdots \\ c \cdot x_n \end{pmatrix}$$

Analogously, we can add or subtract two vectors by applying the respective scalar operation component-wise:

$$x + y = \begin{pmatrix} x_1 + y_1 \\ \vdots \\ x_n + y_n \end{pmatrix} \quad x - y = x + (-1) \cdot y = \begin{pmatrix} x_1 - y_1 \\ \vdots \\ x_n - y_n \end{pmatrix}$$

Fig. A.2 show all three operations by means of example in $\mathbb{R}^2$. Multiplication with a scalar either extends the vector in its direction for $c \geq 1$ or shortens it for

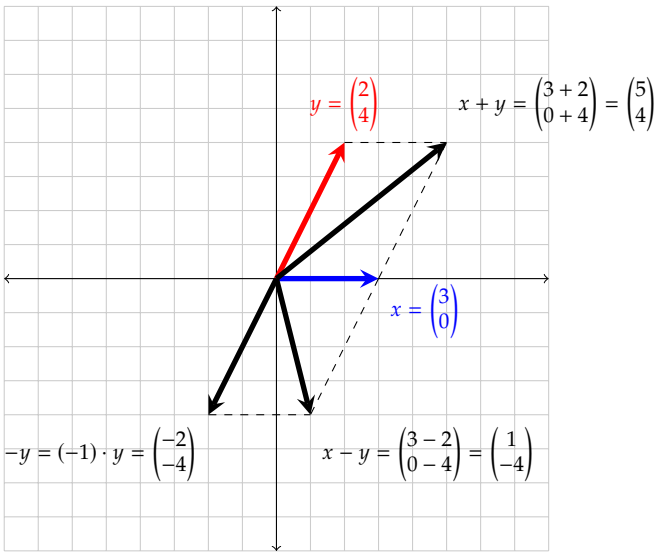


Figure A.2: Geometric interpretation of vector addition, subtraction and multiplication with a scalar.

$0 < c < 1$. A negative value changes the direction. Vector addition (subtraction) works by moving the origin of the second vector in space such that it starts at the arrow tip of the first vector.

Definition A.8 (Length of a vector)**.** The *length* of a vector $x \in \mathbb{R}^n$ is

$$|x| := \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{x_1^2 + \ldots + x_n^2}$$

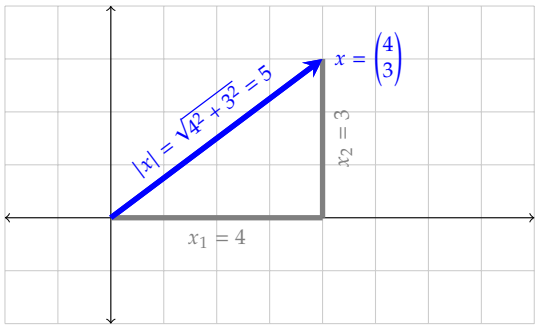


Figure A.3: Illustration of the length of a vector.

Fig. A.3 illustrates the length calculation in the 2-dim. Euclidean space. The length of the vector $(x - y)$ is a quite natural choice for a distance in n-dim.

space termed the *Euclidean distance*:

$$d(x, y) := ||x - y||_2 = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2}.$$

It is the length of the straight line segment between the two points.

Definition A.9 (Matrix)**.** A $(n \times m)$ *matrix* A is a rectangular two-dimensional table of numbers spanning n rows and m columns. We usually denote matrices as follows:

$$X = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1m} \\ x_{21} & x_{22} & \ldots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nm} \end{pmatrix} \sim (n, m).$$

A matrix is a two-dimensional rectangular grid of numbers. Here, x_{ij} is the entry in the i^{th} *row* and j^{th} *column*. $x_{i\cdot} = (x_{i1}, x_{i2}, \ldots, x_{im})$ is the i^{th} *row vector* and $x_{\cdot j} = (x_{1j}, x_{2j}, \ldots, x_{nj})$ is the j^{th} *column vector*. Thus, a matrix can be seen as a set of vectors stacked on top of each other or side by side. Analogous to vectors, we can define multiplication with a scalar and matrix addition for $X, Y \sim (n, m)$ and $c \in \mathbb{R}$ as

$$c \cdot X = \begin{pmatrix} c \cdot x_{11} & \ldots & c \cdot x_{1m} \\ \vdots & \ddots & \vdots \\ c \cdot x_{n1} & \ldots & c \cdot x_{nm} \end{pmatrix}$$

and

$$X + Y = \begin{pmatrix} x_{11} + y_{11} & \ldots & x_{1m} + y_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} + y_{n1} & \ldots & x_{nm} + y_{nm} \end{pmatrix}.$$

In addition it makes sense to define a multiplication of a matrix with a vector. Given a vector $x \in \mathbb{R}^n$ and a "compatible" matrix $A \sim (n, n)$ we can build a new/transformed vector by *matrix vector multiplication*:

$$A \cdot x = \begin{pmatrix} a_{11} \cdot x_1 + a_{12} x_2 + \ldots + a_{1n} \cdot x_n \\ a_{21} \cdot x_1 + a_{22} x_2 + \ldots + a_{2n} \cdot x_n \\ \ldots \\ a_{n1} \cdot x_1 + a_{n2} x_2 + \ldots + a_{nn} \cdot x_n \end{pmatrix}$$

Note that for this to be well defined the number of columns of the matrix needs to match the dimension of the vector. The components of $A \cdot x$ are *linear combinations of the components of* x. This operation describes a transformation

of the input vector x in the n-dimensional space (e.g. stretching, squeezing, rotation).

Let us further investigate rotation by an angle $\theta \in [0, 2\pi]$. The following so-called *rotation matrix* $R(\theta)$ rotates a vector $x \in \mathbb{R}^2$ counterclockwise by θ:

$$R(\theta) := \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \rightsquigarrow R(\theta) \cdot x = \begin{pmatrix} \cos\theta \cdot x_1 - \sin\theta \cdot x_2 \\ \sin\theta \cdot x_1 + \cos\theta \cdot x_2 \end{pmatrix}$$

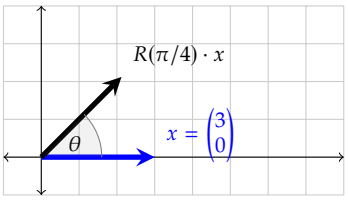


Figure A.4: Rotation of vector in 2D Euclidean space counterclockwise by $\theta = \frac{\pi}{4}$.

This operation is illustrated in Fig. A.4 rotating $x = (3, 0)^T$ by 45 degrees (i.e., $\theta = \frac{\pi}{4}$) counterclockwise. Note that this operation can in fact be undone/reverted by rotating the transformed vector $R(\theta) \cdot x$ clockwise by θ which yields x again. This inverse operation can be realized with the rotation matrix $R^{-1}(\theta) = R(-\theta)$. It holds that

$$\underbrace{R^{-1}(\theta) \cdot R(\theta)}_{=I_2} \cdot x = \underbrace{R(-\theta) \cdot R(\theta)}_{=I_2} \cdot x = I_2 \cdot x = x.$$

where $I_2 = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ is the (2×2) unit matrix. $R^{-1}(\theta)$ is called the inverse matrix of $R(\theta)$.

Definition A.10 (Inverse matrix)**.** A square matrix $A \sim (n, n)$ is invertible if $\det A \neq 0$. In this case there exists a matrix $A^{-1} \sim (n, n)$, the *inverse of* A, such that

$$A \cdot A^{-1} = A^{-1} \cdot A = I_n$$

where I_n is the $(n \times n)$ unit matrix.

The condition $\det A \neq 0$ (determinant unequal zero) says that there is no loss in dimension information by a transformation with A (see below).

Definition A.11 (Determinant)**.** The determinant of a square matrix $A = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \sim$

$(2, 2)$ is

$$\det A = \begin{vmatrix} a & b \\ c & d \end{vmatrix} := ad - cb.$$

For a general square matrix $A \sim (n, n)$ the determinant can be calculated recursively by *developing the j^{th} column*[2]

$$\det A = \sum_{i=1}^{n} (-1)^{i+j} \cdot a_{ij} \cdot \det A_{ij}$$

where A_{ij} is the sub-matrix that we get if we drop the j^{th} column and i^{th} row from A.

This definition is odd and the meaning is not clear. Consider the illustration in Fig. A.5 and in particular the size of the rectangle area spanned by the two vector $x = (3, 0)^T$ and $y = (0, 2)^T$. The area spanned by the transformed vectors Ax and Ay for $A = \left(\begin{smallmatrix} 2 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ is twice as large and it turns out that $\det A = 2$. I.e., the determinant can be seen as the scaling factor by which (arbitrary) shapes are transformed by the linear mapping A. The determinant can be zero. In this case every shape is reduced to zero size which means that the transformation projects into a lower-dimensional space; this is why no inverse matrix exists in such cases (see definition of a matrix).

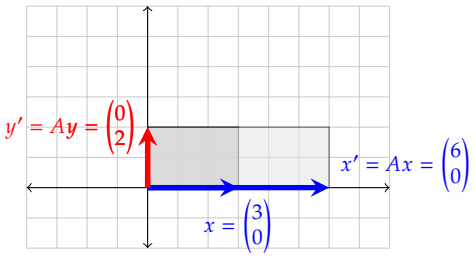


Figure A.5: We have $\text{Area}(x, y) = 6$ and $\text{Area}(Ax, Ay) = 12 = 2 \cdot \text{Area}(x, y)$. With area we mean the area spanned by the two vectors. The scaling factor $\det A = 2$ is the *determinant of the matrix A*.

Finally, we introduce the concept of Eigenvalues and Eigenvectors.

Definition A.12 (Eigenvalues & -vectors)**.** Let $A \sim (n, n)$ be a square matrix. A scalar value λ is called an *Eigenvalue* and a vector $v \neq 0$ is called the corre-

[2] Analogously, we can develop the i^{th} row.

sponding *Eigenvector* if

$$A \cdot v = \lambda \cdot v. \tag{A.1}$$

Recall that a square $(n \times n)$-matrix[3] actually defines a mapping $f_A : \mathbb{R}^n \to \mathbb{R}^n$ with $f_A(v) \mapsto A \cdot v$. I.e., we put in a vector and obtain a transformed vector (e.g., a rotated version of the input). An Eigenvector v of A has, according to Eq. A.1, the special property that it is merely stretched by the transformation as if it was multiplied with a scalar value, the Eigenvalue.

We can easily derive how to calculate the Eigenvalues of A. By definition for $A \sim (n, n)$: $Av = \lambda v$ must hold. Via reformulation we obtain:

$$\begin{aligned} Av = \lambda v \quad &\Leftrightarrow \quad (Av - \lambda v) = 0 \\ &\Leftrightarrow \quad (Av - \lambda I_n v) = 0 \\ &\Leftrightarrow \quad (A - \lambda I_n)v = 0 \\ &\Leftrightarrow \quad \det(A - \lambda I_n) = 0 \end{aligned} \tag{A.2}$$

The Eigenvalues are thus the values of λ that make Eq. A.2 true. The left-hand side of Eq. A.2 is called the *characteristic polynomial (of A)*. It is usually denoted as

$$\chi_A(\lambda) := \det(A - \lambda I_n).$$

As an example let us calculate the Eigenvalues and -vectors of the matrix

$$A = \begin{pmatrix} 1 & 5 \\ 2 & -2 \end{pmatrix}.$$

First we build $A - \lambda I_2$:

$$A - \lambda I_2 = \begin{pmatrix} 1 & 5 \\ 2 & -2 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 - \lambda & 5 \\ 2 & -2 - \lambda \end{pmatrix}$$

Next, we calculate $\chi_A(\lambda)$:

$$\det(A - \lambda I_2) = \begin{vmatrix} 1 - \lambda & 5 \\ 2 & -2 - \lambda \end{vmatrix} = (1 - \lambda)(-2 - \lambda) - 10 = \lambda^2 + \lambda - 12.$$

[3] Not just square matrices, but the square case is sufficient for our needs.

We get the nulls of $\chi_A(\lambda)$ by applying the p/q-formula:

$$\det(A - \lambda I_n) \stackrel{!}{=} 0 \Leftrightarrow \lambda^2 + \lambda - 12 = 0 \Leftrightarrow \lambda \in \{3, -4\}.$$

It is convenient to give the Eigenvalues numbered names, $\lambda_1 = 3, \lambda_2 = -4$ without a specific order.
We now proceed with the calculation of the corresponding Eigenvectors. For $\lambda_1 = 3$ we need to solve the homogeneous linear equations system $(A - \lambda_1 I_2)v = 0$. So let us plug in λ_1 and apply the Gaussian elimination algorithm to simplify the equations:

$$A - \lambda_1 I_2 = \begin{pmatrix} -2 & 5 \\ 2 & -5 \end{pmatrix} \begin{matrix} \urcorner^{1} \\ \llcorner_{+} \end{matrix} \rightsquigarrow \begin{pmatrix} -2 & 5 \\ 0 & 0 \end{pmatrix}.$$

Let $v_2 \in \mathbb{R} \setminus \{0\}$. Then the second equation is true and from the first equation we get $v_1 = 5/2v_2$. Thus the sub-space of Eigenvectors for $\lambda_1 = 3$ is

$$\sigma(\lambda_1) = \left\{ \begin{pmatrix} 5/2v_2 \\ v_2 \end{pmatrix} \middle| \, v_2 \in \mathbb{R} \setminus \{0\} \right\}.$$

For $\lambda_2 = -4$ with similar calculations we get (check the details)

$$A - \lambda_2 I_2 = \begin{pmatrix} 5 & 5 \\ 2 & 2 \end{pmatrix} \; | \cdot 1/2 \rightsquigarrow \begin{pmatrix} 5 & 5 \\ 1 & 1 \end{pmatrix} \begin{matrix} \llcorner^{+} \\ \urcorner_{-5} \end{matrix} \rightsquigarrow \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$$

and consequently

$$\sigma(\lambda_2) = \left\{ \begin{pmatrix} v_1 \\ -v_1 \end{pmatrix} \middle| \, v_1 \in \mathbb{R} \setminus \{0\} \right\}.$$

You should convince yourself that the Eigenvectors are indeed correct by checking if Eq. A.1 holds.

A.4 Statistics

We are dealing with data in Data Analytics 1. Hence, some elementary knowledge on statistics is unavoidable. For motivation consider the experiment of throwing a regular fair dice. I.e., each of the six number of eyes is equally likely. Holding the dice in our hand we do not know the outcome. Hence, we model the experiment as a *random variable (RV)* X which counts the number of eyes. The possible outcomes are in $\mathcal{T}_X = \{1, \ldots, 6\}$. We call this set the *support set of*

X. The dice is fair. I.e. for the probability it is reasonable to assume

$$\Pr(X = x) = 1/6 \text{ for all } x \in \mathcal{T}_X.$$

If we now actually throw the dice we obtain a *realisation* of the RV which we call *observation*.[4]

Now that we have modelled our experiments we can inspect theoretical statements and empirical statements based on observations. We might ask the questions: what is the *average* number of eyes that we can expect when throwing the dice (performing the experiment)? Empirically this is easy to answer: repeat the experiment n times, collect results $x_1, \ldots, x_n$ and calculate the (arithmetic) mean $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. E.g. for $x_1 = 6, x_2 = 2, x_3 = 4$ we would get $\bar{x} = \frac{1}{3}(6 + 2 + 4) = 4$. However, it also makes sense to think of the theoretical average value which we could implement as the sum of all possible outcomes where each outcome is weighted with its theoretical probabilities 1/6, i.e., $\bar{X} = \frac{1}{6}(1 + 2 + \ldots + 6) = 3.5$. The reason why this actually makes sense is the so-called *law of large numbers* which hand-wavy states that if we repeat an experiment often, i.e., $n \to \infty$, the relative frequencies h_i of the outcomes $i \in \mathcal{T}_X$ will converge against the probabilities $\Pr(X = i)$. In consequence the arithmetic mean converges against its theoretical pendant since

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} h(i) \cdot i.$$

We now give formal definitions of theoretical measures (of center and variation) and mentions their empirical versions. Adopting common notation we use capital letters for random variables and small-case letters for realisations.

The expected value is the most common measure of mass and corresponds to the weighted sum we used in our introductory example:

Definition A.13 (Expected value / expectation)**.** Let X be a (discrete) random variable that takes values from a set $\mathcal{T}_X = \{x_1, x_2, \ldots\}$. Then the *expected value* or *expectation* of X is the sum of the values each weighted with its probability:

$$\mu_X = E(X) := \sum_{x \in \mathcal{T}_X} x \cdot \Pr(X = x).$$

The empirical version is the *arithmetic mean* $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

Theorem A.1. *For a discrete random variable X and $a, b \in \mathbb{R}$ linearity of expectation*

[4] Hence, actual data sets are collections of realisations of RVs.

holds:

$$E(aX + b) = aE(x) + b.$$

Proof. The proof is pretty simple:

$$\begin{aligned} E(aX + b) &= \sum_{x \in \mathcal{T}_X} (ax + b) \cdot \Pr(X = x) \\ &= a \cdot \underbrace{\sum_{x \in \mathcal{T}_X} x \Pr(X = x)}_{=E(X)} + b \cdot \underbrace{\sum_{x \in \mathcal{T}_X} \Pr(X = x)}_{=1} \\ &= aE(X) + b. \end{aligned}$$

□

Based on the expected value the variance is a measure for deviation. It is defined as the average squared deviation of the outcomes to the expected value.

Definition A.14 (Variance)**.** Let X be a (discrete) random variable that takes values from a set $\mathcal{T}_X = \{x_1, x_2, \ldots\}$ with finite expectation $E(X)$. Then the *variance of* X is the expected squared deviation from the expected value

$$\sigma_X^2 = \text{Var}(X) = E\left[(X - E(X))^2\right] = \sum_{x \in \mathcal{T}_X} (x - E(X))^2 \cdot \Pr(X = x).$$

By definition, $\text{Var}(X) \geq 0$ even though $\text{Var}(X) = 0$ does not make much sense (Why?). Due to the square the variance is on another scale than the outcomes of X. Thus, often the square rooted version $\sigma_X = \sqrt{\text{Var}(X)}$, the *standard deviation*, is used. The empirical version of the variance is the average squared deviation of the observations to the arithmetic mean $s^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$. The variance is not linear; not surprising since we square the deviations. However,

Theorem A.2. *For a discrete random variable X and $a, b \in \mathbb{R}$ we have*

$$Var(aX + b) = a^2 \cdot Var(X).$$

I.e., a shift has no effect at all and the factor a can be factored out with an additional power of two (try to proof the theorem).
Expectation and variance describe single random variables. To measure interactions of two variables X and Y we need a different idea.

Definition A.15 (Covariance)**.** Let X and Y be two random variables. Then

$$\text{Cov}(X, Y) := E\left[(X - E(X)) \cdot (Y - E(Y))\right]$$

is the *covariance* of X and Y.

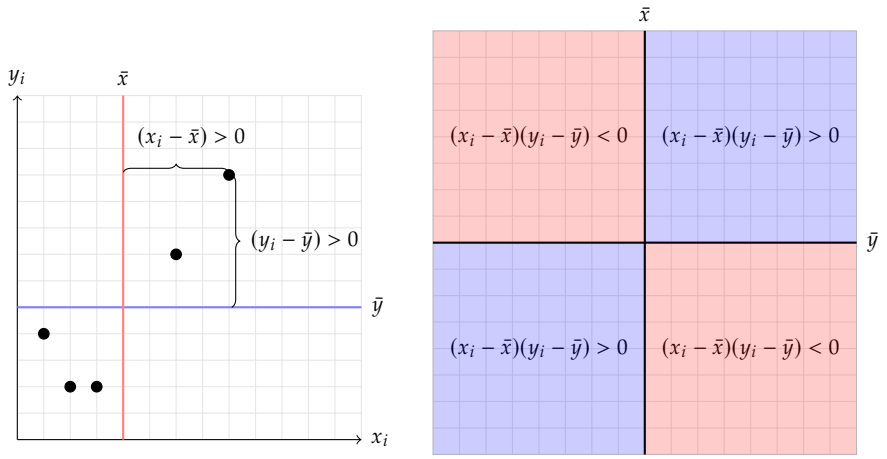


Figure A.6: Illustration of the summands in the (empirical) covariance (left) and the effect of the location of outcomes/observations on the summands' sign (right).

The empirical analogue is the *empirical covariance* $s_{xy} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x}) \cdot (y_i - \bar{y})$. A close look at the definition reveals that it is quite similar to the variance in the uni-variate case. In fact $\text{Cov}(X, X) = \text{Var}(X)$. However, for $X \neq Y$ we multiply non-squared deviations of both variables outcomes. I.e., the direction of the deviation(s) actually matters here! The left-hand side of Fig. A.6 illustrates one summand in the calculation of the empirical covariance with some exemplary observations. The right-hand side highlights the four regions/areas separated by the respective mean values where the summands have different signs. So, if $\text{Cov}(X, Y) > 0$, a positive linear relationship is given: high values of X go hand in hand with high values of Y and low values of X with low values of Y. If $\text{Cov}(X, Y) < 0$, a negative linear relationship holds: high values of X go hand in hand with low values of Y and low values of X with high values of Y. The covariance is a measure of linear relationship. $\text{Cov}(X, Y) = 0$ indicates no linear relationship. However, keep in mind, that there might be a (perfect) non-linear relationship (e.g., imagine something like $Y = X^3$) which cannot be detected by the covariance.

For a meaningful interpretation it would be nice to have a bounded measure, e.g., something that takes values between -1 and 1 only with 1 indicating perfect positive linear relationship and -1 perfect negative linear relationship. The *correlation* is this kind of measure. It achieves this by normalising the covariance.

Definition A.16 (Correlation)**.** Let X and Y be two random variables. Then

$$\text{Cor}(X, Y) := \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)} \cdot \sqrt{\text{Var}(Y)}} \in [-1, 1].$$

is the *correlation* between X and Y.

For multiple random variables $X_1, \ldots, X_p$ it is a convenient to store the pairwise co-variances $\text{Cov}(X_i, X_j), 1 \leq i, j \leq p$ in a matrix.

Definition A.17 (Covariance matrix)**.** Let $X = (X_1, \ldots, X_p)^T$ be a vector of random variables each with finite expected value and variance. The the *covariance matrix* $\text{Cov}(X)$ of X is a square $(p \times p)$ matrix where the components describe the covariance between pairs of variables. I.e., for $1 \leq i, j \leq p$:

$$\text{Cov}(X)_{ij} = \text{Cov}(X_i, X_j) = E\left[(X_i - E(X_i)) \cdot (X_j - E(X_j))\right].$$

The correlation matrix $\text{Cor}(X)$ is defined analogously.

A.5 Exercises

Exercise 1 (Sarrus rule)
Actually, there is a nice rule on how to get the determinant of a (3×3)-matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

without the need for the recursive formula. The *Sarrus rule* says:

$$\det A = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33}.$$

Proof this statement (Hint: use the recursive formula, simplify and rearrange).

Exercise 2 (Eigenvalues & -vectors of a symmetric matrix)
Calculate the Eigenvalues and Eigenvectors of the symmetric matrix

$$A = \begin{pmatrix} 2 & 3 \\ 3 & 2 \end{pmatrix}.$$

For symmetric matrices the Eigenvectors of different Eigenvalues are orthogonal to each other. Can you use this fact to determine the Eigenvectors for the second Eigenvalue without explicitly solving $(A - \lambda_2 I_2)v = 0$?

Exercise 3 (Steiner theorem)
Proof Steiner's theorem for discrete discrete[5] random variables. I.e., show that for a random variable X with expected value $E(X)$ it holds that

$$\text{Var}(X) = E((X - E(X))^2) = E(X^2) - E(X)^2.$$

Exercise 4 (Expected value)
In this exercise we consider a discrete random variable X with support set $\mathcal{T}_X$ and $\Pr(X = x) = 1/|\mathcal{T}_X|$ for all $x \in \mathcal{T}_X$.

1. Show that
$$E(X) = \frac{\sum_{x \in \mathcal{T}_X}}{|\mathcal{T}_X|}.$$
2. Show that for the special case $\mathcal{T}_X = \{1, 2, \ldots, n\}, n \in \mathcal{N}$ the expected value is
$$E(X) = \frac{n+1}{2}.$$

[5] The theorem also holds for continuous random variables.

Appendix B

Runtime of Algorithms

A thorough introduction into analysis of algorithms, algorithmics and algorithm engineering is certainly out of scope. We refer the interested reader to the standard text book *Introduction to Algorithms, 3rd edition* by Cormen, Leiserson, Rivest and Stein [**CormenLRS2009introAlgorithhms**]. Here, we just state the very basics which are necessary to understand the runtime of different machine learning algorithms.

The runtime of algorithms is usually measured by the number of elementary operations (e.g., basic arithmetic like addition, assignments or comparisons to name a few). The reason is that this measure is independent of the machine, the programming language, the skill-set of the programmer etc. Given an algorithm A we are interested in the running time $T_A(n) : \mathbb{N} \rightarrow \mathbb{R}^+$ where n is the problem-dependent size of the problem, e.g., the number of observations N in a data set, the number of numbers to sort for a sorting problem, the number of nodes in a graph for graph-based algorithm etc. Often it also makes sense to measure the time on basis of multiple input parameter (say, N and the number of variables p for data sets, the number of nodes and the number of edges in the context of graphs). However, for the following introduction we will stick to a single input parameter to keep it simple and focus on the essentials. How do we determine the runtime of an algorithm? Well, the simple answer is: we count. This could yield and exact function like $T_A(n) = 10n^2 - 4n + 19$. In certain situations this is fine, but way to complicated in the general case. Here, we are interested in the order of growth only. For polynomials like $T_A(n) = 10n^2 - 4n + 19$ the quadratic term $10n^2$ will certainly dominate for large-enough – and interesting – input sizes n. In addition, the constant 10 just scales the term, but does not determine the order of growth which is purely n^2. If the algorithm runs in time at most order of n^2 for each input of length n we say that its runtime is *asymptotically upper bounded* and denote this by writing

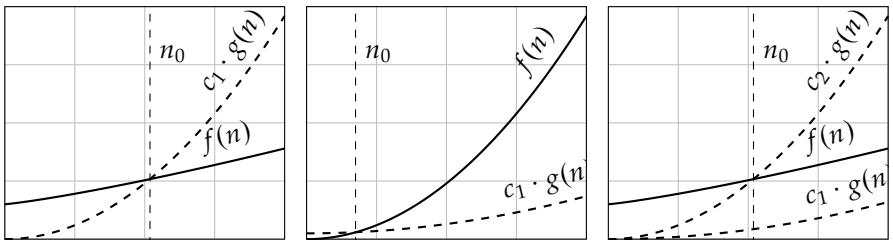


Figure B.1: Illustration of asymptotic upper bound (left), lower bound (center) and tight bound (right).

$T_A(n) = O(n^2)$. This notation is made rigorous in the following definition.

Definition B.1 (O-notation, asymptotic upper bound)**.** A function $f : \mathbb{N} \to \mathbb{R}^+$ belongs to the complexity class $O(g(n))$, if there is a constant $c_1 \in \mathbb{R}^+$ and a $n_0 \in \mathbb{N}$, such that $f(n) \leq c_1 \cdot g(n)$ holds for all $n \geq n_0$.

This means, that $T_A(n) = O(g(n))$ if for all input sizes n larger than some threshold n_0 the running time of A is upper bounded by some constant c_1 times the function $g(n)$. E.g., lets say $T_A(n) = 5n^2 + n$. For all $n \geq n_0 = 1$ we have $T_A(n) = 5n^2 + n \leq 10n^2$. Hence, setting $c_1 = 10$, we see $T_A(n) = O(n^2)$.
Analogously, we can define lower bounds and tight bound which combine lower and upper bounds.

Definition B.2 (Ω-notation, asymptotic lower bound)**.** A function $f : \mathbb{N} \to \mathbb{R}^+$ belongs to the complexity class $\Omega(g(n))$, if there is a constant $c_2 \in \mathbb{R}^+$ and a $n_0 \in \mathbb{N}$, such that $f(n) \geq c_2 \cdot g(n)$ holds for all $n \geq n_0$.

Definition B.3 (Θ-notation, asymptotic tight bound)**.** A function $f : \mathbb{N} \to \mathbb{R}^+$ belongs to the complexity class $\Theta(g(n))$, if there is are two constants $c_1, c_2 \in \mathbb{R}^+$ and a $n_0 \in \mathbb{N}$, such that $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ holds for all $n \geq n_0$.

Figure B.1 illustrates these ideas. For upper/lower bounds the graph of the function $f(n)$ for all $n \geq n_0$ is always below/above a constant times $g(n)$. For the tight bound the graph of $f(n)$ is always "sandwiched" between $c_1 \cdot g(n)$ and $c_2 \cdot g(n)$.

Literature

[Dar59] Charles Darwin. *On the Origin of Species by Means of Natural Selection*. or the Preservation of Favored Races in the Struggle for Life. London: Murray, 1859.

[Mac67] J. B. MacQueen. "Some Methods for Classification and Analysis of MultiVariate Observations". In: *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*. Ed. by L. M. Le Cam and J. Neyman. Vol. 1. University of California Press, 1967, pp. 281–297.

[Ran71] William M. Rand. "Objective Criteria for the Evaluation of Clustering Methods". In: *Journal of the American Statistical Association* 66.336 (1971), pp. 846–850. DOI: 10.1080/01621459.1971.10482356.

[Gra72] R.L. Graham. "An efficient algorithm for determining the convex hull of a finite planar set". In: *Information Processing Letters* 1.4 (1972), pp. 132–133. DOI: https://doi.org/10.1016/0020-0190(72)90045-2.

[Sib73] R. Sibson. "SLINK: An optimally efficient algorithm for the single-link cluster method". In: *The Computer Journal* 16.1 (Jan. 1973), pp. 30–34. DOI: 10.1093/comjnl/16.1.30.

[Dun74] J. C. Dunn. "Well-Separated Clusters and Optimal Fuzzy Partitions". In: *Journal of Cybernetics* 4.1 (1974), pp. 95–104. DOI: 10.1080/01969727408546059.

[Def77] D. Defays. "An efficient algorithm for a complete link method". In: *The Computer Journal* 20.4 (Jan. 1977), pp. 364–366. DOI: 10.1093/comjnl/20.4.364.

[Tuk77] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[DB79] David L. Davies and Donald W. Bouldin. "A Cluster Separation Measure". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2 (1979), pp. 224–227. DOI: 10.1109/TPAMI.1979.4766909.

[Haw80] D. M. Hawkins. *Identification of outliers*. Monographs on applied probability and statistics. Chapman and Hall, 1980.

[Llo82] S. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489.

[Cha+83] J.M. Chambers et al. "Graphical Methods for Data Analysis". In: *The Wadsworth Statistics/Probability Series. Boston, MA: Duxury* (1983).

[Rou87] Peter J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. DOI: https://doi.org/10.1016/0377-0427(87)90125-7.

[Rub87] D. B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. Wiley, 1987, p. 258.

[Est+96] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: AAAI Press, 1996, pp. 226–231.

[RR96] Ida Ruts and Peter J. Rousseeuw. "Computing Depth Contours of Bivariate Point Clouds". In: *Comput. Stat. Data Anal.* 23.1 (1996), 153–168. DOI: 10.1016/S0167-9473(96)00027-8.

[ST96] Nathaniel Schenker and Jeremy M.G. Taylor. "Partially parametric techniques for multiple imputation". In: *Computational Statistics & Data Analysis* 22.4 (1996), pp. 425–446. DOI: https://doi.org/10.1016/0167-9473(95)00057-7.

[JSW98] Donald R. Jones, Matthias Schonlau, and William J. Welch. "Efficient Global Optimization of Expensive Black-Box Functions". In: *Journal of Global Optimization* 13.4 (1998), pp. 455–492.

[Lec+98] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[Jon01] D. R. Jones. "A taxonomy of global optimization methods based on response surfaces". In: *Journal of Global Optimization* 21.4 (2001), pp. 345–383.

[TGH01] Robert Tibshirani, Walther Guenther, and Trevor Hastie. "Estimating the Number of Clusters in a Data Set via the Gap Statistic". In: *Journal of the Royal Statistical Society Series B* (2001).

[Deb+02] K. Deb et al. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017.

[HR03] Geoffrey Hinton and Sam Roweis. “Stochastic Neighbor Embedding”. In: *Advances in neural information processing systems* 15 (2003). Ed. by S Thrun S Becker and KEditors Obermayer, pp. 833–840.

[EBN05] Michael Emmerich, Nicola Beume, and Boris Naujoks. “An EMO Algorithm Using the Hypervolume Measure as Selection Criterion”. In: *Evolutionary Multi-Criterion Optimization*. Ed. by Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 62–76.

[BNE07] Nicola Beume, Boris Naujoks, and Michael Emmerich. “SMS-EMOA: Multiobjective selection based on dominated hypervolume”. In: *European Journal of Operational Research* 181.3 (2007), pp. 1653–1669. DOI: https://doi.org/10.1016/j.ejor.2006.08.008.

[FSK08] Alexander I. J. Forrester, Andras Sobester, and Andy J. Keane. *Engineering Design via Surrogate Modelling - A Practical Guide*. Wiley, 2008, pp. I–XVIII, 1–210.

[KSZ08] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. “Angle-Based Outlier Detection in High-Dimensional Data”. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’08. Las Vegas, Nevada, USA: Association for Computing Machinery, 2008, 444–452. DOI: 10.1145/1401890.1401946.

[MH08] Laurens van der Maaten and Geoffrey Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

[SB08] Juned Siddique and Thomas R. Belin. “Multiple imputation using an iterative hot-deck with distance-based donor selection.” In: *Statistics in medicine* 27 1 (2008), pp. 83–102.

[Alo+09] Daniel Aloise et al. “NP-hardness of euclidean sum-of-squares clustering”. In: *Machine Learning* 75.2 (Jan. 2009), pp. 245–248. published.

[Wic09] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009.

[End10] C.K. Enders. *Applied Missing Data Analysis*. Methodology in the social sciences. Guilford Publications, 2010.

[Bur11] P. Burns. *The R Inferno*. Lulu Com, 2011.

[MNV12] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. "The planar k-means problem is NP-hard". In: *Theoretical Computer Science* 442 (2012). Special Issue on the Workshop on Algorithms and Computation (WALCOM 2009), pp. 13–21. DOI: https://doi.org/10.1016/j.tcs.2010.05.034.

[Wic14] H. Wickham. *Advanced R*. Chapman & Hall/CRC The R Series. Taylor & Francis, 2014.

[ES15] Ágoston E Eiben and James E Smith. *Introduction to Evolutionary Computing*. Springer, 2015. DOI: 10.1007/978-3-662-44874-8.

[GT15] Junhao Gan and Yufei Tao. "DBSCAN Revisited: Mis-Claim, Un-Fixability, and Approximation". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. SIGMOD '15. New York, NY, USA: Association for Computing Machinery, 2015, 519–530. DOI: 10.1145/2723372.2737792.

[Hor+15] Daniel Horn et al. "Model-Based Multi-objective Optimization: Taxonomy, Multi-Point Proposal, Toolbox and Benchmark". In: *Evolutionary Multi-Criterion Optimization*. Ed. by António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos Coello Coello. Vol. 9018. Lecture Notes in Computer Science. Springer, 2015, pp. 64–78.

[Wic15] Hadley Wickham. *R Packages*. 1st. O'Reilly Media, Inc., 2015.

[Bis+16] Bernd Bischl et al. "mlr: Machine Learning in R". In: *Journal of Machine Learning Research* 17.170 (2016), pp. 1–5.

[Bis+17] Bernd Bischl et al. *mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions*. 2017.

[Ros17] D. G. Rositter. *Tutorial: An example of statistical data analysis using the R environment for statistical computing*. 2017.

[Sch+17] Erich Schubert et al. "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN". In: *ACM Trans. Database Syst.* 42.3 (July 2017). DOI: 10.1145/3068335.

[WG17] Hadley Wickham and Garrett Grolemund. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 1st ed. O'Reilly Media, Jan. 2017.

[Lan+19] Michel Lang et al. "mlr3: A modern object-oriented machine learning framework in R". In: *Journal of Open Source Software* (2019). DOI: 10.21105/joss.01903.

[R C21] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2021.