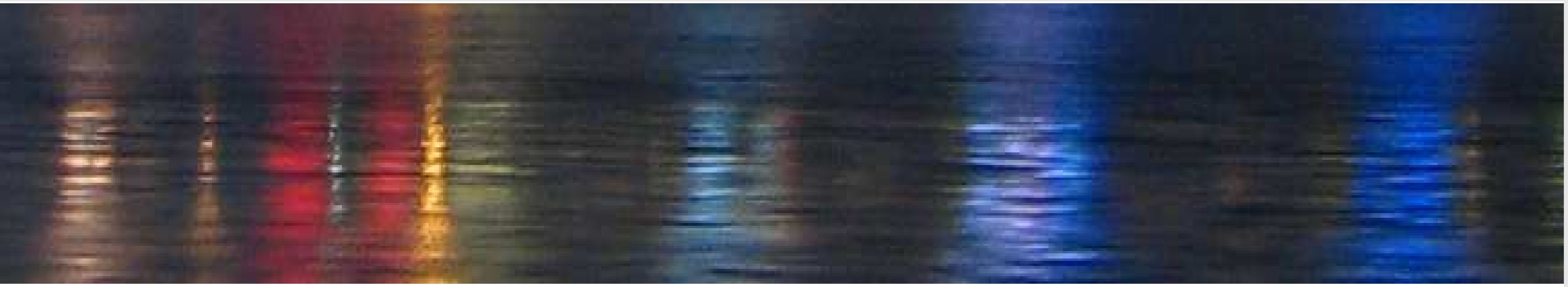


Intelligent Data Analysis

Prof. Rudolf Kruse
Pascal Held

Computational Intelligence Group
Department of Knowledge Processing and Language Engineering
Faculty of Computer Science
`kruse@iws.cs.uni-magdeburg.de`

About me

- 1979 diploma (Mathematics) degree from the University of Braunschweig, Germany
- 1980 PhD in Mathematics, 1984 the venia legendi in Mathematics from the same university
- 2 years at the Fraunhofer Gesellschaft
- 1986 joined the University of Braunschweig as a professor of computer science
- Since 1996 he is a full professor at the Department of Computer Science of the University of Magdeburg
- **Research:** statistics, artificial intelligence, expert systems, fuzzy control, fuzzy data analysis, computational intelligence, and information mining

Organisational

Lecture

- Thursday, 13.15-14.45, 29-K059
- Prof. Kruse
- Consultation: wednesday, 11 - 12 a.m., G29-008
- Preferred way of contact: `kruse@iws.cs.uni-magdeburg.de`

Exercises

- Tuesday 11.15-12.45 22a-122
- Tutor: Pascal Held
- G29-015, `pheld@iws.cs.uni-magdeburg.de`

Updated Information on the Course

- `http://fuzzy.cs.uni-magdeburg.de/`

Acknowledgement

- We thank Christian Borgelt for providing several slides for this course, that he produced during his time as a scientific researcher in our institute.

Organisational

Conditions for Certificates (Scheine)

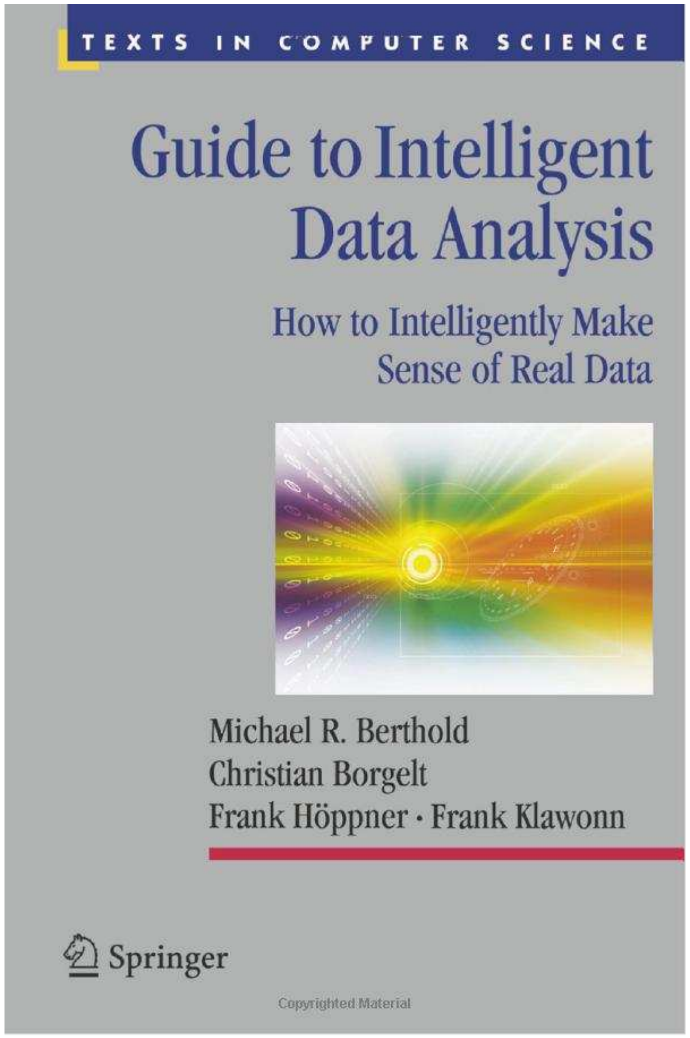
- ticked at least two thirds of the assignments,
- presented at least two times a solution during the exercise, and
- passed a small colloquium (approx. 10 min) or a written test (if there are more than 20 students) after the course.

Conditions for Exams

- no ticking required

Book

Berthold, Borgelt, Höppner, Klawonn: Guide to Intelligent Data Analysis, Springer 2011


TEXTS IN COMPUTER SCIENCE
Guide to Intelligent Data Analysis
How to Intelligently Make Sense of Real Data
Michael R. Berthold
Christian Borgelt
Frank Höppner · Frank Klawonn
Springer
Copyrighted Material

Intelligent Data Analysis

- **Introduction**
- **Data and Knowledge**
  - Characteristics and Differences of Data and Knowledge
  - Quality Criteria for Knowledge
  - Example: Tycho Brahe and Johannes Kepler
- **Knowledge Discovery and Data Mining**
  - How to Find Knowledge?
  - The Knowledge Discovery Process (KDD Process)
  - Data Analysis / Data Mining Tasks
  - Data Analysis / Data Mining Methods
- **Summary**

Introduction

- Today every enterprise uses electronic information processing systems.
  - Production and distribution planning
  - Stock and supply management
  - Customer and personnel management
- Usually these systems are coupled with a database system (e.g. databases of customers, suppliers, parts etc.).
- Every possible individual piece of information can be retrieved.
- However: **Data alone are not enough.**
  - In a database one may "not see the wood for the trees".
  - General patterns, structures, regularities go undetected.
  - Often such patterns can be exploited to increase turnover (e.g. joint sales in a supermarket).

Data

Examples of Data

- "Columbus discovered America in 1492."
- "Mr Jones owns a Volkswagen Golf."

Characteristics of Data

- refer to single instances
  (single objects, persons, events, points in time etc.)
- describe individual properties
- are often available in huge amounts (databases, archives)
- are usually easy to collect or to obtain
  (e.g. cash registers with scanners in supermarkets, Internet)
- do not allow us to make predictions

Knowledge

Examples of Knowledge

- "All masses attract each other."
- "Every day at 5 pm there runs a train from Magdeburg to Berlin."

Characteristic of Knowledge

- refers to *classes* of instances
  (*sets* of objects, persons, points in time etc.)
- describes general patterns, structure, laws, principles etc.
- consists of as few statements as possible (this is an objective!)
- is usually difficult to find or to obtain
  (e.g. natural laws, education)
- allows us to make predictions

Criteria to Assess Knowledge

- Not all statements are equally important, equally substantial, equally useful.

  $\Rightarrow$ Knowledge must be assessed.

Assessment Criteria

- Correctness (probability, success in tests)
- Generality (range of validity, conditions of validity)
- Usefulness (relevance, predictive power)
- Comprehensibility (simplicity, clarity, parsimony)
- Novelty (previously unknown, unexpected)

Priority

- Science: correctness, generality, simplicity
- Economy: usefulness, comprehensibility, novelty

Tycho Brahe (1546–1601)

Who was Tycho Brahe?

- Danish nobleman and astronomer
- In 1582 he built an observatory on the island of Ven (32 km NE of Copenhagen).
- He determined the positions of the sun, the moon and the planets (accuracy: one angle minute, without a telescope!).
- He recorded the motions of the celestial bodies for several years.

Brahe's Problem

- He could not summarize the data he had collected in a uniform and consistent scheme.
- The planetary system he developed (the so-called Tychonic system) did not stand the test of time.

Johannes Kepler (1571–1630)

Who was Johannes Kepler?

- German astronomer and assistant of Tycho Brahe
- He advocated the Copernican planetary system.
- He tried all his life to find the laws that govern the motion of the planets.
- He started from the data that Tycho Brahe had collected.

Kepler's Laws

1. Each planet moves around the sun in an ellipse, with the sun at one focus.
2. The radius vector from the sun to the planet sweeps out equal areas in equal intervals of time.
3. The squares of the periods of any two planets are proportional to the cubes of the semi-major axes of their respective orbits: $T \sim a^{\frac{3}{2}}$.

How to find Knowledge?

We do not know any universal method to discover knowledge.

Problems

- Today huge amounts of data are available in databases.

  We are drowning in information,
  but starving for knowledge. John Naisbett

- Manual methods of analysis have long ceased to be feasible.
- Simple aids (e.g. displaying data in charts) are too limited.

Attempts to Solve the Problems

- Intelligent Data Analysis
- Knowledge Discovery in Databases
- Data Mining

Knowledge Discovery and Data Mining

As a response to the challenge raised by the growing volume of data a new research area has emerged, which is usually characterized by one of the following phrases:

- **Knowledge Discovery in Databases** (KDD)

  Usual characterization:

  KDD is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. [Fayyad et al. 1996]

- **Data Mining**
  - Data mining is that step of the knowledge discovery process in which data analysis methods are applied to find interesting patterns.
  - It can be characterized by a set of types of tasks that have to be solved.
  - It uses methods from a variety of research areas.

    (statistics, databases, machine learning, artificial intelligence, soft computing etc.)

The Knowledge Discovery Process (KDD Process)

Preliminary Steps

- estimation of potential benefit
- definition of goals, feasibility study

Main Steps

- check data availability, data selection, if necessary, data collection
- preprocessing (60-80% of total overhead)
  - unification and transformation of data formats
  - data cleaning (error correction, outlier detection, imputation of missing values)
  - reduction / focusing (sample drawing, feature selection, prototype generation)
- **Data Mining** (using a variety of methods)
- visualization (also in parallel to preprocessing, data mining, and interpretation)
- interpretation, evaluation, and test of results
- deployment and documentation

Data Analysis / Data Mining Tasks

- **Classification**
  Is this customer credit-worthy?
- **Segmentation, Clustering**
  What groups of customers do I have?
- **Concept Description**
  Which properties characterize fault-prone vehicles?
- **Prediction, Trend Analysis**
  What will the exchange rate of the dollar be tomorrow?
- **Dependence/Association Analysis**
  Which products are frequently bought together?
- **Deviation Analysis**
  Are there seasonal or regional variations in turnover?

Data Analysis / Data Mining Methods 1

- **Classical Statistics**
  (charts, parameter estimation, hypothesis testing, model selection, regression)
  tasks: classification, prediction, trend analysis
- **Bayes Classifiers**
  (probabilistic classification, naive and full Bayes classifiers)
  tasks: classification, prediction
- **Decision and Regression Trees**
  (top down induction, attribute selection measures, pruning)
  tasks: classification, prediction
- **k-nearest Neighbor/Case-based Reasoning**
  (lazy learning, similarity measures, data structures for fast search)
  tasks: classification, prediction

Data Analysis / Data Mining Methods 2

- **Artificial Neural Networks**
  (multilayer perceptrons, radial basis function networks, learning vector quantization)
  tasks: classification, prediction, clustering

- **Cluster Analysis**
  (k-means and fuzzy clustering, hierarchical agglomerative clustering)
  tasks: segmentation, clustering

- **Association Rule Induction**
  (frequent item set mining, rule generation)
  tasks: association analysis

- **Inductive Logic Programming**
  (rule generation, version space, search strategies, declarative bias)
  tasks: classification, association analysis, concept description

Statistics

Statistics

- **Descriptive Statistics**
  - Tabular and Graphical Representations
  - Characteristic Measures
  - Principal Component Analysis
- **Inductive Statistics**
  - Parameter Estimation
    (point and interval estimation, finding estimators)
  - Hypothesis Testing
    (parameter test, goodness-of-fit test, dependence test)
  - Model Selection
    (information criteria, minimum description length)
- **Summary**

Statistics: Introduction

Statistics is the art to collect, to display, to analyze, and to interpret data in order to gain new knowledge.

[Sachs 1999]

[...] statistics, that is, the mathematical treatment of reality, [...]

Hannah Arendt

There are lies, damned lies, and statistics.

Benjamin Disraeli

Statistics, n. Exactly 76.4% of all statistics (including this one) are invented on the spot. However, in 83% of cases it is inappropriate to admit it.

The Devil's IT Dictionary

86.8748648% of all statistics pretend an accuracy that is not justified by the applied methods.

source unknown

Basic Notions

- **Object, Case**
  Data describe objects, cases, persons etc.
- **(Random) Sample**
  The objects or cases described by a data set is called a *sample*,
  their number the *sample size*.
- **Attribute**
  Objects and cases are described by *attributes*,
  patients in a hospital, for example, by age, sex, blood pressure etc.
- **(Attribute) Value**
  Attributes have different possible *values*.
  The age of a patient, for example, is a non-negative number.
- **Sample Value**
  The value an attribute has for an object in the sample is called *sample value*.

Scale Types / Attribute Types

Scale Type	Possible Operations	Examples
nominal (categorical, qualitative)	equality	sex blood group
ordinal (rank scale, comparative)	equality greater/less than	exam grade wind strength
metric (interval scale, quantitative)	equality greater/less than difference maybe ratio	length weight time temperature

- Nominal scales are sometimes divided into *dichotomic* (two values) and *polytomic* (more than two values).
- Metric scales may or may not allow us to form a ratio:
  weight and length do, temperature does not.
  time as duration does, time as calender time does not.

Descriptive Statistics

Tabular Representations: Frequency Table

- Given data set: $x = (3, 4, 3, 2, 5, 3, 1, 2, 4, 3, 3, 4, 4, 1, 5, 2, 2, 3, 5, 3, 2, 4, 3, 2, 3)$

a_k	h_k	r_k	$\sum_{i=1}^{k} h_i$	$\sum_{i=1}^{k} r_i$
1	2	$\frac{2}{25} = 0.08$	2	$\frac{2}{25} = 0.08$
2	6	$\frac{6}{25} = 0.24$	8	$\frac{8}{25} = 0.32$
3	9	$\frac{9}{25} = 0.36$	17	$\frac{17}{25} = 0.68$
4	5	$\frac{5}{25} = 0.20$	22	$\frac{22}{25} = 0.88$
5	3	$\frac{3}{25} = 0.12$	25	$\frac{25}{25} = 1.00$

- **Absolute Frequency** h_k (frequency of an attribute value a_k in the sample).
- **Relative Frequency** $r_k = \frac{h_k}{n}$, where n is the sample size (here $n = 25$).
- **Cumulated Absolute/Relative Frequency** $\sum_{i=1}^{k} h_i$ and $\sum_{i=1}^{k} r_i$.

Tabular Representations: Contingency Tables

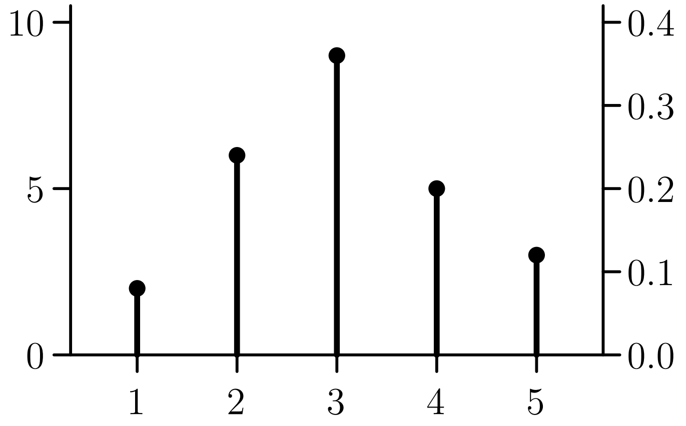
- Frequency tables for two or more attributes are called **contingency tables**.
- They contain the absolute or relative frequency of **value combinations**.

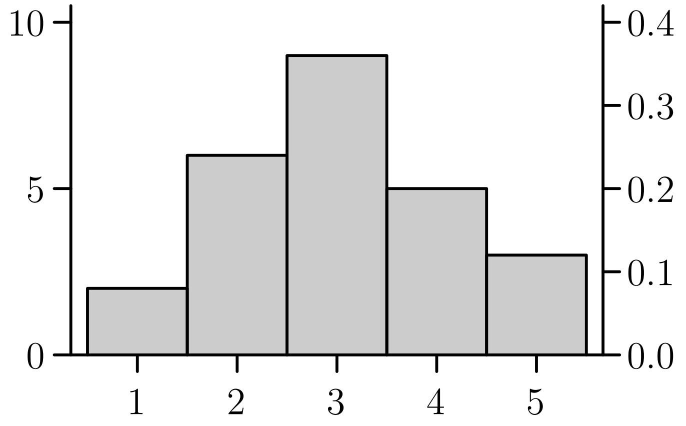
	a_1	a_2	a_3	a_4	$\sum$
b_1	8	3	5	2	18
b_2	2	6	1	3	12
b_3	4	1	2	7	14
$\sum$	14	10	8	12	44

- A contingency table may also contain the **marginal frequencies**, i.e., the frequencies of the values of individual attributes.
- Contingency tables for a higher number of dimensions (> 4) may be difficult to read.

Graphical Representations: Pole and Bar Chart

- Numbers, which may be, for example, the frequencies of attribute values are represented by the lengths of poles (left) or bars (right).

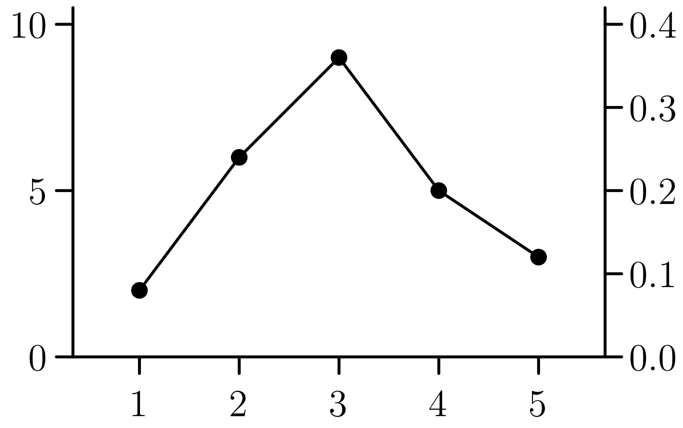




- Bar charts are the most frequently used and most comprehensible way of displaying absolute frequencies.
- A wrong impression can result if the vertical scale does not start at 0 (for frequencies or other absolute numbers).

Frequency Polygon and Line Chart

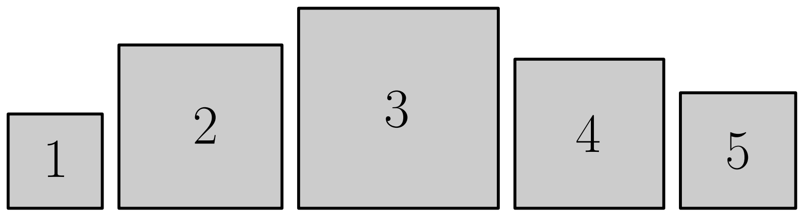
- Frequency polygon: the ends of the poles of a pole chart are connected by lines. (Numbers are still represented by lengths.)



- If the attribute values on the horizontal axis are not ordered, connecting the ends of the poles does not make sense.
- Line charts are frequently used to display time series.

Area and Volume Charts

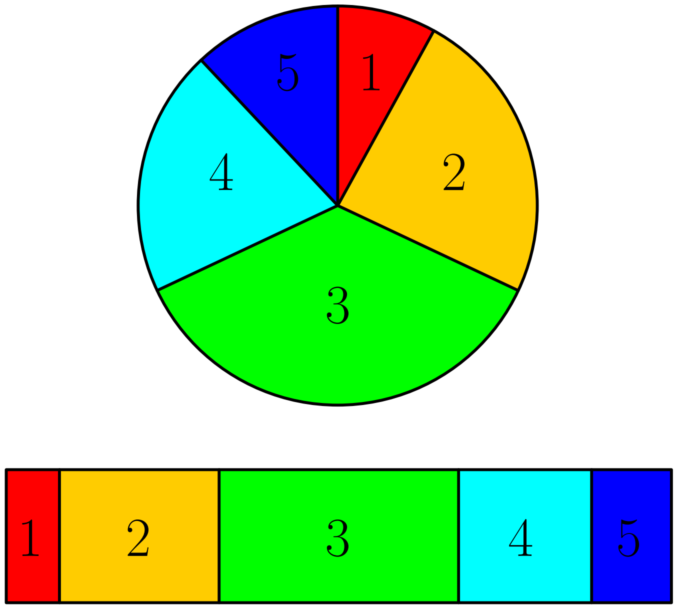
- Numbers may also be represented by other geometric quantities than lengths, like areas or volumes.
- Area and volume charts are usually less comprehensible than bar charts, because humans have more difficulties to compare areas and especially volumes than lengths. (exception: the represented numbers are areas or volumes)



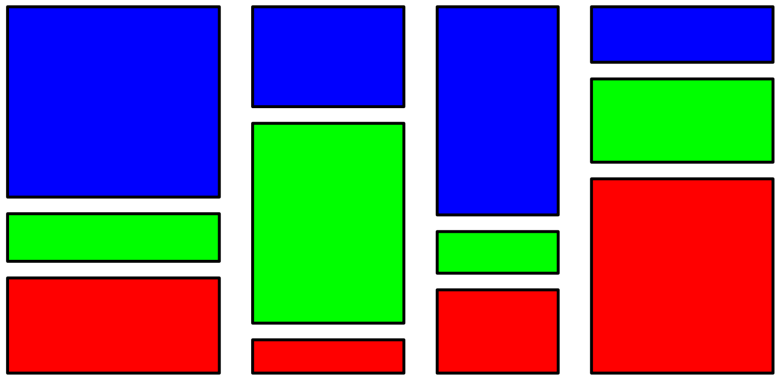
- Sometimes the height of a two- or three-dimensional object is used to represent a number. The diagram then conveys a misleading impression.

Pie and Stripe Charts

- Relative numbers may be represented by angles or sections of a stripe.



Mosaic Chart

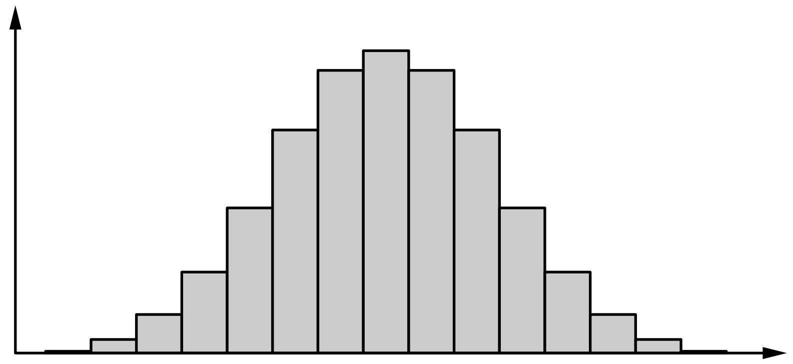
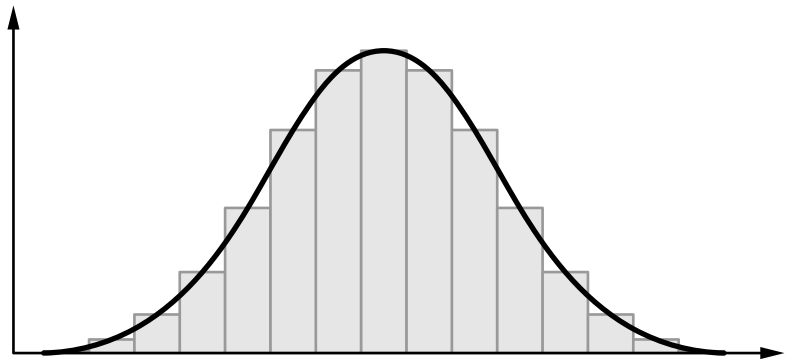


- Mosaic charts can be used to display contingency tables.
- More than two attributes are possible, but then separation distances and color must support the visualization to keep it comprehensible.

Histograms

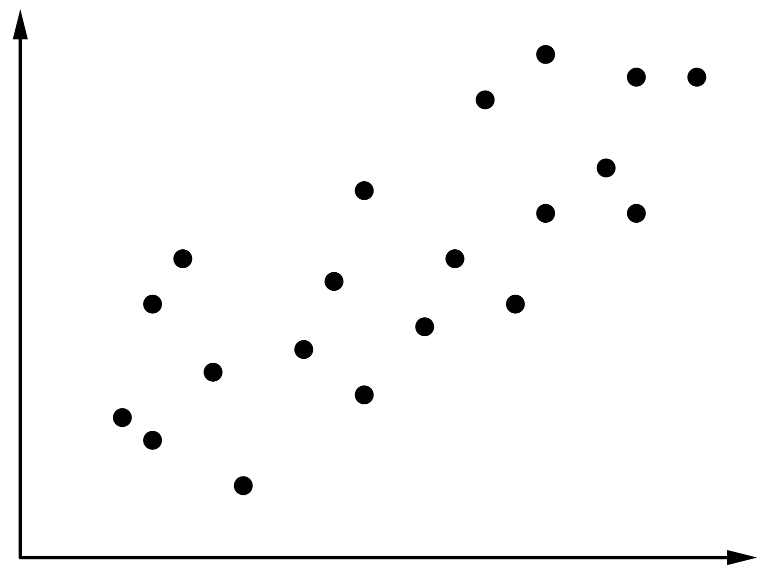
- Intuitively: **Histograms are frequency bar charts for metric data.**
- However: Since there are so many different values, **values have to be grouped** in order to arrive a proper representation.

  Most common approach: form equally sized intervals (so-called **bins**) and count the frequency of sample values inside each interval.
- **Attention:** Depending on the size and the position of the bins the histogram may look considerably different.
- In sketches often only a rough outline of a histogram is drawn:

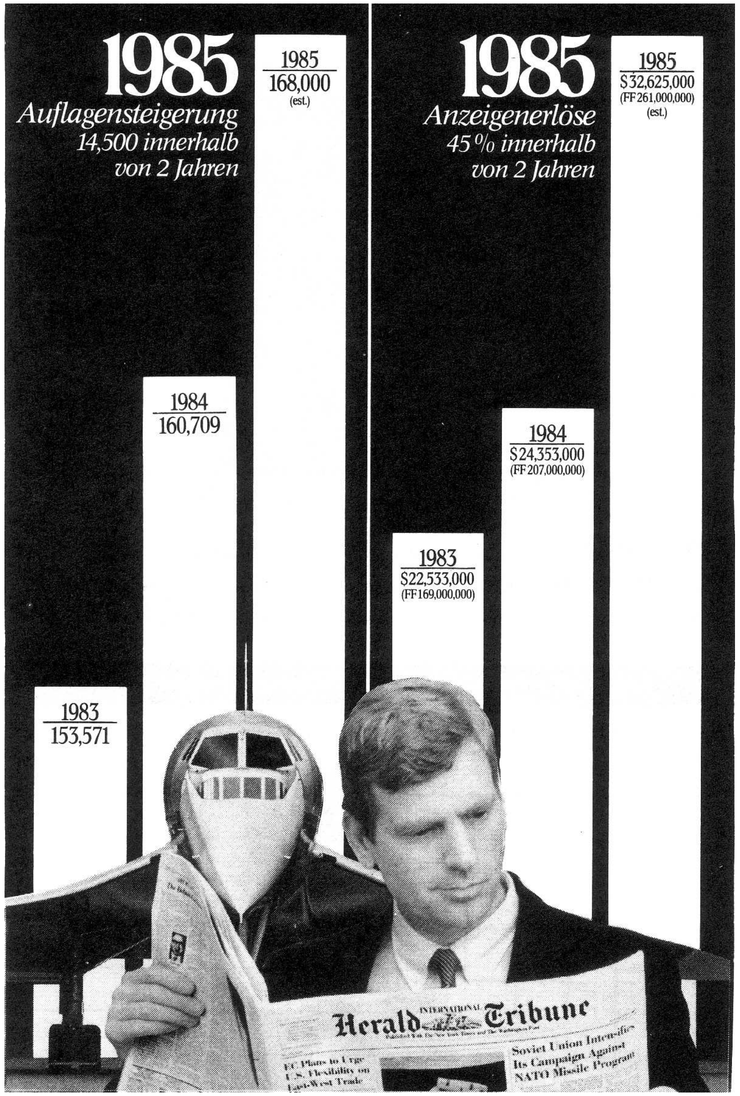
Scatter Plots

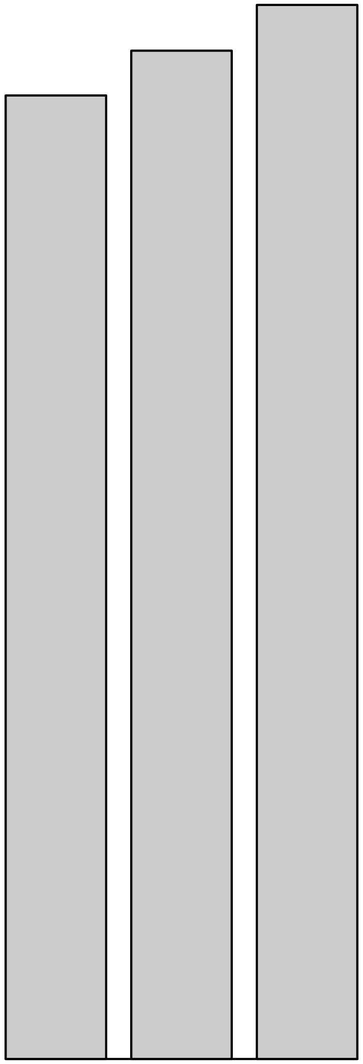
- Scatter plots are used to display two-dimensional metric data sets.
- Sample values are the coordinates of a point.
  (Numbers are represented by lengths.)



- Scatter plots provide a simple means for checking for dependency.

How to Lie with Statistics





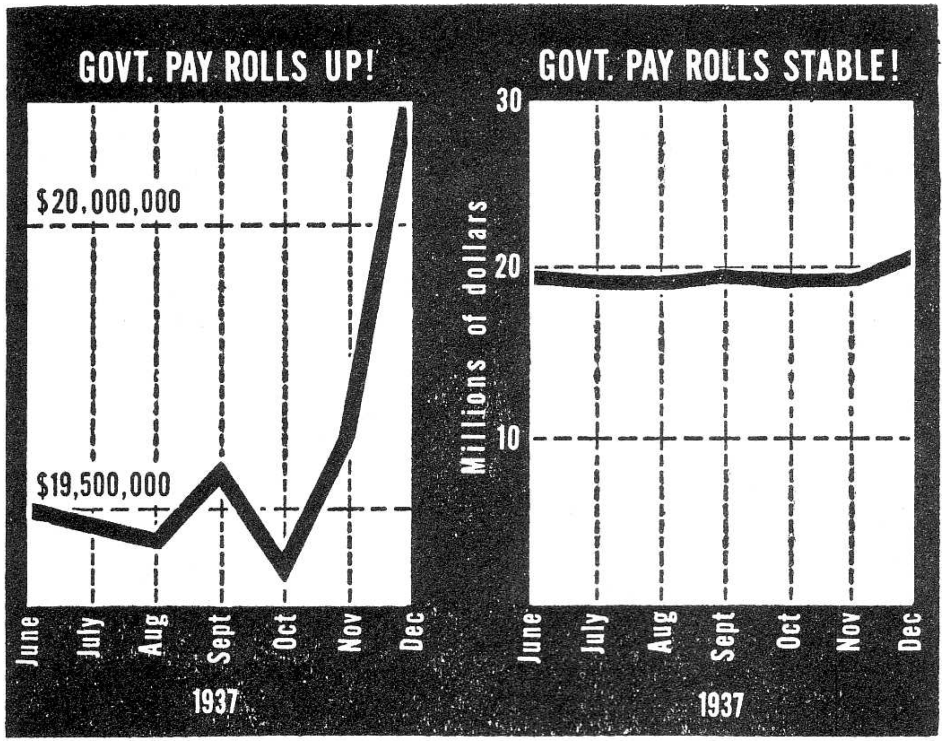
Often the vertical axis of a pole or bar chart does not start at zero, but at some higher value.

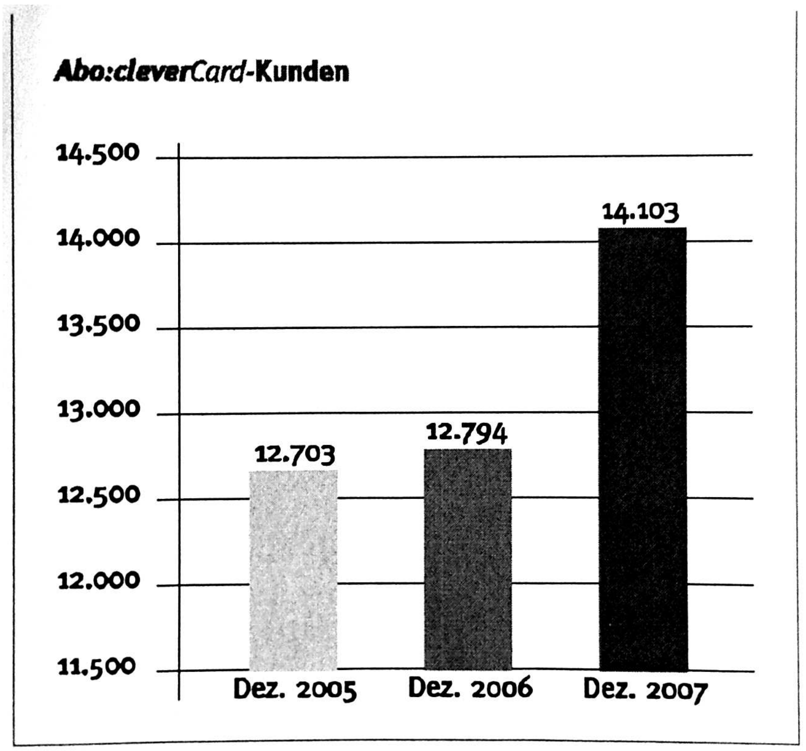
In such a case the conveyed impression of the ratio of the depicted values is completely wrong.

This effect is used to brag about increases in turnover, speed etc.

Sources of these diagrams and those on the following transparencies:
D. Huff: How to Lie with Statistics.
W. Krämer: So lügt man mit Statistik.

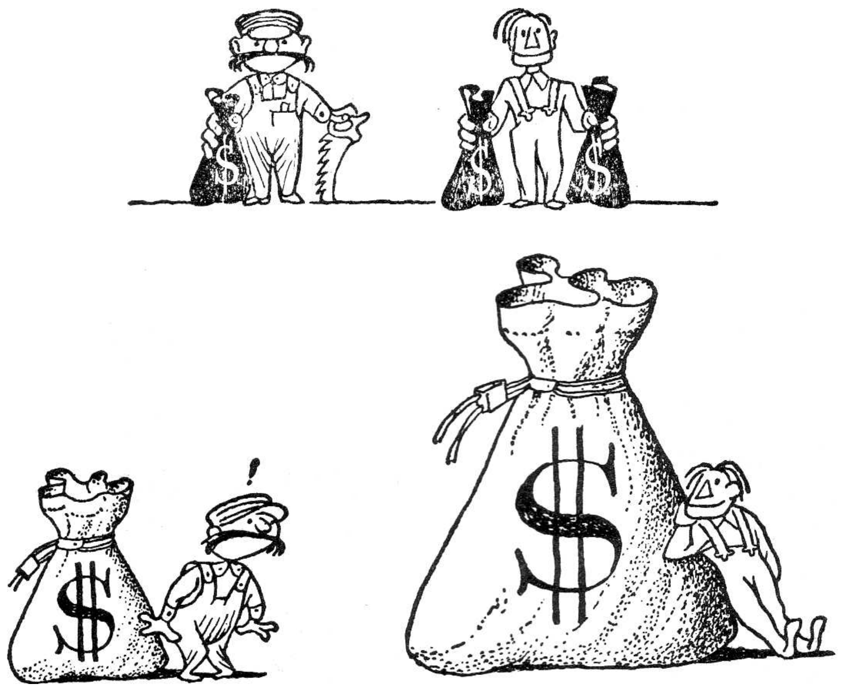
How to Lie with Statistics

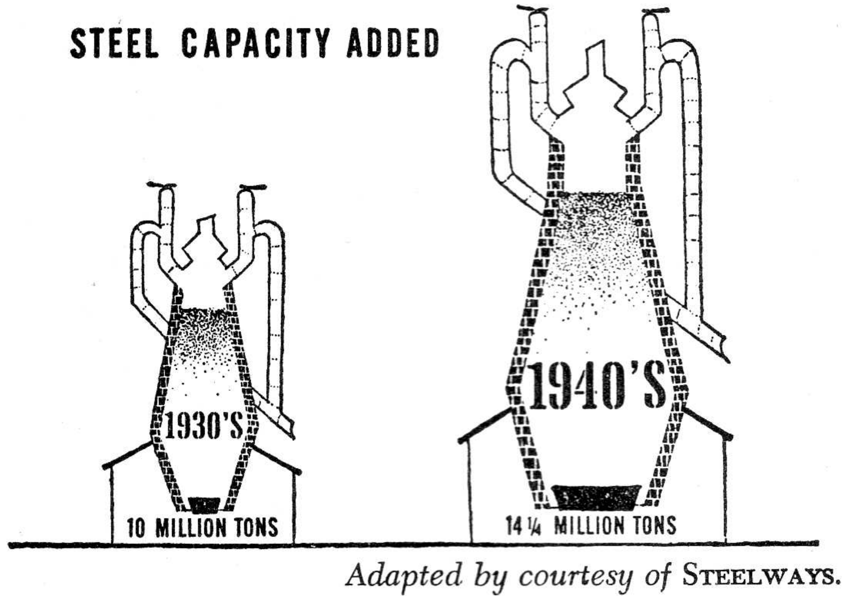




- Depending on the position of the zero line of a pole, bar, or line chart completely different impressions can be conveyed.

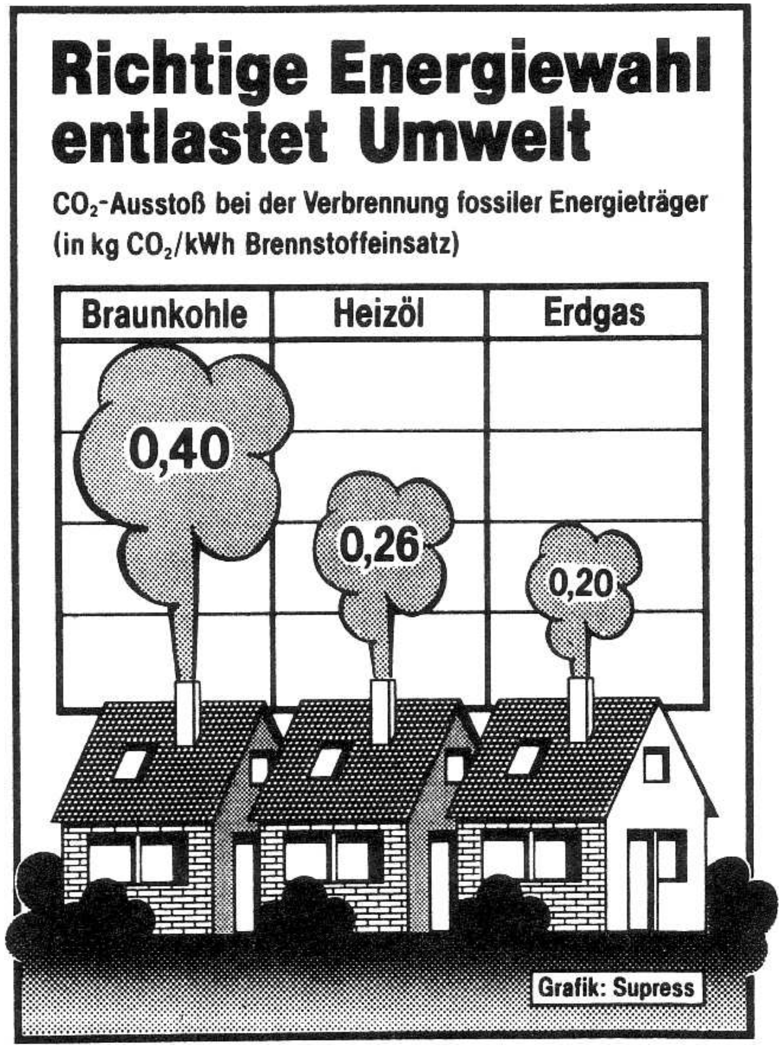
How to Lie with Statistics

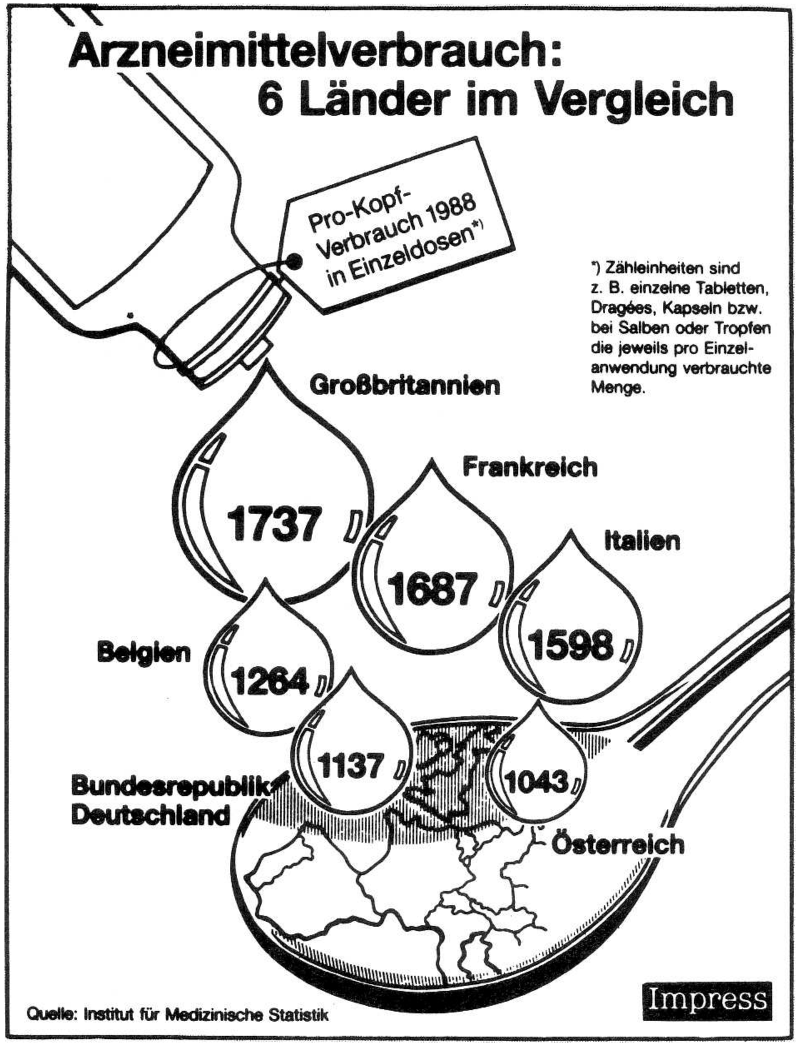




- Poles and bars are frequently replaced by (sketches of) objects in order to make the diagram more aesthetically appealing.
- However, objects are perceived as 2- or even 3-dimensional and thus convey a completely different impression of the numerical ratios.

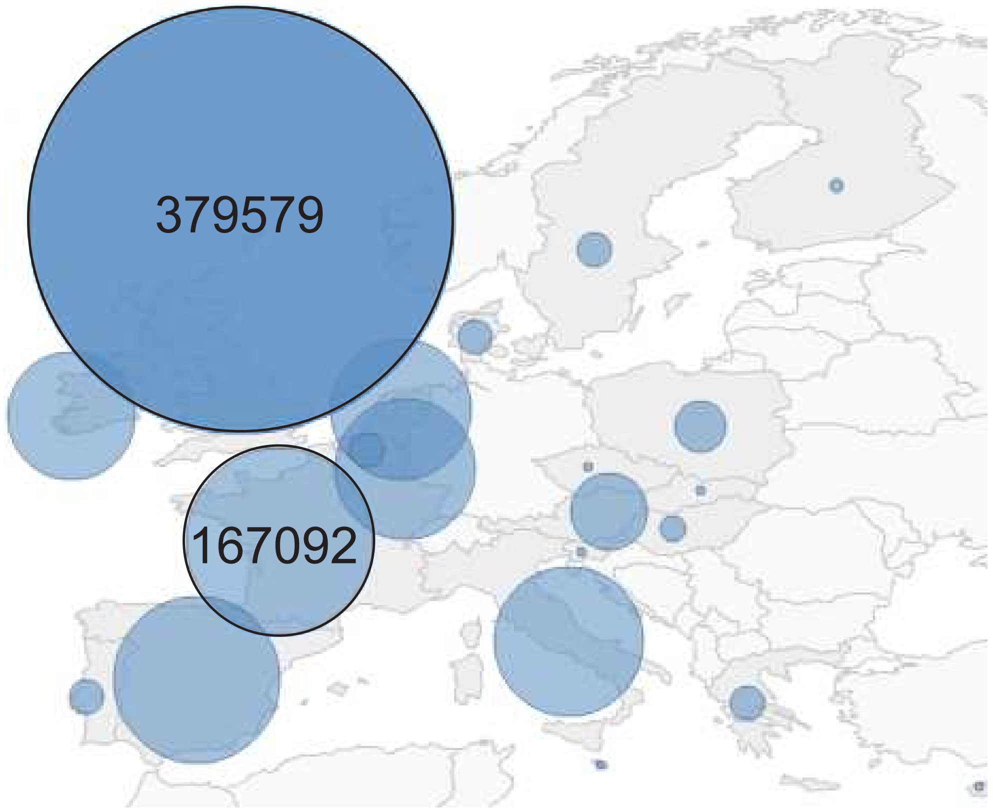
How to Lie with Statistics



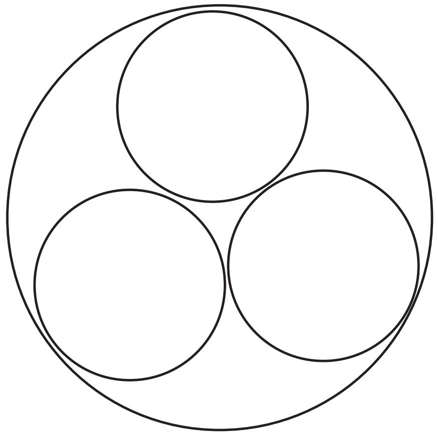


How to Lie with Statistics

Foreign outstanding debits of German banks in million Euros as of 2010:



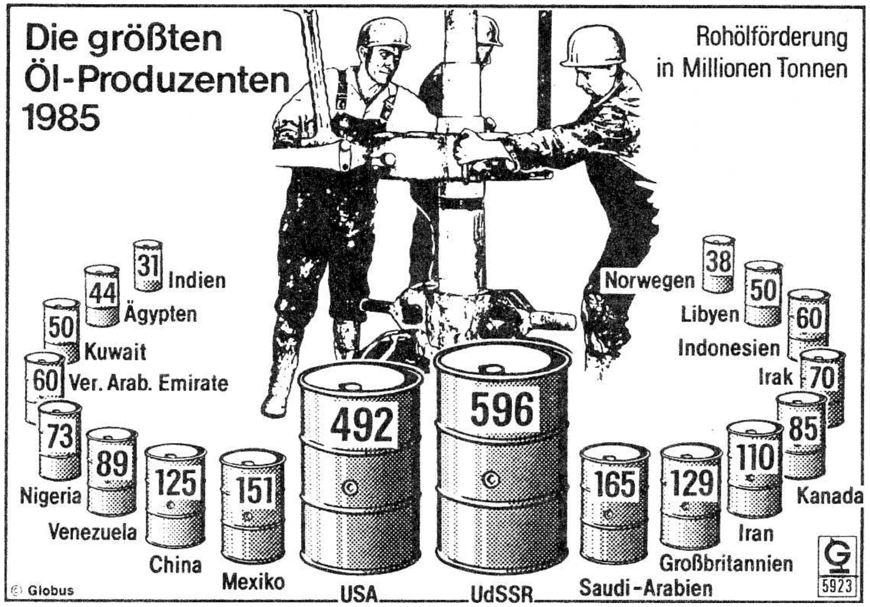
Source: Spiegel Online

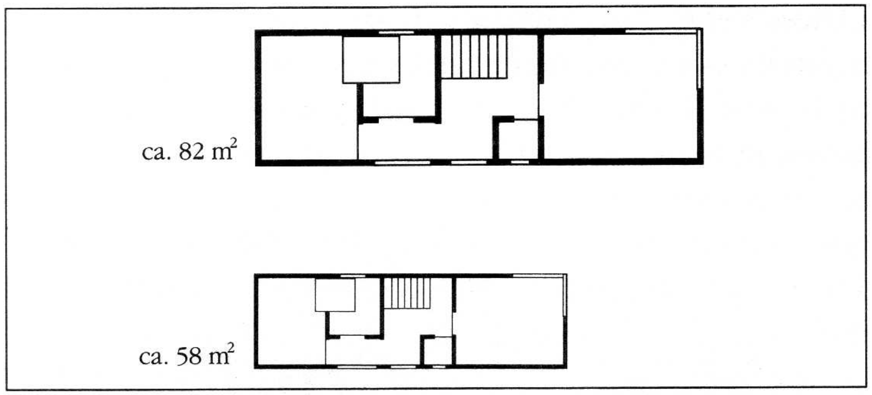


$$\frac{379579}{167092} \approx 2.2$$

$$\frac{A_{\text{UK}}}{A_{\text{F}}} \approx 5.0$$

How to Lie with Statistics





Quelle: "Zahlenspiegel" Bundesrepublik Deutschland - DDR – Ein Vergleich. 2. Auflage, Juli 1983, S. 63. Herausgeber: Bundesministerium für innerdeutsche Beziehungen.

- In the left diagram the areas of the barrels represent the numerical value. However, since the barrels are drawn 3-dimensional, a wrong impression of the numerical ratios is conveyed.
- The right diagram is particularly striking: an area measure is represented by the *side length* of a rectangle representing the apartment.

Descriptive Statistics: Characteristic Measures

Idea: Describe a given sample by few characteristic measures and thus summarize the data.

- **Localization Measures**
  Localization measures describe, usually by a single number, where the data points of a sample are located in the domain of an attribute.
- **Dispersion Measures**
  Dispersion measures describe how much the data points vary around a localization parameter and thus indicate how well this parameter captures the localization of the data.
- **Shape Measures**
  Shape measures describe the shape of the distribution of the data points relative to a reference distribution. The most common reference distribution is the normal distribution (Gaussian).

Localization Measures: Mode and Median

- **Mode** x^*
  The mode is the attribute value that is most frequent in the sample.
  It need not be unique, because several values can have the same frequency.
  It is the most general measure, because it is applicable for all scale types.

- **Median** $\tilde{x}$
  The median minimizes the sum of absolute differences:

$$\sum_{i=1}^{n} |x_i - \tilde{x}| = \min. \qquad \text{and thus it is} \qquad \sum_{i=1}^{n} \operatorname{sgn}(x_i - \tilde{x}) = 0$$

If $x = (x_{(1)}, \ldots, x_{(n)})$ is a sorted data set, the median is defined as

$$\tilde{x} = \begin{cases} x_{\left(\frac{n+1}{2}\right)}, & \text{if } n \text{ is odd,} \\ \frac{1}{2}\left(x_{\left(\frac{n}{2}\right)} + x_{\left(\frac{n}{2}+1\right)}\right), & \text{if } n \text{ is even.} \end{cases}$$

The median is applicable to ordinal and metric attributes.

Localization Measures: Arithmetic Mean

- **Arithmetic Mean** $\bar{x}$
  The arithmetic mean minimizes the sum of squared differences:

  $$\sum_{i=1}^{n}(x_i - \bar{x})^2 = \min. \qquad \text{and thus it is} \qquad \sum_{i=1}^{n}(x_i - \bar{x}) = \sum_{i=1}^{n} x_i - n\bar{x} = 0$$

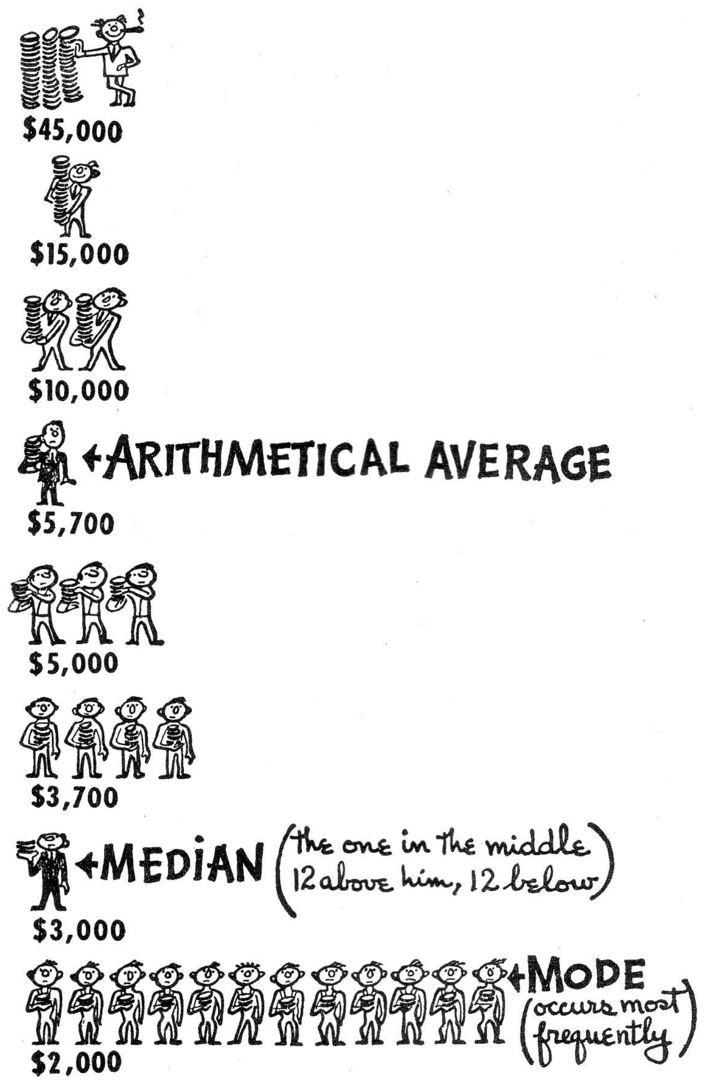
  The arithmetic mean is defined as

  $$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

  The arithmetic mean is only applicable to metric attributes.

- Even though the arithmetic mean is the most common localization measure, the **median** is preferable if
  - there are few sample cases,
  - the distribution is asymmetric, and/or
  - one expects that outliers are present.

How to Lie with Statistics


$45,000
$15,000
$10,000
←ARITHMETICAL AVERAGE
$5,700
$5,000
$3,700
←MEDIAN (the one in the middle 12 above him, 12 below)
$3,000
←MODE (occurs most frequently)
$2,000

»Sollen wir das arithmetische Mittel als durchschnittliche Körpergröße nehmen und den Gegner erschrecken, oder wollen wir ihn einlullen und nehmen den Median?«

Dispersion Measures: Range and Interquantile Range

A man with his head in the freezer and feet in the oven
is *on the average* quite comfortable.

old statistics joke

- **Range** R
  The range of a data set is the difference between the maximum and the minimum value.
  $$R = x_{\max} - x_{\min} = \max_{i=1}^{n} x_i - \min_{i=1}^{n} x_i$$

- **Interquantile Range**
  The p-quantile of a data set is a value such that a fraction of p of all sample values are smaller than this value. (The median is the $\frac{1}{2}$-quantile.)

  The p-interquantile range, $0 < p < \frac{1}{2}$, is the difference between the $(1-p)$-quantile and the p-quantile.

  The most common is the *interquartile range* ($p = \frac{1}{4}$)

Dispersion Measures: Average Absolute Deviation

- **Average Absolute Deviation**
  The average absolute deviation is the average of the absolute deviations of the sample values from the median or the arithmetic mean.

- Average Absolute Deviation from the **Median**

$$d_{\tilde{x}} = \frac{1}{n} \sum_{i=1}^{n} |x_i - \tilde{x}|$$

- Average Absolute Deviation from the **Arithmetic Mean**

$$d_{\bar{x}} = \frac{1}{n} \sum_{i=1}^{n} |x_i - \bar{x}|$$

- It is always $d_{\tilde{x}} \leq d_{\bar{x}}$, since the median minimizes the sum of absolute deviations. (see the definition of the median)

Dispersion Measures: Variance and Standard Deviation

- **Variance** s^2
  It would be natural to define the variance as the average squared deviation:

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

  However, inductive statistics suggests that it is better defined as

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

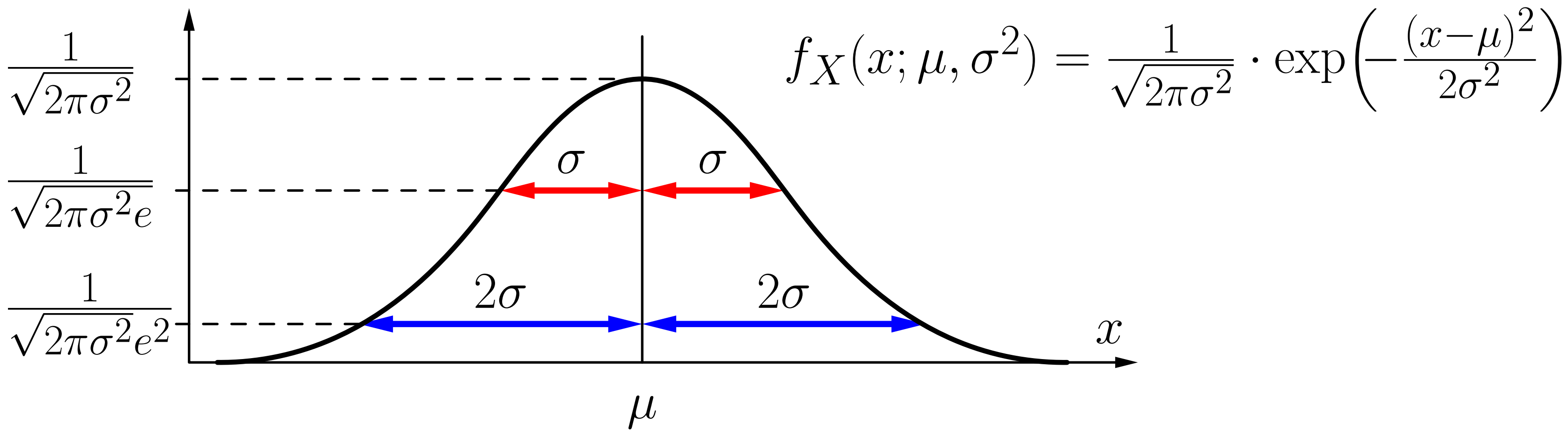
- **Standard Deviation** s
  The standard deviation is the square root of the variance, i.e.,

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

Dispersion Measures: Variance and Standard Deviation

- **Special Case: Normal/Gaussian Distribution**

  The variance/standard deviation provides information about the height of the mode and the width of the curve.



- μ: expected value, estimated by mean value $\bar{x}$
  σ^2: variance, estimated by (empirical) variance s^2
  σ: standard deviation, estimated by (empirical) standard deviation s
  (Details about parameter estimation are studied later.)

Dispersion Measures: Variance and Standard Deviation

Note that it is often more convenient to compute the variance using the formula that results from the following transformation:

$$\begin{aligned}
s^2 &= \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2 \;=\; \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i^2-2x_i\bar{x}+\bar{x}^2\right)\\
&= \frac{1}{n-1}\left(\sum_{i=1}^{n}x_i^2-2\bar{x}\sum_{i=1}^{n}x_i+\sum_{i=1}^{n}\bar{x}^2\right)\\
&= \frac{1}{n-1}\left(\sum_{i=1}^{n}x_i^2-2n\bar{x}^2+n\bar{x}^2\right) \;=\; \frac{1}{n-1}\left(\sum_{i=1}^{n}x_i^2-n\bar{x}^2\right)\\
&= \frac{1}{n-1}\left(\sum_{i=1}^{n}x_i^2-\frac{1}{n}\left(\sum_{i=1}^{n}x_i\right)^2\right)
\end{aligned}$$

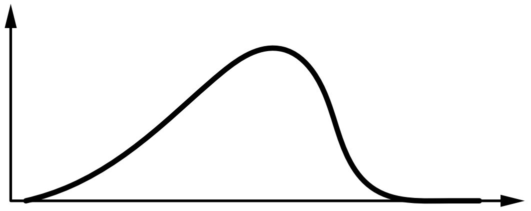
- Advantage: The sums $\sum_{i=1}^{n}x_i$ and $\sum_{i=1}^{n}x_i^2$ can both be computed in the same traversal of the data and from them both mean and variance are computable.

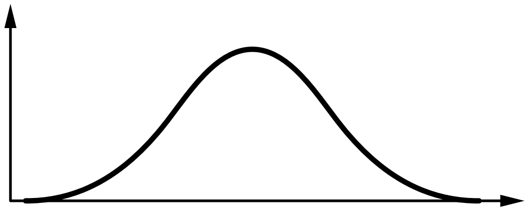
Shape Measures: Skewness

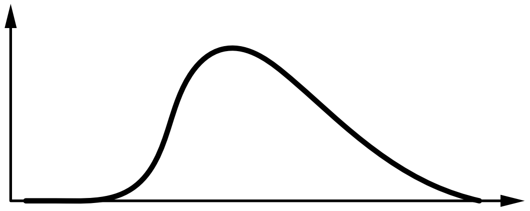
- The **skewness** α_3 (or **skew** for short) measures whether, and if, how much, a distribution differs from a symmetric distribution.
- It is computed from the 3rd moment about the mean, which explains the index 3.

$$\alpha_3 = \frac{1}{n \cdot v^3} \sum_{i=1}^{n} (x_i - \bar{x})^3 = \frac{1}{n} \sum_{i=1}^{n} z_i^3$$

$$\text{where} \quad z_i = \frac{x_i - \bar{x}}{v} \quad \text{and} \quad v^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2.$$


$\alpha_3 < 0$: right steep


$\alpha_3 = 0$: symmetric

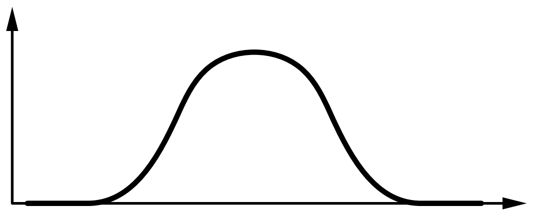

$\alpha_3 > 0$: left steep

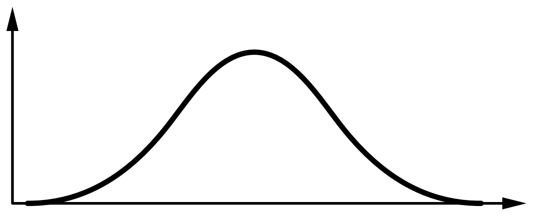
Shape Measures: Kurtosis

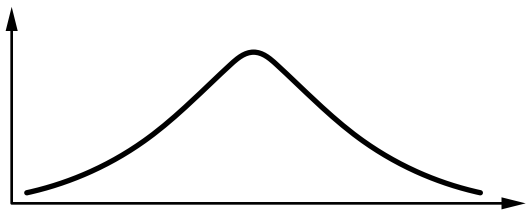
- The **kurtosis** or **excess** α_4 measures how much a distribution is arched, usually compared to a Gaussian distribution.
- It is computed from the 4th moment about the mean, which explains the index 4.

$$\alpha_4 \;=\; \frac{1}{n \cdot v^4} \sum_{i=1}^{n} (x_i - \bar{x})^4 \;=\; \frac{1}{n} \sum_{i=1}^{n} z_i^4$$

$$\text{where} \quad z_i = \frac{x_i - \bar{x}}{v} \quad \text{and} \quad v^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2.$$


$\alpha_4 < 3$: leptokurtic


$\alpha_4 = 3$: Gaussian


$\alpha_4 > 3$: platikurtic

Moments of Data Sets

- The k-th **moment** of a dataset is defined as

$$m'_k = \frac{1}{n}\sum_{i=1}^{n} x_i^k.$$

The first moment is the **mean** $m'_1 = \bar{x}$ of the data set.
Using the moments of a data set the **variance** s^2 can also be written as

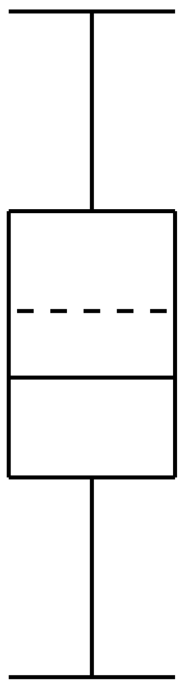
$$s^2 = \tfrac{1}{n-1}\left(m'_2 - \tfrac{1}{n}m'^2_1\right) \qquad \text{and also} \qquad v^2 = \tfrac{1}{n}m'_2 - \tfrac{1}{n^2}m'^2_1.$$

- The k-th **moment about the mean** is defined as

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

It is $m_1 = 0$ and $m_2 = v^2$ (i.e., the **average squared deviation**).
The **skewness** is $\alpha_3 = \frac{m_3}{m_2^{3/2}}$ and the **kurtosis** is $\alpha_4 = \frac{m_4}{m_2^2}$.

Visualizing Characteristic Measures: Box Plots



$x_{\max}$	maximum
Q_3	3. quartile
$\bar{x}$	arithmetic mean
$\tilde{x} = Q_2$	median/2. quartile
Q_1	1. quartile
$x_{\min}$	minimum

A box plot is a common way to combine some important characteristic measures into a single graphic.

Often the central box is drawn laced (⟩⟨) w.r.t. the arithmetic mean in order to emphasize its location.

Box plots are often used to get a quick impression of the distribution of the data by showing them side by side for several attributes.

Multidimensional Characteristic Measures

General Idea: Transfer the formulae to vectors.

- **Arithmetic Mean**
  The arithmetic mean for multidimensional data is the vector mean of the data points. For two dimensions it is

$$\overline{(x, y)} = \frac{1}{n} \sum_{i=1}^{n} (x_i, y_i) = (\bar{x}, \bar{y})$$

  For the arithmetic mean the transition to several dimensions only combines the arithmetic means of the individual dimensions into one vector.

- Other measures are transferred in a similar way.
  However, sometimes the transfer leads to new quantities, as for the variance.

Excursion: Vector Products

For the variance, the square of the difference to the mean has to be generalized.

Inner Product
Scalar Product

$$\begin{array}{cc}
\vec{v}^\top \vec{v} & \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix} \\
(v_1, v_2, \ldots, v_m) & \sum_{i=1}^{m} v_i^2
\end{array}$$

Outer Product
Matrix Product

$$\begin{array}{cc}
\vec{v}\vec{v}^\top & \begin{pmatrix} v_1, & v_2, & \ldots, & v_m \end{pmatrix} \\
\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix} & \begin{pmatrix} v_1^2 & v_1 v_2 & \cdots & v_1 v_m \\ v_1 v_2 & v_2^2 & \cdots & v_2 v_m \\ \vdots & & \ddots & \vdots \\ v_1 v_m & v_2 v_m & \cdots & v_m^2 \end{pmatrix}
\end{array}$$

- In principle both vector products may be used for a generalization.
- The second, however, yields more information about the distribution:
  - a measure of the (linear) dependence of the attributes,
  - a description of the direction dependence of the dispersion.

Covariance Matrix

- **Covariance Matrix**
  Compute variance formula with vectors (square: outer product $\vec{v}\vec{v}^\top$):

$$\mathbf{S} = \frac{1}{n-1}\sum_{i=1}^{n}\left(\begin{pmatrix}x_i\\y_i\end{pmatrix} - \begin{pmatrix}\bar{x}\\\bar{y}\end{pmatrix}\right)\left(\begin{pmatrix}x_i\\y_i\end{pmatrix} - \begin{pmatrix}\bar{x}\\\bar{y}\end{pmatrix}\right)^\top = \begin{pmatrix} s_x^2 & s_{xy} \\ s_{xy} & s_y^2 \end{pmatrix}$$

where

$$s_x^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right) \qquad \text{(variance of } x\text{)}$$

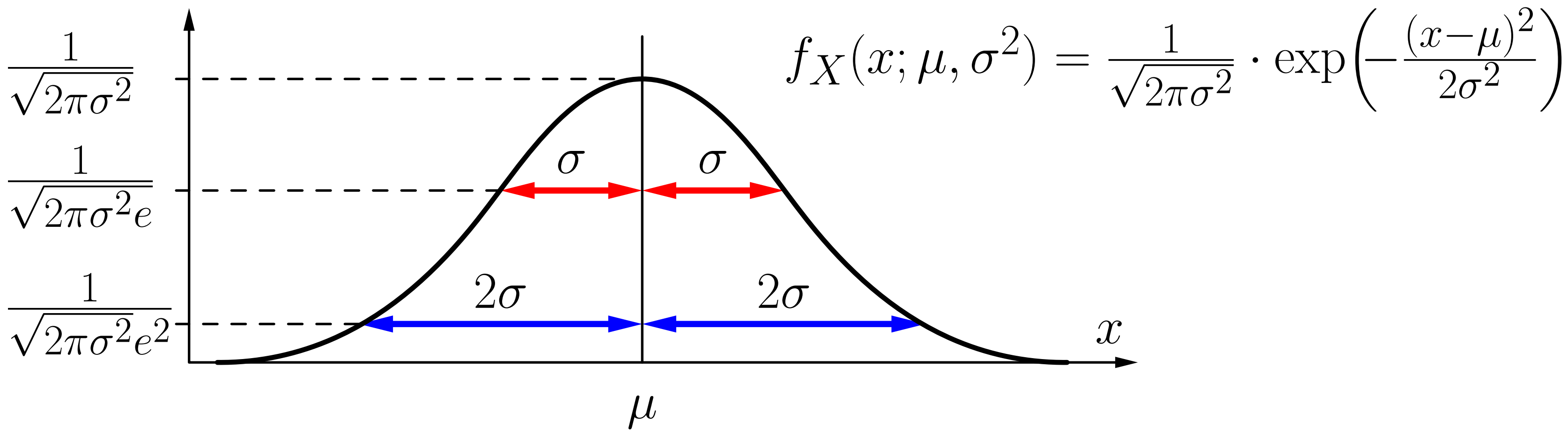
$$s_y^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} y_i^2 - n\bar{y}^2\right) \qquad \text{(variance of } y\text{)}$$

$$s_{xy} = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}\right) \qquad \text{(covariance of } x \text{ and } y\text{)}$$

Reminder: Variance and Standard Deviation

- **Special Case: Normal/Gaussian Distribution**

  The variance/standard deviation provides information about the height of the mode and the width of the curve.



$\frac{1}{\sqrt{2\pi\sigma^2}}$
$\frac{1}{\sqrt{2\pi\sigma^2 e}}$
$\frac{1}{\sqrt{2\pi\sigma^2 e^2}}$
σ
σ
2σ
2σ
x
μ
$f_X(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

- μ: expected value, estimated by mean value $\bar{x}$,
  σ^2: variance, estimated by (empirical) variance s^2,
  σ: standard deviation, estimated by (empirical) standard deviation s.
  Important: standard deviation has same unit as expected value.

Multivariate Normal Distribution

- A **univariate normal distribution** has the density function

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

μ:	expected value,	estimated by mean value $\bar{x}$,
σ^2:	variance,	estimated by (empirical) variance s^2,
σ:	standard deviation,	estimated by (empirical) standard deviation s.

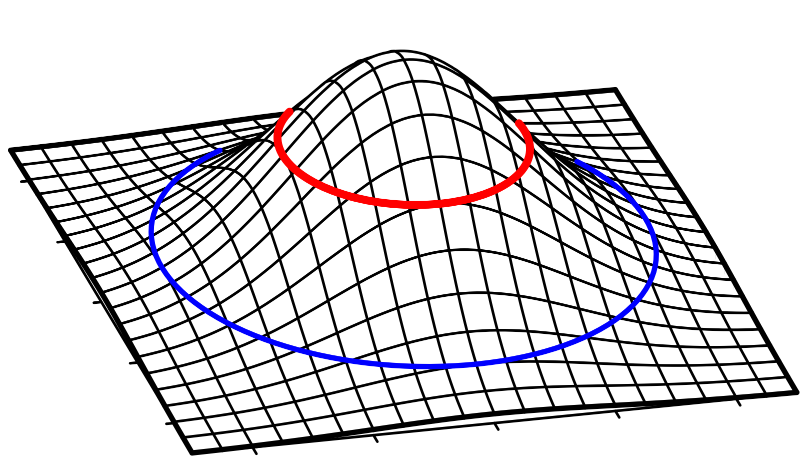
- A **multivariate normal distribution** has the density function

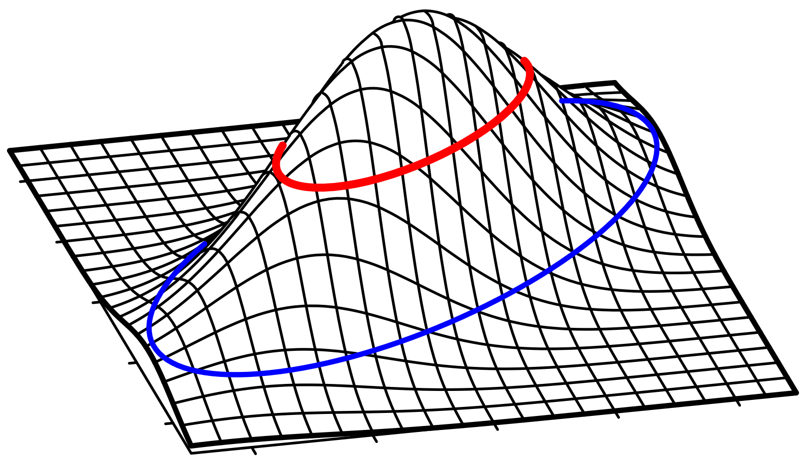
$$f_{\vec{X}}(\vec{x};\vec{\mu},\mathbf{\Sigma}) = \frac{1}{\sqrt{(2\pi)^m|\mathbf{\Sigma}|}} \cdot \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^\top \mathbf{\Sigma}^{-1}(\vec{x}-\vec{\mu})\right)$$

m:	size of the vector $\vec{x}$	(it is m-dimensional),
$\vec{\mu}$:	mean value vector,	estimated by (empirical) mean value vector $\bar{\vec{x}}$,
$\mathbf{\Sigma}$:	covariance matrix,	estimated by (empirical) covariance matrix $\mathbf{S}$,
$\lvert\mathbf{\Sigma}\rvert$:	determinant of the covariance matrix $\mathbf{\Sigma}$.	

Interpretation of a Covariance Matrix

- The variance/standard deviation relates the spread of the distribution to the spread of a **standard normal distribution** ($\sigma^2 = \sigma = 1$).
- The covariance matrix relates the spread of the distribution to the spread of a **multivariate standard normal distribution** ($\mathbf{\Sigma} = \mathbf{1}$).
- Example: bivariate normal distribution


standard


general

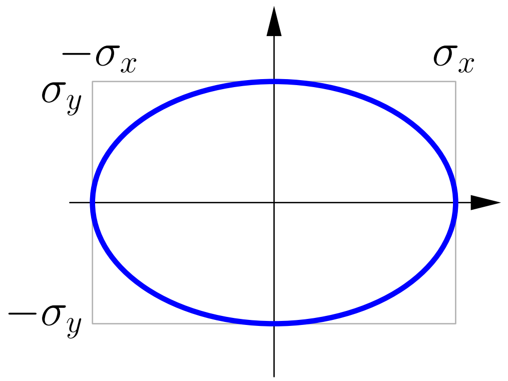
- **Question:** Is there a multivariate analog of standard deviation?

Interpretation of a Covariance Matrix

Question: Is there a multivariate analog of standard deviation?

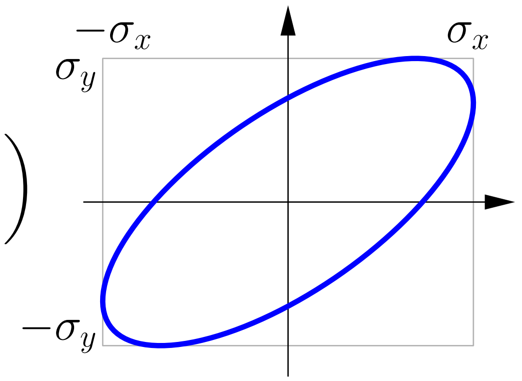
First insight:
If all covariances vanish,
the contour lines are axes-parallel ellipses.
The upper ellipse is inscribed into the
rectangle $[-\sigma_x, \sigma_x] \times [-\sigma_y, \sigma_y]$.

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix}$$



Second insight:
If the covariances do not vanish,
the contour lines are rotated ellipses.
Still the upper ellipse is inscribed into the
rectangle $[-\sigma_x, \sigma_x] \times [-\sigma_y, \sigma_y]$.

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix}$$



Consequence: A covariance matrix describes a scaling and a rotation.

Cholesky Decomposition

- Intuitively: **Compute an analog of standard deviation**.
- Let $\mathbf{S}$ be a symmetric, positive definite matrix (e.g. a covariance matrix). Cholesky decomposition serves the purpose to compute a "square root" of $\mathbf{S}$.
  - symmetric: $\forall 1 \leq i, j \leq m : s_{ij} = s_{ji}$
  - positive definite: for all m-dimensional vectors $\vec{v} \neq \vec{0}$ it is $\vec{v}^\top \mathbf{S}\,\vec{v} > 0$
- Formally: Compute a left/lower triangular matrix $\mathbf{L}$ such that $\mathbf{L}\mathbf{L}^\top = \mathbf{S}$. ($\mathbf{L}^\top$ is the transpose of the matrix $\mathbf{L}$.)

$$
\begin{aligned}
l_{ii} &= \left(s_{ii} - \sum_{k=1}^{i-1} l_{ik}^2\right)^{\frac{1}{2}} \\
l_{ji} &= \frac{1}{l_{ii}}\left(s_{ij} - \sum_{k=1}^{i-1} l_{ik} l_{jk}\right), \qquad j = i+1, i+2, \ldots, m.
\end{aligned}
$$

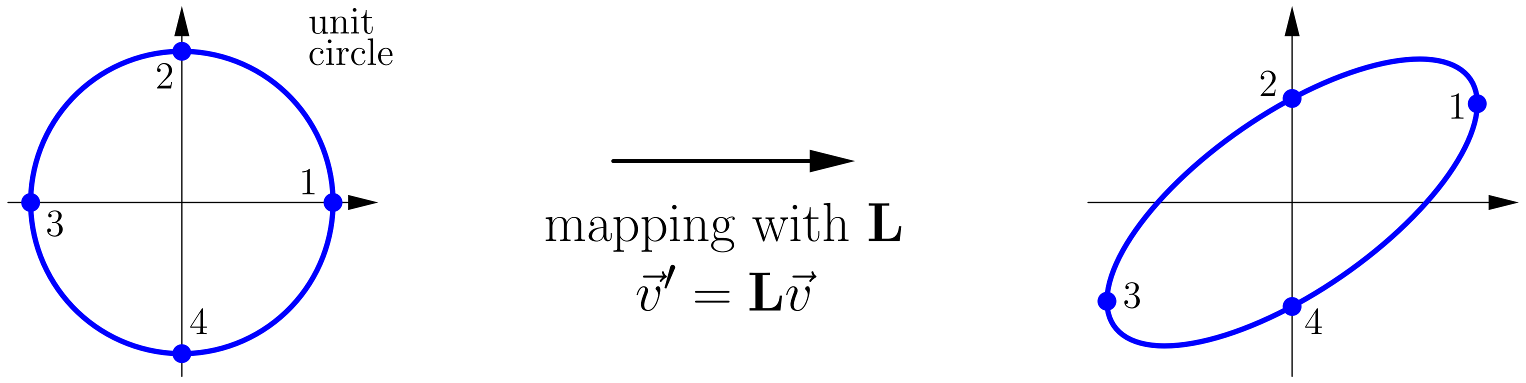
Special Case: Two Dimensions

- Covariance matrix

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix}$$

- Cholesky decomposition

$$\mathbf{L} = \begin{pmatrix} \sigma_x & 0 \\ \dfrac{\sigma_{xy}}{\sigma_x} & \dfrac{1}{\sigma_x}\sqrt{\sigma_x^2\sigma_y^2 - \sigma_{xy}^2} \end{pmatrix}$$



Eigenvalue Decomposition

- Also yields an **analog of standard deviation**.
- Computationally more expensive than Cholesky decomposition.
- Let $\mathbf{S}$ be a symmetric, positive definite matrix (e.g. a covariance matrix).
  - $\mathbf{S}$ can be written as

$$\mathbf{S} = \mathbf{R} \ \mathrm{diag}(\lambda_1, \ldots, \lambda_m) \ \mathbf{R}^{-1},$$

    where the λ_j, $j = 1, \ldots, m$, are the eigenvalues of $\mathbf{S}$ and the columns of $\mathbf{R}$ are the (normalized) eigenvectors of $\mathbf{S}$.
  - The eigenvalues λ_j, $j = 1, \ldots, m$, of $\mathbf{S}$ are all positive and the eigenvectors of $\mathbf{S}$ are orthonormal ($\rightarrow \mathbf{R}^{-1} = \mathbf{R}^\top$).
- Due to the above, $\mathbf{S}$ can be written as $\mathbf{S} = \mathbf{T}\,\mathbf{T}^\top$, where

$$\mathbf{T} = \mathbf{R} \ \mathrm{diag}\left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_m}\right)$$

Eigenvalue Decomposition

Special Case: Two Dimensions

- Covariance matrix

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix}$$

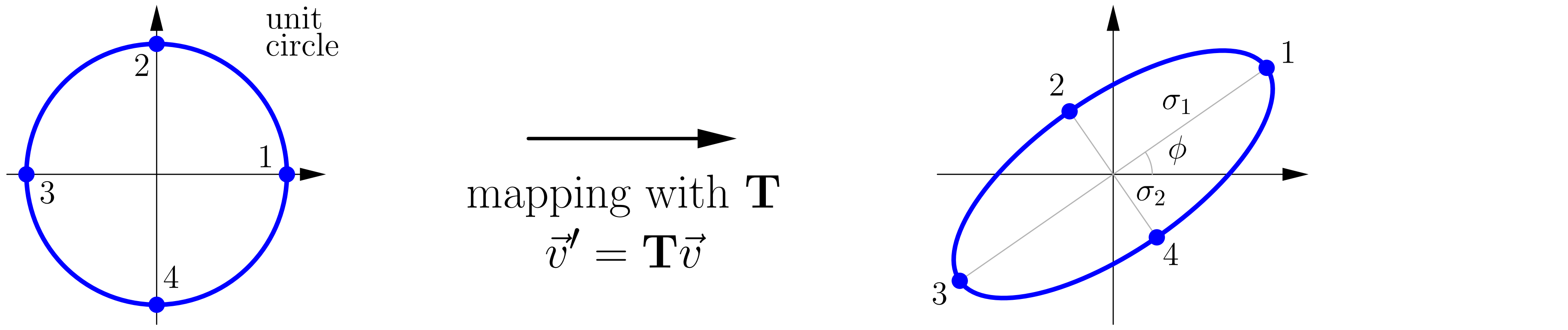
- Eigenvalue decomposition

$$\mathbf{T} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix},$$

$$s = \sin\phi, c = \cos\phi, \phi = \tfrac{1}{2}\arctan\tfrac{2\sigma_{xy}}{\sigma_x^2 - \sigma_y^2},$$

$$\sigma_1 = \sqrt{c^2\sigma_x^2 + s^2\sigma_y^2 + 2sc\sigma_{xy}},$$

$$\sigma_2 = \sqrt{s^2\sigma_x^2 + c^2\sigma_y^2 - 2sc\sigma_{xy}}.$$



Eigenvalue Decomposition

Eigenvalue decomposition enables us to write a covariance matrix $\mathbf{\Sigma}$ as

$$\mathbf{\Sigma} = \mathbf{T}\mathbf{T}^\top \qquad \text{with} \qquad \mathbf{T} = \mathbf{R}\,\text{diag}\left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_m}\right).$$

As a consequence we can write its inverse $\mathbf{\Sigma}^{-1}$ as

$$\mathbf{\Sigma}^{-1} = \mathbf{U}^\top\mathbf{U} \qquad \text{with} \qquad \mathbf{U} = \text{diag}\left(\lambda_1^{-\frac{1}{2}}, \ldots, \lambda_m^{-\frac{1}{2}}\right)\mathbf{R}^\top.$$

$\mathbf{U}$ describes the inverse mapping of $\mathbf{T}$, i.e., rotates the ellipse so that its axes coincide with the coordinate axes and then scales the axes to unit length. Hence:

$$(\vec{x} - \vec{y})^\top\mathbf{\Sigma}^{-1}(\vec{x} - \vec{y}) \;=\; (\vec{x} - \vec{y})^\top\mathbf{U}^\top\mathbf{U}(\vec{x} - \vec{y}) \;=\; (\vec{x}\,' - \vec{y}\,')^\top(\vec{x}\,' - \vec{y}\,'),$$

where $\quad \vec{x}\,' = \mathbf{U}\vec{x} \quad$ and $\quad \vec{y}\,' = \mathbf{U}\vec{y}.$

Result: $(\vec{x} - \vec{y})^\top\mathbf{\Sigma}^{-1}(\vec{x} - \vec{y})$ is equivalent to the squared **Euclidean distance** in the properly scaled eigensystem of the covariance matrix $\mathbf{\Sigma}$.

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^\top\mathbf{\Sigma}^{-1}(\vec{x} - \vec{y})}$$ is called **Mahalanobis distance**.

Eigenvalue Decomposition

Eigenvector decomposition also shows that the determinant of the covariance matrix $\mathbf{\Sigma}$ provides a measure of the (hyper-)volume of the (hyper-)ellipsoid. It is

$$|\mathbf{\Sigma}| = |\mathbf{R}|\ |\mathrm{diag}(\lambda_1, \ldots, \lambda_m)|\ |\mathbf{R}^\top| = |\mathrm{diag}(\lambda_1, \ldots, \lambda_m)| = \prod_{i=1}^{m} \lambda_i,$$

since $|\mathbf{R}| = |\mathbf{R}^\top| = 1$ as $\mathbf{R}$ is orthogonal with unit length columns, and thus

$$\sqrt{|\mathbf{\Sigma}|} = \prod_{i=1}^{m} \sqrt{\lambda_i},$$

which is proportional to the (hyper-)volume of the (hyper-)ellipsoid.

To be precise, the volume of the m-dimensional (hyper-)ellipsoid a (hyper-)sphere with radius r is mapped to with a covariance matrix $\mathbf{\Sigma}$ is

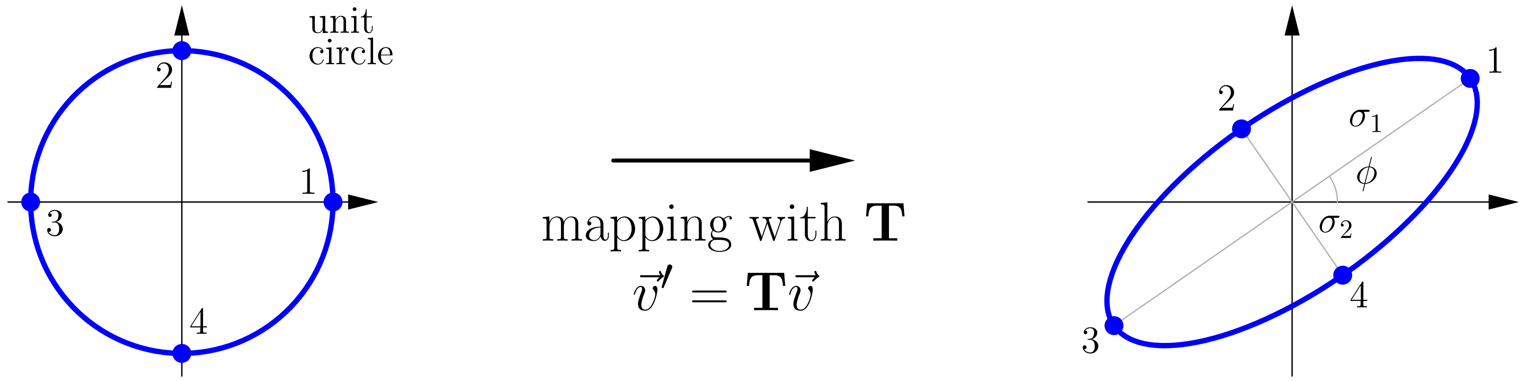
$$V_m(r) = \frac{\pi^{\frac{m}{2}} r^m}{\Gamma\left(\frac{m}{2} + 1\right)} \sqrt{|\mathbf{\Sigma}|}, \quad \text{where} \quad \begin{array}{l} \Gamma(x) = \int_0^\infty e^{-t} t^{x-1} \mathrm{d}t, \quad x > 0, \\ \Gamma(x+1) = x \cdot \Gamma(x), \quad \Gamma(\frac{1}{2}) = \sqrt{\pi}, \ \Gamma(1) = 1. \end{array}$$

Eigenvalue Decomposition

Special Case: Two Dimensions

- Covariance matrix and its eigenvalue decomposition:

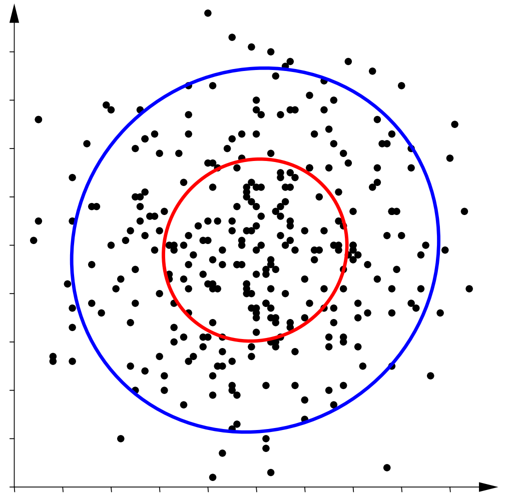
$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix} \qquad \text{and} \qquad \mathbf{T} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}.$$



- The area of the ellipse, to which the unit circle (area π) is mapped, is

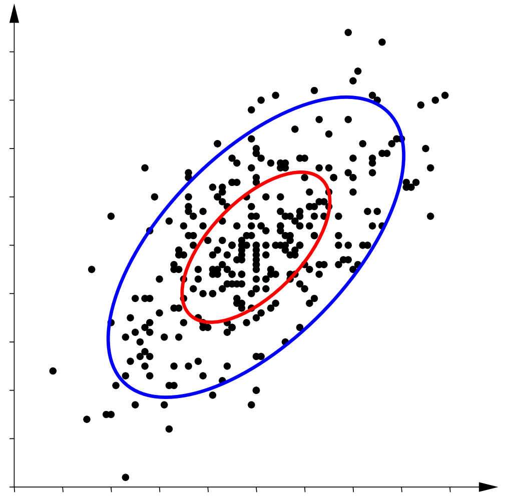
$$A = \pi\sigma_1\sigma_2 = \pi\sqrt{|\mathbf{\Sigma}|}.$$

Covariance Matrices of Example Data Sets



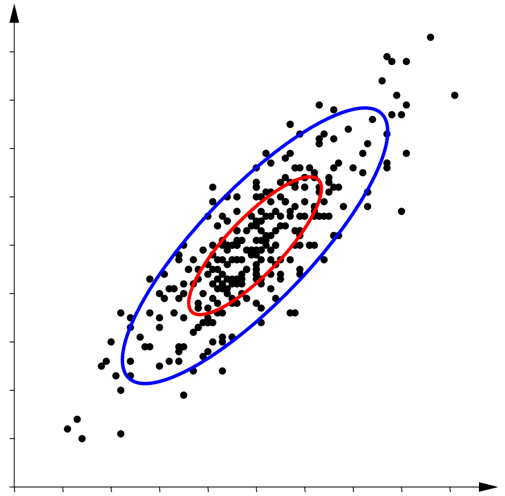
$$\mathbf{\Sigma} = \begin{pmatrix} 3.59 & 0.19 \\ 0.19 & 3.54 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1.90 & 0 \\ 0.10 & 1.88 \end{pmatrix}$$



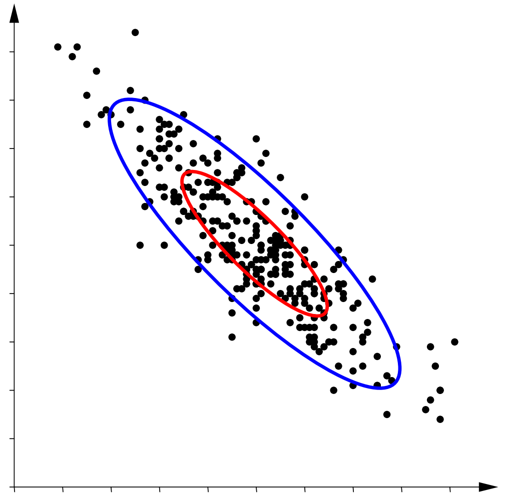
$$\mathbf{\Sigma} = \begin{pmatrix} 2.33 & 1.44 \\ 1.44 & 2.41 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1.52 & 0 \\ 0.95 & 1.22 \end{pmatrix}$$



$$\mathbf{\Sigma} = \begin{pmatrix} 1.88 & 1.62 \\ 1.62 & 2.03 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1.37 & 0 \\ 1.18 & 0.80 \end{pmatrix}$$



$$\mathbf{\Sigma} = \begin{pmatrix} 2.25 & -1.93 \\ -1.93 & 2.23 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1.50 & 0 \\ -1.29 & 0.76 \end{pmatrix}$$

Covariance Matrix: Summary

- A covariance matrix provides information about the **height of the mode** and about the **spread/dispersion** of a multivariate normal distribution (or of a set of data points that are roughly normally distributed).
- A multivariate **analog of standard deviation** can be computed with Cholesky decomposition and eigenvalue decomposition. The resulting matrix describes the distribution's shape and orientation.
- The shape and the orientation of a two-dimensional normal distribution can be visualized as an **ellipse** (curve of equal probability density; similar to a **contour line** — line of equal height — on a map.)
- The shape and the orientation of a three-dimensional normal distribution can be visualized as an **ellipsoid** (surface of equal probability density).
- The (square root of the) **determinant** of a covariance matrix describes the spread of a multivariate normal distribution with a single value. It is a measure of the area or (hyper-)volume of the (hyper-)ellipsoid.

Correlation and Principal Component Analysis

Correlation Coefficient

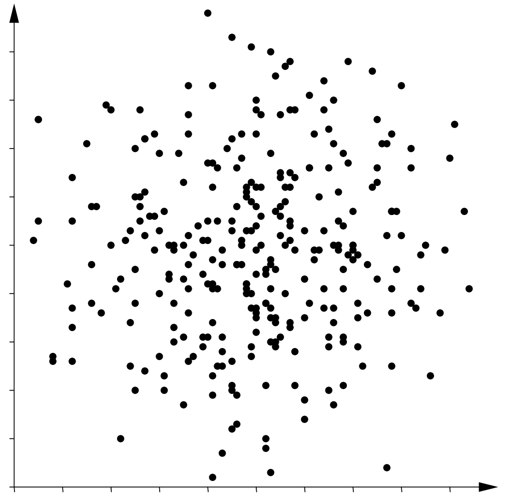
- The covariance is a measure of the strength of **linear dependence** of the two quantities.
- However, its value depends on the variances of the individual dimensions.
  $\Rightarrow$ Normalize to unit variance in the individual dimensions.
- **Correlation Coefficient**
  (more precisely: Pearson's Product Moment Correlation Coefficient)

$$r = \frac{s_{xy}}{s_x s_y}, \qquad\qquad r \in [-1, +1].$$

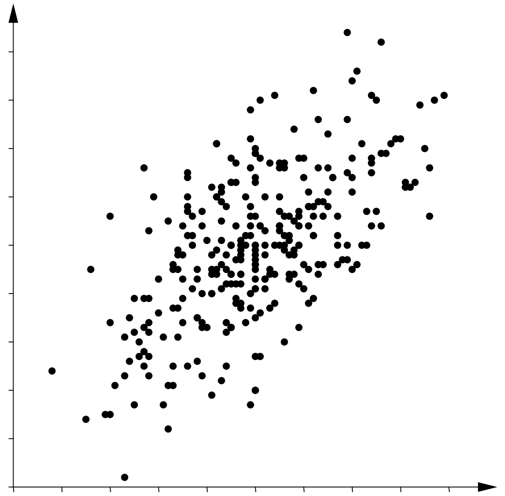
- r measures the strength of linear dependence:
  $r = -1$: the data points lie perfectly on a descending straight line.
  $r = +1$: the data points lie perfectly on an ascending straight line.
- $r = \quad 0$: there is no **linear** dependence between the two attributes
  (but there may be a non-linear dependence!).

Correlation Coefficients of Example Data Sets

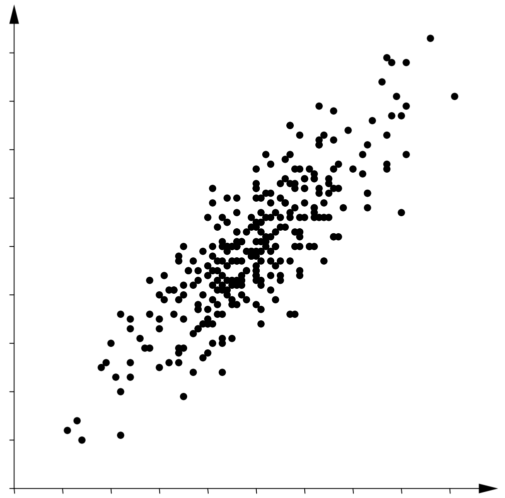
no correlation ($r \approx 0.05$)



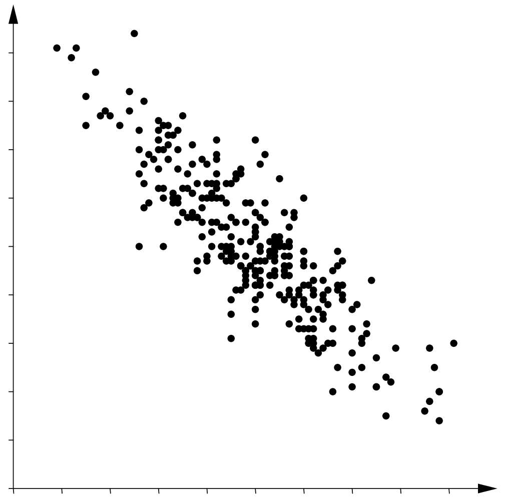
weak positive correlation ($r \approx 0.61$)



strong positive correlation ($r \approx 0.83$)



strong negative correlation ($r \approx -0.86$)



Correlation Matrix

- **Normalize Data**

  Transform data to mean value 0 and variance/standard deviation 1:

$$\forall i; 1 \leq i \leq n: \qquad x_i' = \frac{x_i - \bar{x}}{s_x}, \qquad y_i' = \frac{y_i - \bar{y}}{s_y}.$$

- **Compute Covariance Matrix of Normalized Data**

  Sum outer products of transformed data vectors:

$$\mathbf{\Sigma}' = \frac{1}{n-1} \sum_{i=1}^{n} \begin{pmatrix} x_i' \\ y_i' \end{pmatrix} \begin{pmatrix} x_i' \\ y_i' \end{pmatrix}^\top = \begin{pmatrix} 1 & r \\ r & 1 \end{pmatrix}$$

  Subtraction of mean vector is not necessary (because it is $(0,0)^\top$).
  Diagonal elements are always 1 (because of unit variance in each dimension).

- Normalizing the data and then computing the covariances or computing the covariances and then normalizing them has the same effect.

Correlation Matrix: Interpretation

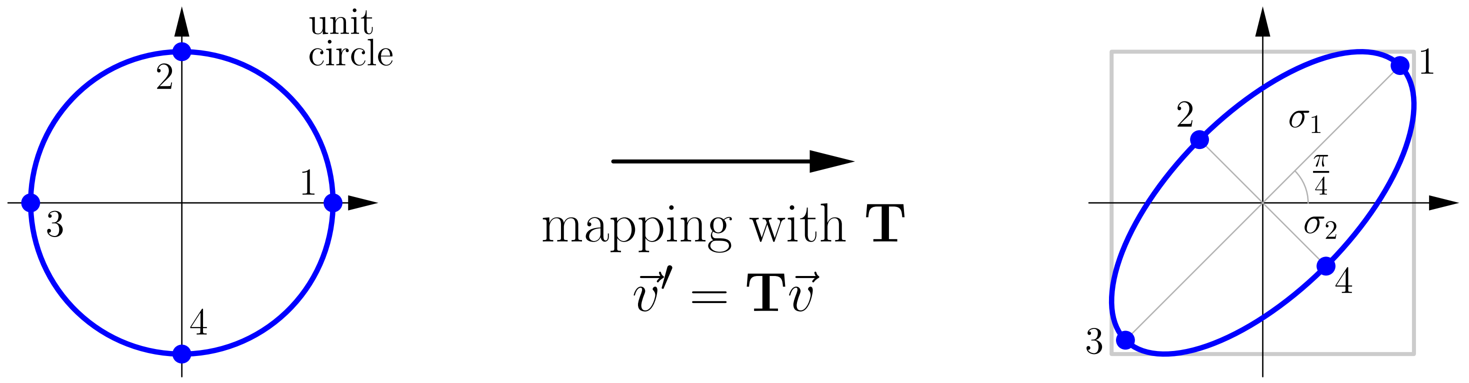
Special Case: Two Dimensions

- Correlation matrix

$$\mathbf{\Sigma}' = \begin{pmatrix} 1 & r \\ r & 1 \end{pmatrix}, \qquad \begin{array}{ll} \text{eigenvalues:} & \sigma_1^2,\ \sigma_2^2 \\ \text{correlation:} & r = \frac{\sigma_1^2 - \sigma_2^2}{\sigma_1^2 + \sigma_2^2} \end{array}$$

- Eigenvalue decomposition

$$\mathbf{T} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}, \qquad \begin{array}{ll} s = \sin\frac{\pi}{4} = \frac{1}{\sqrt{2}}, & \sigma_1 = \sqrt{1+r}, \\ c = \cos\frac{\pi}{4} = \frac{1}{\sqrt{2}}, & \sigma_2 = \sqrt{1-r}. \end{array}$$



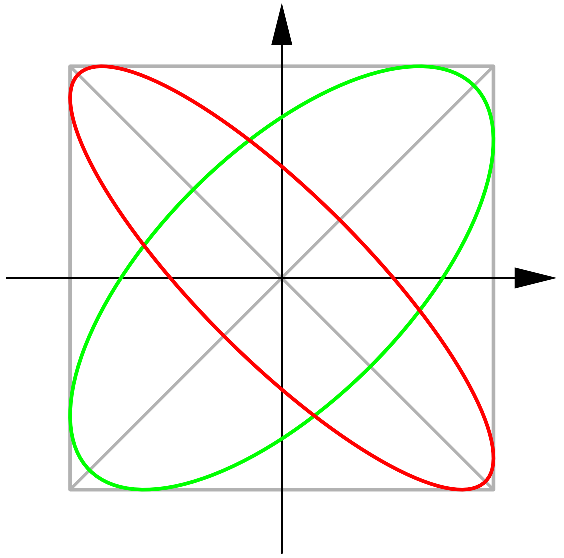
Correlation Matrix: Interpretation

- For two dimensions the eigenvectors of a correlation matrix are always

$$\vec{v}_1 = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right) \qquad \text{and} \qquad \vec{v}_2 = \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$$

  (or their opposites $-\vec{v}_1$ or $-\vec{v}_2$ or exchanged).

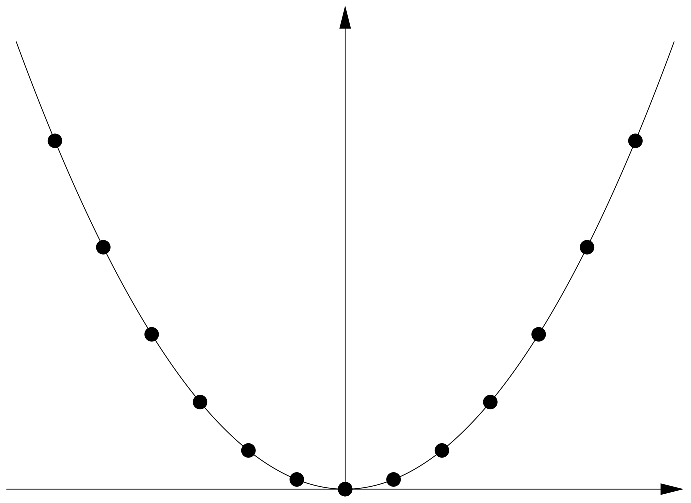
  The reason is that the normalization transforms the data points in such a way, that the ellipse, the unit circle is mapped to by the "square root" of the covariance matrix of the normalized data, is always inscribed into the square $[-1, 1] \times [-1, 1]$. Hence the ellipse's major axes are the square's diagonals.



- The situation is analogous in m-dimensional spaces: the eigenvectors are always m of the 2^{m-1} diagonals of the m-dimensional unit (hyper-)cube around the origin.

Correlation and Stochastic (In)Dependence

- Note: stochastic independence $\Rightarrow r = 0$,
  but: $r = 0 \not\Rightarrow$ stochastic independence.
- Example: Suppose the data points lie symmetrically on a parabola.



- The correlation coefficient of this data set is $r = 0$,
  because there is **no linear** dependence between the two attributes.
  However, there is a perfect **quadratic** dependence,
  and thus the two attributes are **not** stochastically independent.

Regression Line

- Since the covariance/correlation measures linear dependence,
  it is not surprising that it can be used to define a **regression line**:

$$(y - \bar{y}) = \frac{s_{xy}}{s_x^2}(x - \bar{x}) \qquad \text{or} \qquad y = \frac{s_{xy}}{s_x^2}(x - \bar{x}) + \bar{y}.$$

- The regression line can be seen as a conditional arithmetic mean:
  there is one arithmetic mean for the y-dimensions for each x-value.

- This interpretation is supported by the fact that the regression line minimizes the sum of squared differences in y-direction.
  (Reminder: the arithmetic mean minimizes the sum of squared differences.)

- More information on **regression** and the **method of least squares**
  in the corresponding chapter.

Principal Component Analysis

- Correlations between the attributes of a data set can be used to **reduce the number of dimensions**:
  - Of two strongly correlated features only one needs to be considered.
  - The other can be reconstructed approximately from the regression line.
  - However, the feature selection can be difficult.
- Better approach: **Principal Component Analysis** (PCA)
  - Find the direction in the data space that has the highest variance.
  - Find the direction in the data space that has the highest variance among those perpendicular to the first.
  - Find the direction in the data space that has the highest variance among those perpendicular to the first and second and so on.
  - Use first directions to describe the data.

Principal Component Analysis: Physical Analog

- The rotation of a body around an axis through its center of gravity can be described by a so-called **inertia tensor**, which is a 3×3-matrix

$$\Theta = \begin{pmatrix} \Theta_{xx} & \Theta_{xy} & \Theta_{xz} \\ \Theta_{xy} & \Theta_{yy} & \Theta_{yz} \\ \Theta_{xz} & \Theta_{yz} & \Theta_{zz} \end{pmatrix}.$$

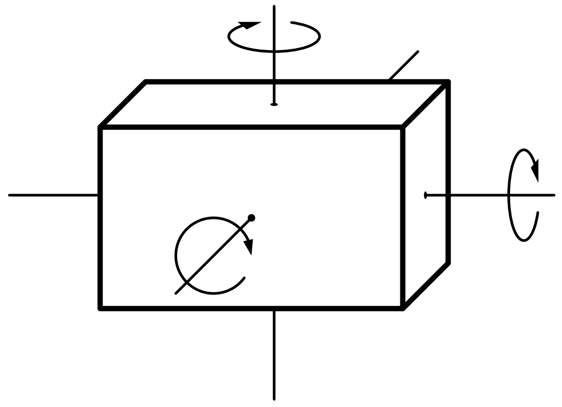
- The diagonal elements of this tensor are called the **moments of inertia**. They describe the "resistance" of the body against being rotated.

- The off-diagonal elements are the so-called **deviation moments**. They describe forces vertical to the rotation axis.

- All bodies possess three perpendicular axes through their center of gravity, around which they can be rotated without forces perpendicular to the rotation axis. These axes are called **principal axes of inertia**.

  There are bodies that possess more than 3 such axes (example: a homogeneous sphere), but all bodies have at least three such axes.

Principal Component Analysis: Physical Analog



The principal axes of inertia of a box.

- The deviation moments cause “rattling” in the bearings of the rotation axis, which cause the bearings to wear out quickly.
- A car mechanic who balances a wheel carries out, in a way, a principal axes transformation. However, instead of changing the orientation of the axes, he/she adds small weights to minimize the deviation moments.
- A statistician who does a principal component analysis, finds, in a way, the axes through a weight distribution with unit weights at each data point, around which it can be rotated most easily.

Principal Component Analysis: Formal Approach

- Normalize all attributes to arithmetic mean 0 and standard deviation 1:

$$x' = \frac{x - \bar{x}}{s_x}$$

- Compute the **correlation matrix** $\mathbf{\Sigma}$
  (i.e., the covariance matrix of the normalized data)

- Carry out a **principal axes transformation** of the correlation matrix, that is, find a matrix $\mathbf{R}$, such that $\mathbf{R}^\top \mathbf{\Sigma} \mathbf{R}$ is a diagonal matrix.

- Formal procedure:
  - Find the **eigenvalues** and **eigenvectors** of the correlation matrix, i.e., find the values λ_i and vectors $\vec{v}_i$, such that $\mathbf{\Sigma}\vec{v}_i = \lambda_i \vec{v}_i$.
  - The eigenvectors indicate the desired directions.
  - The eigenvalues are the variances in these directions.

Principal Component Analysis: Formal Approach

- Select dimensions using the **percentage of explained variance**.
    - The eigenvalues λ_i are the variances σ_i^2 in the principal dimensions.
    - It can be shown that the sum of the eigenvalues of an $m \times m$ correlation matrix is m. Therefore it is plausible to define $\frac{\lambda_i}{m}$ as the share the i-th principal axis has in the total variance.
    - Sort the λ_i descendingly and find the smallest value k, such that

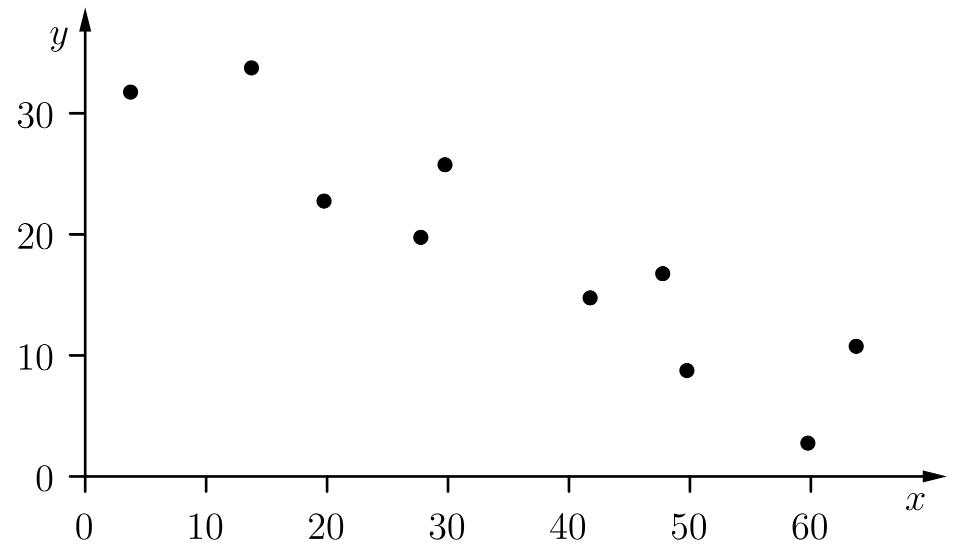
      $$\sum_{i=1}^{k} \frac{\lambda_i}{m} \geq \alpha,$$

      where α is a user-defined parameter (e.g. $\alpha = 0.9$).
    - Select the corresponding k directions (given by the eigenvectors).

- Transform the data to the new data space by multiplying the data points with a matrix, the rows of which are the eigenvectors of the selected dimensions.

Principal Component Analysis: Example

x	5	15	21	29	31	43	49	51	61	65
y	33	35	24	21	27	16	18	10	4	12



- Strongly correlated features $\Rightarrow$ Reduction to one dimension possible.

Principal Component Analysis: Example

Normalize to arithmetic mean 0 and standard deviation 1:

$$\begin{aligned}
\bar{x} &= \frac{1}{10}\sum_{i=1}^{10} x_i && = \frac{370}{10} && = 37, \\
\bar{y} &= \frac{1}{10}\sum_{i=1}^{10} y_i && = \frac{200}{10} && = 20, \\
s_x^2 &= \frac{1}{9}\left(\sum_{i=1}^{10} x_i^2 - 10\bar{x}^2\right) && = \frac{17290 - 13690}{9} && = 400 && \Rightarrow s_x = 20, \\
s_y^2 &= \frac{1}{9}\left(\sum_{i=1}^{10} y_i^2 - 10\bar{y}^2\right) && = \frac{4900 - 4000}{9} && = 100 && \Rightarrow s_y = 10.
\end{aligned}$$

x'	−1.6	−1.1	−0.8	−0.4	−0.3	0.3	0.6	0.7	1.2	1.4
y'	1.3	1.5	0.4	0.1	0.7	−0.4	−0.2	−1.0	−1.6	−0.8

Principal Component Analysis: Example

- Compute the correlation matrix (covariance matrix of normalized data).

$$\boldsymbol{\Sigma} = \frac{1}{9}\begin{pmatrix} 9 & -8.28 \\ -8.28 & 9 \end{pmatrix} = \begin{pmatrix} 1 & -\frac{23}{25} \\ -\frac{23}{25} & 1 \end{pmatrix}.$$

- Find the eigenvalues and eigenvectors, i.e., the values λ_i and vectors $\vec{v}_i$, $i = 1, 2$, such that

$$\boldsymbol{\Sigma}\vec{v}_i = \lambda_i \vec{v}_i \qquad \text{or} \qquad (\boldsymbol{\Sigma} - \lambda_i \mathbf{1})\vec{v}_i = \vec{0}.$$

  where $\mathbf{1}$ is the unit matrix.

- Here: Find the eigenvalues as the roots of the characteristic polynomial.

$$c(\lambda) = |\boldsymbol{\Sigma} - \lambda\mathbf{1}| = (1-\lambda)^2 - \frac{529}{625}.$$

  For more than 3 dimensions, this method is numerically unstable and should be replaced by some other method (Jacobi-Transformation, Householder Transformation to tridiagonal form followed by the QR algorithm etc.).

Principal Component Analysis: Example

- The roots of the characteristic polynomial $c(\lambda) = (1-\lambda)^2 - \frac{529}{625}$ are

$$\lambda_{1/2} = 1 \pm \sqrt{\frac{529}{625}} = 1 \pm \frac{23}{25}, \qquad \text{i.e.} \qquad \lambda_1 = \frac{48}{25} \quad \text{and} \quad \lambda_2 = \frac{2}{25}$$

- The corresponding eigenvectors are determined by solving for $i = 1, 2$ the (under-determined) linear equation system

$$(\mathbf{\Sigma} - \lambda_i \mathbf{1})\vec{v}_i = \vec{0}$$

- The resulting eigenvectors (normalized to length 1) are

$$\vec{v}_1 = \left(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right) \qquad \text{and} \qquad \vec{v}_2 = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right),$$

(Note that for two dimensions always these two vectors result.
Reminder: directions of the eigenvectors of a correlation matrix.)

Principal Component Analysis: Example

- Therefore the transformation matrix for the principal axes transformation is

$$\mathbf{R} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}, \qquad \text{for which it is} \qquad \mathbf{R}^\top \mathbf{\Sigma} \mathbf{R} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

- However, instead of $\mathbf{R}^\top$ we use $\sqrt{2}\mathbf{R}^\top$ to transform the data:

$$\begin{pmatrix} x'' \\ y'' \end{pmatrix} = \sqrt{2} \cdot \mathbf{R}^\top \cdot \begin{pmatrix} x' \\ y' \end{pmatrix}.$$

Resulting data set:

x''	-2.9	-2.6	-1.2	-0.5	-1.0	0.7	0.8	1.7	2.8	2.2
y''	-0.3	0.4	-0.4	-0.3	0.4	-0.1	0.4	-0.3	-0.4	0.6

- y'' is discarded ($s^2_{y''} = 2\lambda_2 = \frac{4}{25}$) and only x'' is kept ($s^2_{x''} = 2\lambda_1 = \frac{96}{25}$).

Probability Foundations

Reminder: Probability Theory

- **Goal**: Make statements and/or predictions about results of physical processes.
- Even processes that seem to be simple at first sight may reveal considerable difficulties when trying to predict.
- Describing real-world physical processes always calls for a simplifying mathematical model.
- Although everybody will have some intuitive notion about probability, we have to formally define the underlying mathematical structure.
- Randomness or chance enters as the incapability of precisely modelling a process or the inability of measuring the initial conditions.
    - *Example*: Predicting the trajectory of a billard ball over more than 9 banks requires more detailed measurement of the initial conditions (ball location, applied momentum etc.) than physically possible according to Heisenberg's uncertainty principle.

Reality vs. Model

- Producing a result of a physical process is referred to as an **observed outcome**.
- Assessing or predicting the probability of every possible outcome is not straightforward but often implicitly assumed to be clear.
- We will study this "non-straightforwardness" with three real-world examples:
  - Rolling a die.
  - Arrivals of inquiries at a call center.
  - The weight of a bread roll purchased from a bakery.
    (Inspired by a broadcast of Quarks & Co. from WDR.)
- Obviously, all examples differ in the nature of the space of possible observable outcomes.

Example 1: Rolling a Die

- **Physical Process**
  Shaking a six-sided die in a dice cup.
  Then cast it and read off the number of pips.

- **Possible Outcomes**

- **Sources of Randomness**
  - Inaccurate knowledge about locations, momenta.
  - Inellastic collisions inside the dice cup.
  - Inhomogeneous material distribution of the die.
  - Uneven table surface.
  - Unknown frictions, airflow etc.

- **Model**
  Outcomes have equal probability.

Example 2: Phone Calls at a Call Center

- **Physical Process**
  Counting the number of phone calls that arrive
  at a call center within a predefined time window.
- **Possible Outcomes**
  The events (if any) happening in time and space.
- **Sources of Randomness**
  - Calls are initiated by human beings: no predictability.
  - Misdialed calls.
  - Technical problems resulting in lost calls.
- **Model**
  Poisson distribution of number of calls.

Example 3: Bread Rolls at a Bakery

- **Physical Process**
  Baking a bread roll from a piece of dough.
  Measuring its weight (with arbitrary precision).

- **Possible Outcomes**
  Bread rolls.

- **Sources of Randomness**
  - Amount of dough put on the baking sheet.
  - Baking process (ingredients, temperature, time).

- **Model**
  Gaussian distribution of the weight.

Formal Approach on the Model Side

- We conduct an experiment that has a set Ω of possible outcomes.
  E. g.:
  - Rolling a die ($\Omega = \{1, 2, 3, 4, 5, 6\}$)
  - Arrivals of phone calls ($\Omega = \mathbb{N}_0$)
  - Bread roll weights ($\Omega = \mathbb{R}_+$)
- Such an outcome is called an **elementary event**.
- All possible elementary events are called the **frame of discernment** Ω (or sometimes **universe of discourse**).
- The set representation stresses the following facts:
  - All possible outcomes are covered by the elements of Ω.
    (**collectively exhaustive**).
  - Every possible outcome is represented by exactly one element of Ω.
    (**mutual disjoint**).

Events

- Often, we are interested in *higher-level* events (e. g. casting an odd number, arrival of at least 5 phone calls or purchasing a bread roll heavier than 80 grams)
- Any subset $A \subseteq \Omega$ is called an **event** which **occurs**, if the outcome $\omega_0 \in \Omega$ of the random experiment lies in A:

$$\text{Event } A \subseteq \Omega \text{ occurs} \quad \Leftrightarrow \quad \bigvee_{\omega \in A} (\omega = \omega_0) \; = \; \mathsf{true} \quad \Leftrightarrow \quad \omega_0 \in A$$

- Since events are sets, we can define for two events A and B:
  - $A \cup B$ occurs if A or B occurs; $A \cap B$ occurs if A and B occurs.
  - $\overline{A}$ occurs if A does not occur (i. e., if $\Omega \backslash A$ occurs).
  - A and B are *mutually exclusive*, iff $A \cap B = \emptyset$.

Event Algebra

- A family of sets $\mathcal{E} = \{E_1, \ldots, E_n\}$ is called an **event algebra**, if the following conditions hold:
  - The **certain event** Ω lies in $\mathcal{E}$.
  - If $E \in \mathcal{E}$, then $\overline{E} = \Omega \backslash E \in \mathcal{E}$.
  - If E_1 and E_2 lie in $\mathcal{E}$, then $E_1 \cup E_2 \in \mathcal{E}$ and $E_1 \cap E_2 \in \mathcal{E}$.
- If Ω is uncountable, we require the additional property:

  For a series of events $E_i \in \mathcal{E}, i \in \mathbb{N},$ the events $\bigcup_{i=1}^{\infty} E_i$ and $\bigcap_{i=1}^{\infty} E_i$ are also in $\mathcal{E}$.

  $\mathcal{E}$ is then called a **σ-algebra**.

Side remarks:

- Smallest event algebra: $\mathcal{E} = \{\emptyset, \Omega\}$
- Largest event algebra (for finite or countable Ω): $\mathcal{E} = 2^\Omega = \{A \subseteq \Omega \mid \mathsf{true}\}$

Probability Function

- Given an event algebra $\mathcal{E}$, we would like to assign every event $E \in \mathcal{E}$ its probability with a **probability function** $P: \ \mathcal{E} \to [0,1]$.
- We require P to satisfy the so-called **Kolmogorov Axioms**:
  - $\forall E \in \mathcal{E}: \ 0 \ \leq \ P(E) \ \leq \ 1$
  - $P(\Omega) \ = \ 1$
  - If $E_1, E_2 \in \mathcal{E}$ are mutually exclusive, then $P(E_1 \cup E_2) = P(E_1) + P(E_2)$.
- From these axioms one can conclude the following (incomplete) list of properties:
  - $\forall E \in \mathcal{E}: \ P(\overline{E}) \ = \ 1 - P(E)$
  - $P(\emptyset) \ = \ 0$
  - For pairwise disjoint events $E_1, E_2, \ldots \in \mathcal{E}$ holds:

$$P(\bigcup_{i=1}^{\infty} E_i) \ = \ \sum_{i=1}^{\infty} P(E_i)$$

Note that for $|\Omega| < \infty$ the union and sum are finite also.

Elementary Probabilities and Densities

Question 1: How to calculate P?

Question 2: Are there "default" event algebras?

- Idea for question 1: We have to find a way of distributing (thus the notion *distribution*) the unit mass of probability over all elements $\omega \in \Omega$.
  - If Ω is finite or countable a **probability mass function** p is used:

$$p: \ \Omega \to [0,1] \quad \text{and} \quad \sum_{\omega \in \Omega} p(\omega) = 1$$

  - If Ω is uncountable (i. e., continuous) a **probability density function** f is used:

$$f: \ \Omega \to \mathbb{R} \quad \text{and} \quad \int_{\Omega} f(\omega)\, \mathrm{d}\omega = 1$$

“Default” Event Algebras

- Idea for question 2 (“default” event algebras) we have to distinguish again between the cardinalities of Ω:
  - Ω finite or countable: $\mathcal{E} = 2^{\Omega}$
  - Ω uncountable, e. g. $\Omega = \mathbb{R}$: $\mathcal{E} = \mathcal{B}(\mathbb{R})$
- $\mathcal{B}(\mathbb{R})$ is the **Borel Algebra**, i. e., the smallest σ-algebra that contains all closed intervals $[a, b] \subset \mathbb{R}$ with $a < b$.
- $\mathcal{B}(\mathbb{R})$ also contains all open intervals and single-item sets.
- It is sufficient to note here, that all intervals are contained

$$\{[a,b]\,,\,]a,b]\,,\,]a,b[\,,\,[a,b[\,\subset \mathbb{R} \mid a < b\} \subset \mathcal{B}(\mathbb{R})$$

  because the event of a bread roll having a weight between 80 g and 90 g is represented by the interval $[80, 90]$.

Probability Spaces

- For a sample space A, an event algebra B (over A) and a probability function C, we call the triple

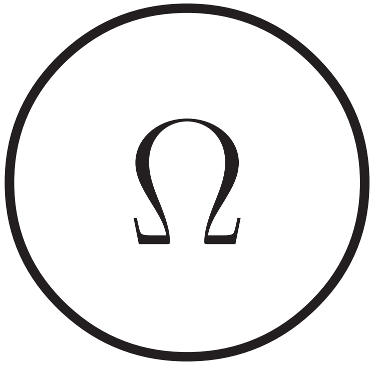
$$(A, B, C)$$

a **probability space**.

Real World

$(\Xi, \mathcal{X}, Q)$

Model



$(\Omega, \mathcal{E}, P)$

Reminder: Preimage of a Function

- Let $f: D \to M$ be a function that assigns to every value of D a value in M.
- For every value of $y \in M$ we can ask which values of $x \in D$ are mapped to y:

$$D \supseteq \{x \in D \mid f(x) = y\} \overset{\text{Def}}{=} f^{-1}(y)$$

- $f^{-1}(y)$ is called the **preimage** of y under f, denoted also as $\{f = y\}$.
- The notion can be generalized from $y \in M$ to sets $B \subseteq M$:

$$D \supseteq \{x \in D \mid f(x) \in B\} \overset{\text{Def}}{=} f^{-1}(B)$$

- If f is bijective then $\forall y \in M : \left|f^{-1}(y)\right| = 1$.
- Examples:
  - $\sin^{-1}(0) = \{k \cdot \pi \mid k \in \mathbb{Z}\}$
  - $\exp^{-1}(1) = \{0\}$
  - $\text{sgn}^{-1}(1) = (0, +\infty) \subset \mathbb{R}$

Random Variable

We still need a means of mapping real-world outcomes in Ξ to our space Ω.

- A function $X: D \to M$ is called a **random variable** iff the preimage of any value of M is an event (in some probability space).
- If X maps Ξ onto Ω, we define

$$P_X(X \in A) \;=\; Q(\{\xi \in \Xi \mid X(\xi) \in A\}).$$

- X may also map from Ω to another domain: $X: \Omega \to \mathrm{dom}(X)$.
  We then define:

$$P_X(X \in A) \;=\; P(\{\omega \in \Omega \mid X(\omega) \in A\}).$$

- If X is numeric, we call $F(x)$ with

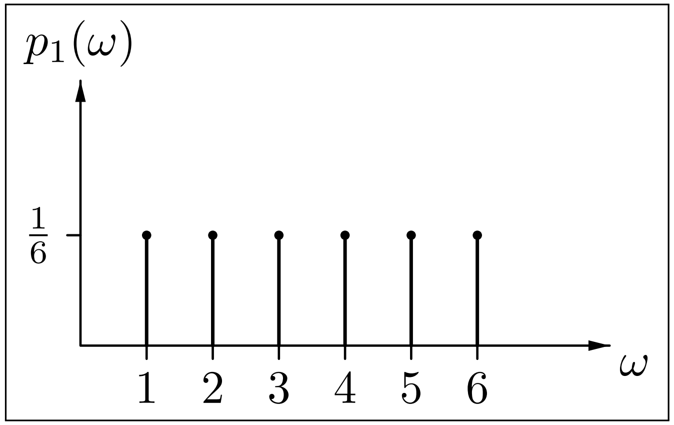
$$F(x) \;=\; P(X \leq x)$$

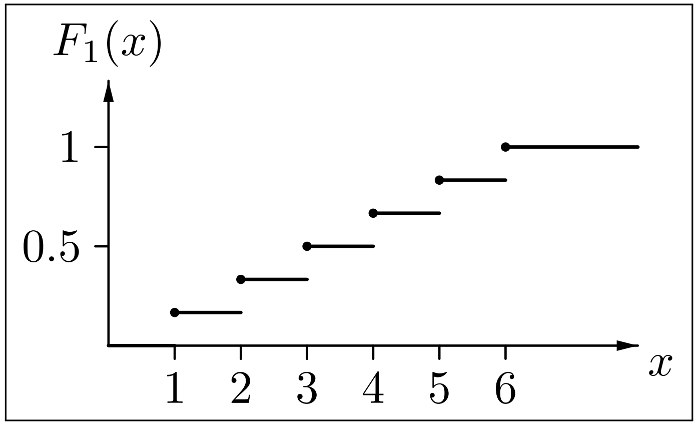
  the **distribution function** of X.

Example: Rolling a Die

$\Omega = \{1,2,3,4,5,6\} \quad X = \text{id}$

$p_1(\omega) = \frac{1}{6}$ $\qquad$ $F_1(x) = P(X \le x)$





$$\sum_{\omega\in\Omega} p_1(\omega) = \sum_{i=1}^{6} p_1(\omega_i) = \sum_{i=1}^{6} \frac{1}{6} = 1$$

$$P(X \le x) = \sum_{x' \le x} P(X = x')$$

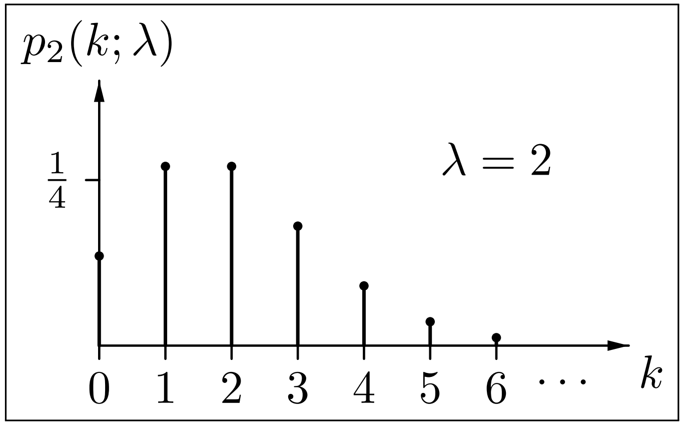
$$P(a < X \le b) = F_1(b) - F_1(a)$$

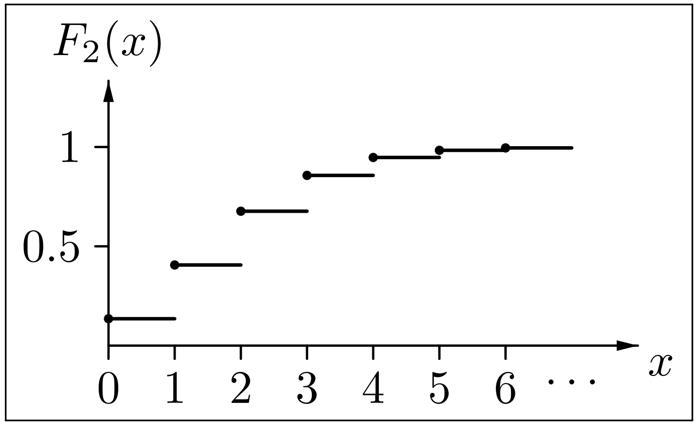
$$P(X = x) = P(\{X = x\}) = P(X^{-1}(x)) = P(\{\omega \in \Omega \mid X(\omega) = x\})$$

Example: Arriving Phone Calls

$\Omega = \mathbb{N}_0 \quad X = \mathrm{id}$

$$p_2(k;\lambda) = e^{-\lambda} \cdot \frac{\lambda^k}{k!} \qquad F_2(k;\lambda) = \sum_{i=0}^{k} e^{-\lambda} \cdot \frac{\lambda^i}{i!}$$





$$\sum_{k\in\mathbb{N}_0} p_2(k;\lambda) = \sum_{k=0}^{\infty} e^{-\lambda} \cdot \frac{\lambda^k}{k!} = e^{-\lambda} \cdot \underbrace{\sum_{k=0}^{\infty} \frac{\lambda^k}{k!}}_{=e^{\lambda}} = e^{-\lambda} \cdot e^{\lambda} = 1$$

$$P(X \le x) = \sum_{x' \le x} P(X = x')$$

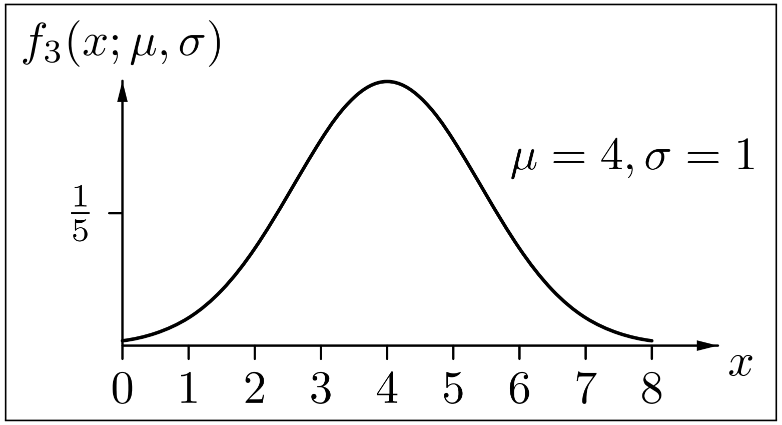
$$P(a < X \le b) = F_2(b) - F_2(a)$$

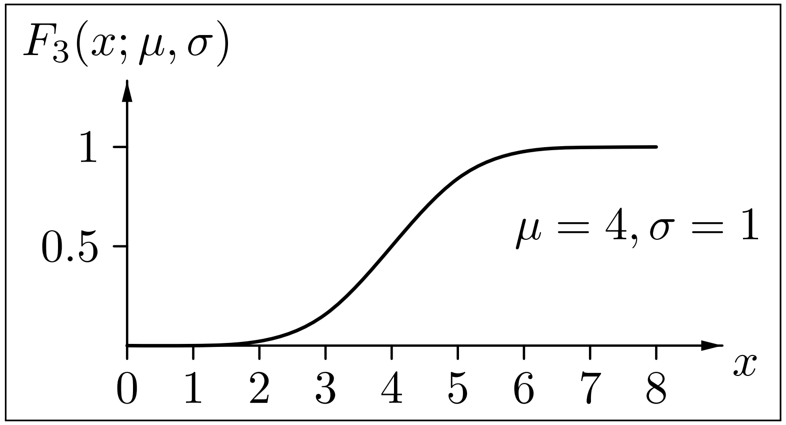
Example: Weight of a Bread Roll

$\Omega = \mathbb{R} \quad X = \mathrm{id}$

$$f_3(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$F_3(x) = \int_{-\infty}^{x} f_3(x)\,\mathrm{d}x$$



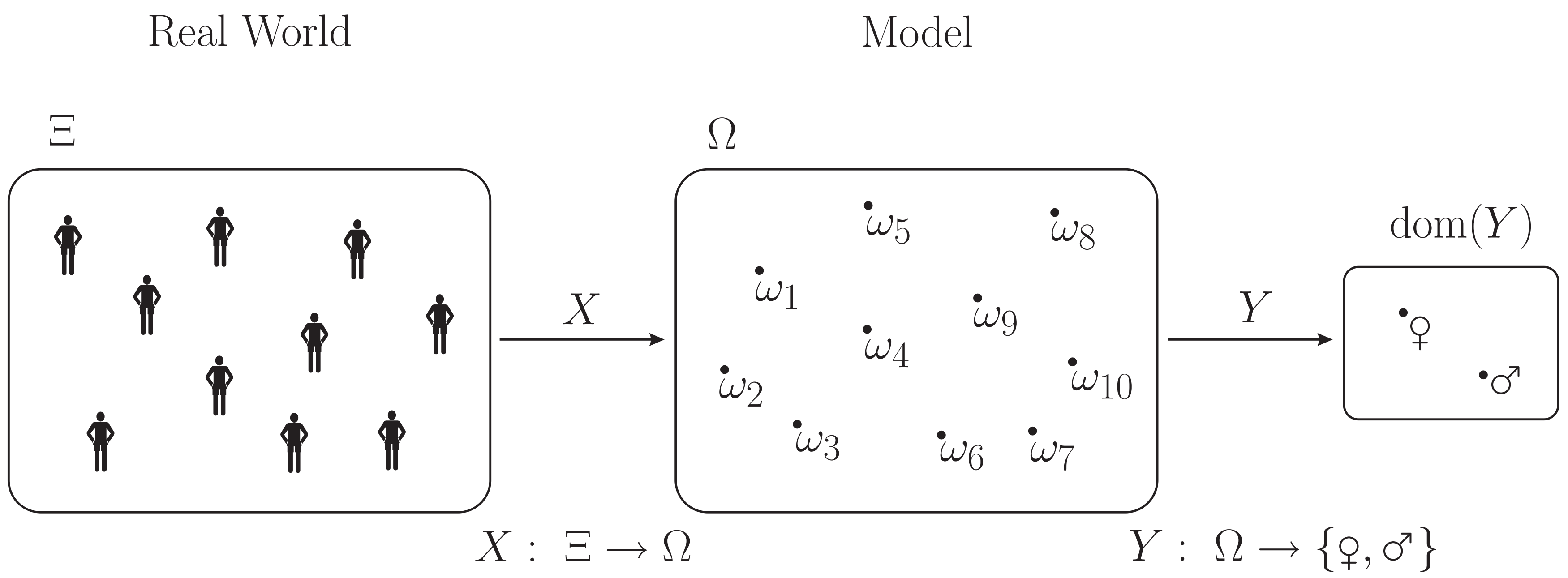


$$\int_{-\infty}^{+\infty} f_3(x)\,\mathrm{d}x = 1$$

$$\begin{aligned} P(X \le x) &= P(]-\infty, x]) \\ &= \int_{-\infty}^{x} f_3(x)\mathrm{d}x \end{aligned}$$

$$\begin{aligned} P(a < X \le b) &= P(]a, b]) \\ &= \int_{a}^{b} f_3(x)\mathrm{d}x \\ &= F_3(b) - F_3(a) \end{aligned}$$

The Big Picture



$$Q\Big(\{\xi \in \Xi \mid X(\xi) \in Y^{-1}(♀)\}\Big) \quad = \quad P\Big(\{\omega \in \Omega \mid Y(\omega) = ♀\}\Big) \quad = \quad P\Big(Y = ♀\Big) \quad = \quad P\Big(♀\Big)$$

Poisson Distribution

- Limit case of the Binomial distribution:

$$\lim_{n\to\infty} b_X(k; n, p) \;=\; \lim_{n\to\infty} \binom{n}{k} p^k (1-p)^{n-k} \;=\; e^{-\lambda} \cdot \frac{\lambda^k}{k!}$$

  with $k = 0, 1, 2, \ldots$ and $\lambda = n \cdot p$.

- Expected Value: $E(X) = \lambda$
- Variance: $V(X) = \lambda$
- Models, e. g.
  - Number of cars that pass a gate.
  - Number of customers at a register.
  - Number of calls at a call center.
- λ is the rate parameter (i. e., occurrences per unit time)

Exponential Distribution

- A continuous random variable with density function

$$f_X(x;\lambda) = \begin{cases} \lambda \cdot e^{-\lambda x} & \text{if } x \geq 0, \lambda > 0 \\ 0 & \text{otherwise} \end{cases}$$

  is **exponentially distributed**.

- Expected Value: $E(X) = \frac{1}{\lambda}$ $\qquad F_X(x;\lambda) = \begin{cases} 1 - e^{-\lambda x} & \text{if } x \geq 0, \lambda > 0 \\ 0 & \text{otherwise} \end{cases}$

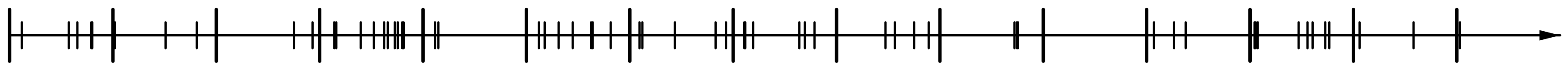
- Variance: $V(X) = \frac{1}{\lambda^2}$

- Models, e. g.
  - Lifetime of electrical devices.
  - Waiting times in a queue.
  - Time between failures of a system.

Relation between Poisson and Exponential Distributions

- Assume an arrival process with λ arrivals (per unit time, say 1h)
- The random variable that describes the **number of arrivals** within the next unit time interval is **Poisson distributed** with parameter λ.
- The random variable that describes the probability of the **waiting times between two arrivals** is **exponentially distributed** with (the same!) λ.

Example:



- Small ticks denote arrivals, large ticks mark unit time windows.
- 60 arrivals, 15 unit time windows.
- Poisson sample $\vec{x}_P = (4, 3, 2, 10, 2, 7, 5, 6, 4, 3, 0, 3, 8, 2, 1)$
- Exponential sample $\vec{x}_E = (0.1192, 0.4544, 0.0821, 0.1352, \ldots)$
- $\lambda = 4$

Inductive Statistics

Inductive Statistics: Main Tasks

- **Parameter Estimation**
  Given an assumption about the type of distribution of the underlying random variable the parameter(s) of the distribution function is estimated.
- **Hypothesis Testing**
  A hypothesis about the data generating process is tested by means of the data.
  - *Parameter Test*
    Test whether a parameter can have certain values.
  - *Goodness-of-Fit Test*
    Test whether a distribution assumption fits the data.
  - *Dependence Test*
    Test whether two attributes are dependent.
- **Model Selection**
  Among different models that can be used to explain the data the best fitting is selected, taking the complexity of the model into account.

Inductive Statistics: Random Samples

- In inductive statistics probability theory is applied to make inferences about the process that generated the data. This presupposes that the sample is the result of a random experiment, a so-called **random sample**.
- The random variable yielding the sample value x_i is denoted X_i.
  x_i is called a **instantiation** of the random variable X_i.
- A random sample $x = (x_1, \ldots, x_n)$ is an instantiation of the **random vector** $X = (X_1, \ldots, X_n)$.
- A random sample is called **independent** if the random variables $X_1, \ldots, X_n$ are (stochastically) independent, i. e. if

$$\forall c_1, \ldots, c_n \in \mathbb{R}: \quad P\left(\bigwedge_{i=1}^{n} X_i \leq c_i\right) = \prod_{i=1}^{n} P(X_i \leq c_i).$$

- An independent random sample is called **simple** if the random variables $X_1, \ldots, X_n$ have the same distribution function.

Parameter Estimation

Given:

- A data set and
- a family of parameterized distributions functions of the same type, e.g.
    - the family of binomial distributions $b_X(x; p, n)$ with the parameters p, $0 \leq p \leq 1$, and $n \in \mathbb{N}$, where n is the sample size,
    - the family of normal distributions $N_X(x; \mu, \sigma^2)$ with the parameters μ (expected value) and σ^2 (variance).

Assumption:

- The process that generated the data can be described well by an element of the given family of distribution functions.

Desired:

- The element of the given family of distribution functions (determined by its parameters) that is the best model for the data.

Parameter Estimation

- Methods that yield an estimate for a parameter are called **estimators**.
- Estimators are **statistics**, i.e. functions of the values in a sample.

  As a consequence they are functions of (instantiations of) random variables and thus (instantiations of) random variables themselves.

  Therefore we can use all of probability theory to analyze estimators.
- There are two types of parameter estimation:
  - **Point Estimators**
    Point estimators determine the best value of a parameter w.r.t. the data and certain quality criteria.
  - **Interval Estimators**
    Interval estimators yield a region, a so-called **confidence interval**, in which the true value of the parameter lies with high certainty.

Point Estimation

Not all statistics, that is, not all functions of the sample values are reasonable and useful estimator. Desirable properties are:

- **Consistency**
  With growing data volume the estimated value should get closer and closer to the true value, at least with higher and higher probability.
  Formally: If T is an estimator for the parameter θ, it should be

  $$\forall \varepsilon > 0: \quad \lim_{n \to \infty} P(|T - \theta| < \varepsilon) = 1,$$

  where n is the sample size.

- **Unbiasedness**
  An estimator should not tend to over- or underestimate the parameter.
  Rather it should yield, on average, the correct value.
  Formally this means

  $$E(T) = \theta.$$

Point Estimation

- **Efficiency**
  The estimation should be as precise as possible, that is, the deviation from the true value should be as small as possible. Formally: If T and U are two estimators for the same parameter θ, then T is called *more efficient* than U if

$$D^2(T) < D^2(U).$$

- **Sufficiency**
  An estimator should exploit all information about the parameter contained in the data. More precisely: two samples that yield the same estimate should have the same probability (otherwise there is unused information).
  Formally: an estimator T for a parameter θ is called sufficient iff for all samples $x = (x_1, \dots, x_n)$ with $T(x) = t$ the expression

$$\frac{f_{X_1}(x_1;\theta) \cdots f_{X_n}(x_n;\theta)}{f_T(t;\theta)}$$

  is independent of θ.

Point Estimation: Example

Given: a family of **uniform distributions** on the interval $[0, \theta]$, i. e.

$$f_X(x;\theta) = \begin{cases} \frac{1}{\theta}, & \text{if } 0 \leq x \leq \theta, \\ 0, & \text{otherwise.} \end{cases}$$

Desired: an estimate for the unknown parameter θ.

- We will now consider two estimators for the parameter θ and compare their properties.
  - $T = \max\{X_1, \ldots, X_n\}$
  - $U = \frac{n+1}{n} \max\{X_1, \ldots, X_n\}$
- **General approach:**
  - Find the probability density function of the estimator.
  - Check the desirable properties by exploiting this density function.

Point Estimation: Example

To analyze the estimator $T = \max\{X_1, \dots, X_n\}$, we compute its density function:

$$\begin{aligned}
f_T(t;\theta) &= \frac{\mathrm{d}}{\mathrm{d}t} F_T(t;\theta) = \frac{\mathrm{d}}{\mathrm{d}t} P(T \le t) \\
&= \frac{\mathrm{d}}{\mathrm{d}t} P(\max\{X_1, \dots, X_n\} \le t) \\
&= \frac{\mathrm{d}}{\mathrm{d}t} P\Big(\textstyle\bigwedge_{i=1}^n X_i \le t\Big) = \frac{\mathrm{d}}{\mathrm{d}t} \prod_{i=1}^n P(X_i \le t) \\
&= \frac{\mathrm{d}}{\mathrm{d}t} (F_X(t;\theta))^n = n \cdot (F_X(t;\theta))^{n-1} f_X(t,\theta)
\end{aligned}$$

where

$$F_X(x;\theta) = \int_{-\infty}^{x} f_X(x;\theta)\mathrm{d}x = \begin{cases} 0, & \text{if } x \le 0, \\ \frac{x}{\theta}, & \text{if } 0 \le x \le \theta, \\ 1, & \text{if } x \ge \theta. \end{cases}$$

Therefore it is

$$f_T(t;\theta) = \frac{n \cdot t^{n-1}}{\theta^n} \quad \text{for} \quad 0 \le t \le \theta, \quad \text{and } 0 \text{ otherwise.}$$

Point Estimation: Example

- The estimator $T = \max\{X_1, \ldots, X_n\}$ is **consistent**:

$$\begin{aligned}
\lim_{n\to\infty} P(|T-\theta| < \epsilon) &= \lim_{n\to\infty} P(T > \theta - \epsilon) \\
&= \lim_{n\to\infty} \int_{\theta-\epsilon}^{\theta} \frac{n \cdot t^{n-1}}{\theta^n} \mathrm{d}t \;=\; \lim_{n\to\infty} \left[\frac{t^n}{\theta^n}\right]_{\theta-\epsilon}^{\theta} \\
&= \lim_{n\to\infty} \left(\frac{\theta^n}{\theta^n} - \frac{(\theta-\epsilon)^n}{\theta^n}\right) \\
&= \lim_{n\to\infty} \left(1 - \left(\frac{\theta-\epsilon}{\theta}\right)^n\right) \;=\; 1
\end{aligned}$$

- It is **not unbiased**:

$$\begin{aligned}
E(T) &= \int_{-\infty}^{\infty} t \cdot f_T(t;\theta)\mathrm{d}t \;=\; \int_0^{\theta} t \cdot \frac{n \cdot t^{n-1}}{\theta^n} \mathrm{d}t \\
&= \left[\frac{n \cdot t^{n+1}}{(n+1)\theta^n}\right]_0^{\theta} \;=\; \frac{n}{n+1}\theta \;<\; \theta \quad \text{for } n < \infty.
\end{aligned}$$

Point Estimation: Example

- The estimator $U = \frac{n+1}{n}\max\{X_1, \dots, X_n\}$ has the density function

$$f_U(u;\theta) = \frac{n^{n+1}}{(n+1)^n}\frac{u^{n-1}}{\theta^n} \qquad \text{for } 0 \leq t \leq \tfrac{n+1}{n}\theta\text{, and 0 otherwise.}$$

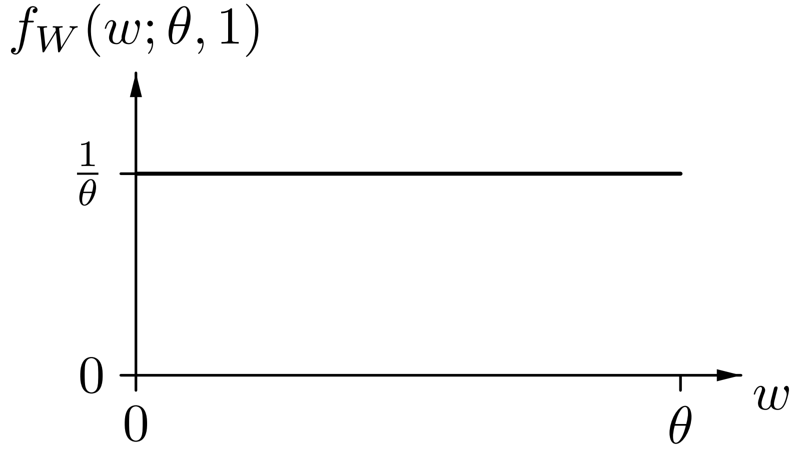
- The estimator U is **consistent** (without formal proof).

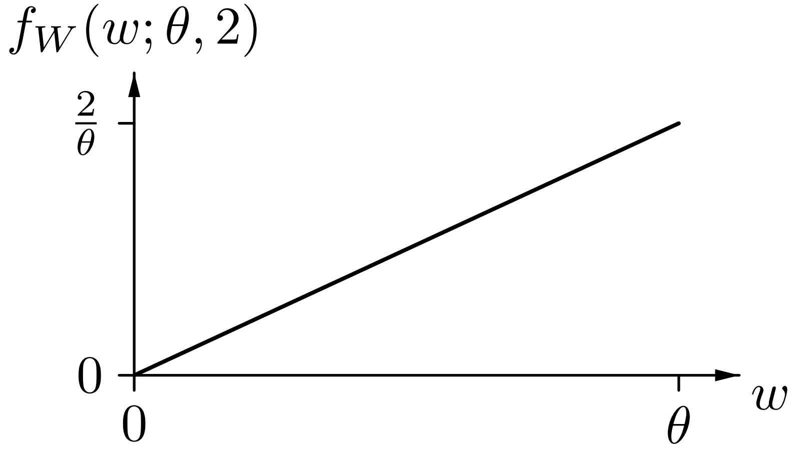
- It is **unbiased**:

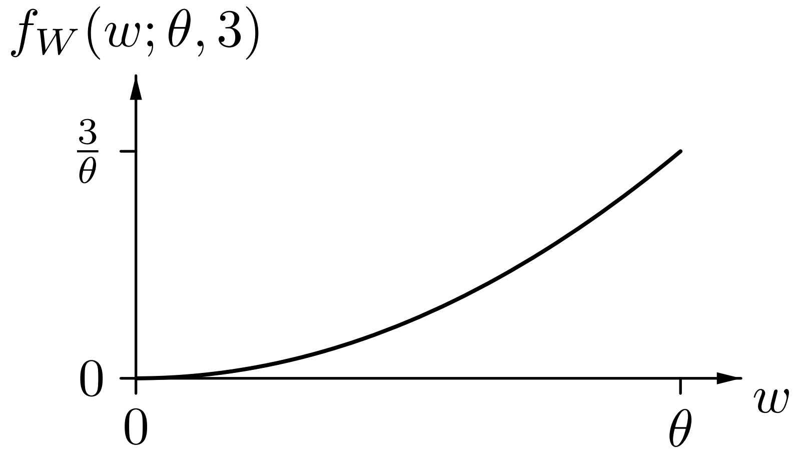
$$\begin{aligned} E(U) &= \int_{-\infty}^{\infty} u \cdot f_U(u;\theta)\mathrm{d}u \\ &= \int_0^{\frac{n+1}{n}\theta} u \cdot \frac{n^{n+1}}{(n+1)^n}\frac{u^{n-1}}{\theta^n}\mathrm{d}u \\ &= \frac{n^{n+1}}{(n+1)^n\theta^n}\left[\frac{u^{n+1}}{n+1}\right]_0^{\frac{n+1}{n}\theta} \\ &= \frac{n^{n+1}}{(n+1)^n\theta^n} \cdot \frac{1}{n+1}\left(\frac{n+1}{n}\theta\right)^{n+1} \quad = \quad \theta \end{aligned}$$

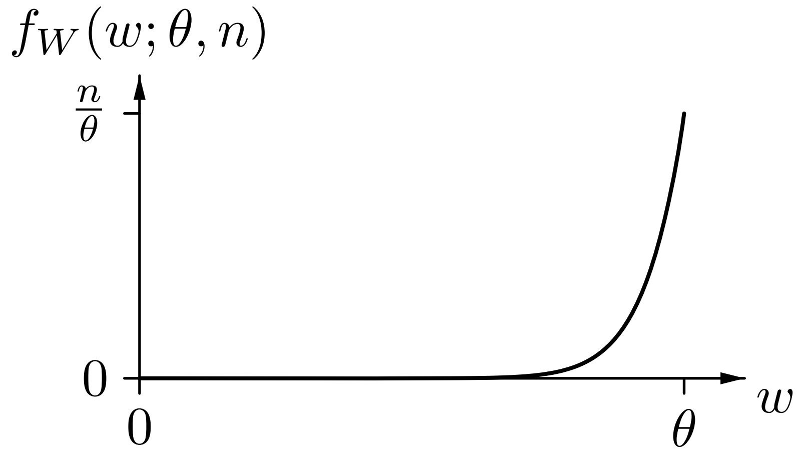
Densities of Estimators

What does the density of the estimator $W = \max\{X_1, \ldots, X_n\}$ look like? (w. r. t. n)









Note the different scales for the y-axes!

Point Estimation: Example

Given: a family of **normal distributions** $N_X(x;\mu,\sigma^2)$

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

Desired: estimates for the unknown parameters μ and σ^2.

- The median and the arithmetic mean of the sample are both consistent and unbiased estimators for the parameter μ.

  The median is less efficient than the arithmetic mean.

- The function $V^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2$ is a consistent, but **biased** estimator for the parameter σ^2 (it tends to underestimate the variance).

  The function $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$, however, is a consistent and **unbiased** estimator for σ^2 (this explains the definition of the empirical variance).

Point Estimation: Example

Given: a family of **polynomial distributions**
(synonym: multinomial distribution)

$$f_{X_1,\ldots,X_k}(x_1,\ldots,x_k;\theta_1,\ldots,\theta_k,n) = \frac{n!}{\prod_{i=1}^k x_i!}\prod_{i=1}^k \theta_i^{x_i},$$

(n is the sample size, the x_i are the frequencies of the different values a_i, $i = 1,\ldots,k$, and the θ_i are the probabilities with which the values a_i occur.)

Desired: estimates for the unknown parameters $\theta_1,\ldots,\theta_k$

- The relative frequencies $R_i = \frac{X_i}{n}$ of the different values a_i, $i = 1,\ldots,k$, are
  - consistent,
  - unbiased,
  - most efficient, and
  - sufficient estimators for the θ_i.

Polynomial Distribution: Example

- Consider the random experiment of picking a person out of a population (with replacement, technically) and determine her eye color.
- $k = 3$: $\quad a_1 \mathrel{\widehat{=}} \text{blue}, \quad a_2 \mathrel{\widehat{=}} \text{green}, \quad a_3 \mathrel{\widehat{=}} \text{brown}$
- $\theta_1 = 0.3, \quad \theta_2 = 0.3, \quad \theta_3 = 0.4$

The probability of finding 2 persons with blue eyes, 4 persons with green eyes and 4 persons with brown eyes (in a sample of size 10) is:

$$f_{X_1,X_2,X_3}(2,4,4;\,\theta_1,\theta_2,\theta_3,10) = \frac{10!}{2!4!4!} \cdot 0.3^2 \cdot 0.3^4 \cdot 0.4^4 \approx 0.0588$$

Note:

- $\sum_{i=1}^{k} x_i = n \quad \text{and} \quad \sum_{i=1}^{k} \theta_i = 1$

How Can We Find Estimators?

- Up to now we analyzed given estimators,
  now we consider the question how to find them.
- There are three main approaches to find estimators:
  - **Method of Moments**
    Derive an estimator for a parameter from the moments of a distribution and its generator function.
    (We do not consider this method here.)
  - **Maximum Likelihood Estimation**
    Choose the (set of) parameter value(s) that makes the sample most likely.
  - **Maximum A-posteriori Estimation**
    Choose a prior distribution on the range of parameter values, apply Bayes' rule to compute the posterior probability from the sample, and choose the (set of) parameter value(s) that maximizes this probability.

Maximum Likelihood Estimation

- General idea: **Choose the (set of) parameter value(s) that makes the sample most likely.**
- If the parameter value(s) were known, it would be possible to compute the probability of the sample. With unknown parameter value(s), however, it is still possible to state this probability as a function of the parameter(s).
- Formally this can be described as choosing the value θ that maximizes

  $$L(D; \theta) = f(D \mid \theta),$$

  where D are the sample data and L is called the **Likelihood Function**.
- Technically the estimator is determined by
  - setting up the likelihood function,
  - forming its partial derivative(s) w.r.t. the parameter(s), and
  - setting these derivatives equal to zero (necessary condition for a maximum).

Brief Excursion: Function Optimization

Task: Find values $\vec{x} = (x_1, \ldots, x_m)$ such that $f(\vec{x}) = f(x_1, \ldots, x_m)$ is optimal.

Often feasible approach:

- A necessary condition for a (local) optimum (maximum or minimum) is that the partial derivatives w.r.t. the parameters vanish (Pierre Fermat).
- Therefore: (Try to) solve the equation system that results from setting all partial derivatives w.r.t. the parameters equal to zero.

Example task: Minimize $\quad f(x, y) = x^2 + y^2 + xy - 4x - 5y.$

Solution procedure:

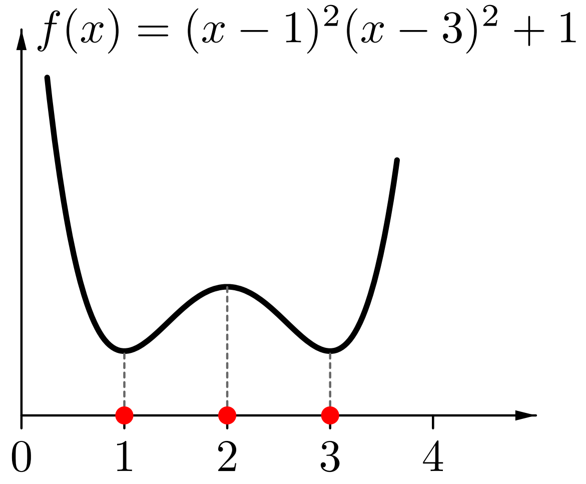
1. Take the partial derivatives of the objective function and set them to zero:

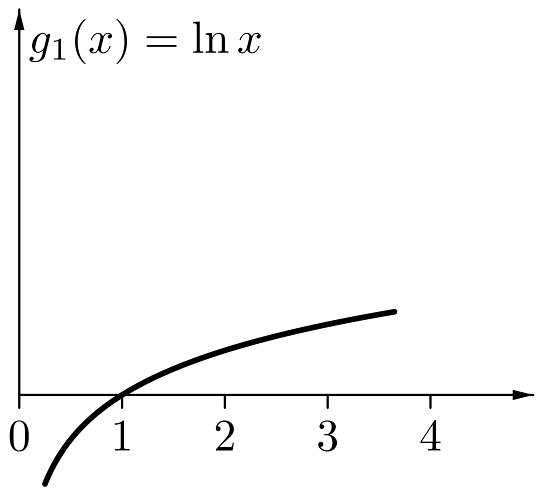
$$\frac{\partial f}{\partial x} = 2x + y - 4 = 0, \qquad \frac{\partial f}{\partial y} = 2y + x - 5 = 0.$$

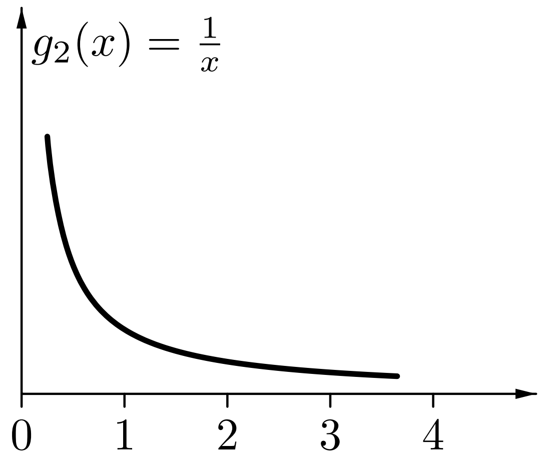
2. Solve the resulting (here: linear) equation system: $\qquad x = 1, \quad y = 2.$

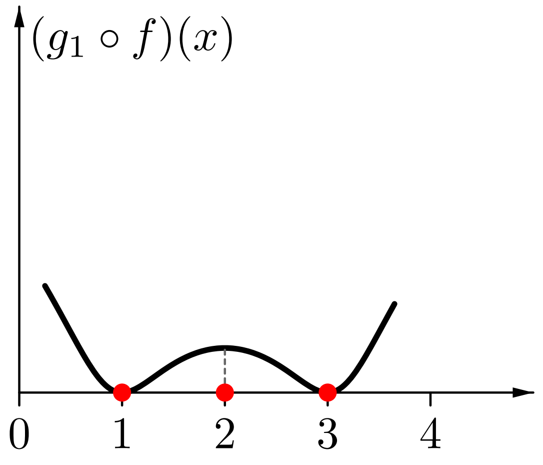
Optima of a Function

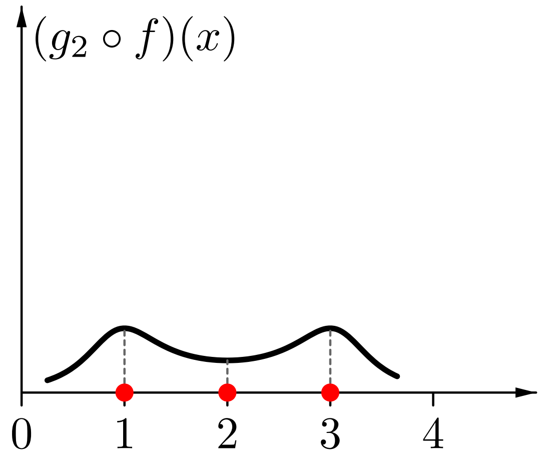
- The **locations** of the optima of a function f do not change if f is composed with a **strictly monotonic** (increasing or decreasing) function g.











Maximum Likelihood Estimation: Example

Given: a family of **normal distributions** $N_X(x; \mu, \sigma^2)$

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

Desired: estimators for the unknown parameters μ and σ^2.

The **Likelihood Function**, which describes the probability of the data, is

$$L(x_1, \ldots, x_n; \mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right).$$

To simplify the technical task of forming the partial derivatives,
we consider the natural logarithm of the likelihood function, i. e.

$$\ln L(x_1, \ldots, x_n; \mu, \sigma^2) = -n \ln\left(\sqrt{2\pi\sigma^2}\right) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2.$$

Maximum Likelihood Estimation: Example

- Estimator for the **expected value** μ:

$$\frac{\partial}{\partial \mu} \ln L(x_1, \ldots, x_n; \mu, \sigma^2) = \frac{1}{\sigma^2} \sum_{i=1}^{n} (x_i - \mu) \overset{!}{=} 0$$

$$\Rightarrow \qquad \sum_{i=1}^{n} (x_i - \mu) = \left(\sum_{i=1}^{n} x_i \right) - n\mu \overset{!}{=} 0 \qquad \Rightarrow \qquad \hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

- Estimator for the **variance** σ^2:

$$\frac{\partial}{\partial \sigma^2} \ln L(x_1, \ldots, x_n; \mu, \sigma^2) = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4} \sum_{i=1}^{n} (x_i - \mu)^2 \overset{!}{=} 0$$

$$\Rightarrow \qquad \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})^2 = \frac{1}{n} \sum_{i=1}^{n} x_i^2 - \frac{1}{n^2} \left(\sum_{i=1}^{n} x_i \right)^2 \qquad \text{(biased!)}$$

Maximum A-posteriori Estimation: Motivation

Consider the following three situations:

- A drunkard claims to be able to predict the side on which a thrown coin will land (head or tails). On ten trials he always states the correct side beforehand.
- A tea lover claims that she is able to taste whether the tea or the milk was poured into the cup first. On ten trials she always identifies the correct order.
- An expert of classical music claims to be able to recognize from a single sheet of music whether the composer was Mozart or somebody else. On ten trials he is indeed correct every time.

Maximum likelihood estimation treats all situations alike, because formally the samples are the same. However, this is implausible:

- We do not believe the drunkard at all, despite the sample data.
- We highly doubt the tea drinker, but tend to consider the data as evidence.
- We tend to believe the music expert easily.

Maximum A-posteriori Estimation

- Background knowledge about the plausible values can be incorporated by
  - using a **prior distribution** on the domain of the parameter and
  - adapting this distribution with **Bayes' rule** and the data.

- Formally maximum a-posteriori estimation is defined as follows: find the parameter value θ that maximizes

$$f(\theta \mid D) = \frac{f(D \mid \theta) f(\theta)}{f(D)} = \frac{f(D \mid \theta) f(\theta)}{\int_{-\infty}^{\infty} f(D \mid \theta') f(\theta') \mathrm{d}\theta'}$$

- As a comparison: maximum likelihood estimation maximizes

$$f(D \mid \theta)$$

- Note that $f(D)$ need not be computed: It is the same for all parameter values and since we are only interested in the value θ that maximizes $f(\theta \mid D)$ and not the *value of* $f(\theta \mid D)$, we can treat it as a normalization constant.

Maximum A-posteriori Estimation: Example

Given: a family of **binomial distributions**

$$f_X(x;\theta,n) = \binom{n}{x}\theta^x(1-\theta)^{n-x}.$$

Desired: an estimator for the unknown parameter θ.

a) **Uniform prior**: $\quad f(\theta) = 1, \qquad 0 \le \theta \le 1.$

$$f(\theta \mid D) = \gamma\binom{n}{x}\theta^x(1-\theta)^{n-x}\cdot 1 \qquad \Rightarrow \qquad \hat{\theta} = \frac{x}{n}$$

b) **Tendency towards $\frac{1}{2}$**: $\quad f(\theta) = 6\theta(1-\theta), \qquad 0 \le \theta \le 1.$

$$f(\theta \mid D) = \gamma\binom{n}{x}\theta^x(1-\theta)^{n-x}\cdot\theta(1-\theta) = \gamma\binom{n}{x}\theta^{x+1}(1-\theta)^{n-x+1}$$

$$\Rightarrow \qquad \hat{\theta} = \frac{x+1}{n+2}$$

Excursion: Dirichlet's Integral

- For computing the normalization factors of the probability density functions that occur with polynomial distributions, **Dirichlet's Integral** is helpful:

$$\int_{\theta_1} \dots \int_{\theta_k} \prod_{i=1}^{k} \theta_i^{x_i} \,\mathrm{d}\theta_1 \dots \mathrm{d}\theta_k \;=\; \frac{\prod_{i=1}^{k} \Gamma(x_i + 1)}{\Gamma(n+k)}, \qquad \text{where} \quad n = \sum_{i=1}^{k} x_i$$

  and the Γ-function is the so-called **generalized factorial**:

$$\Gamma(x) = \int_0^\infty e^{-t} t^{x-1} \mathrm{d}t, \qquad x > 0,$$

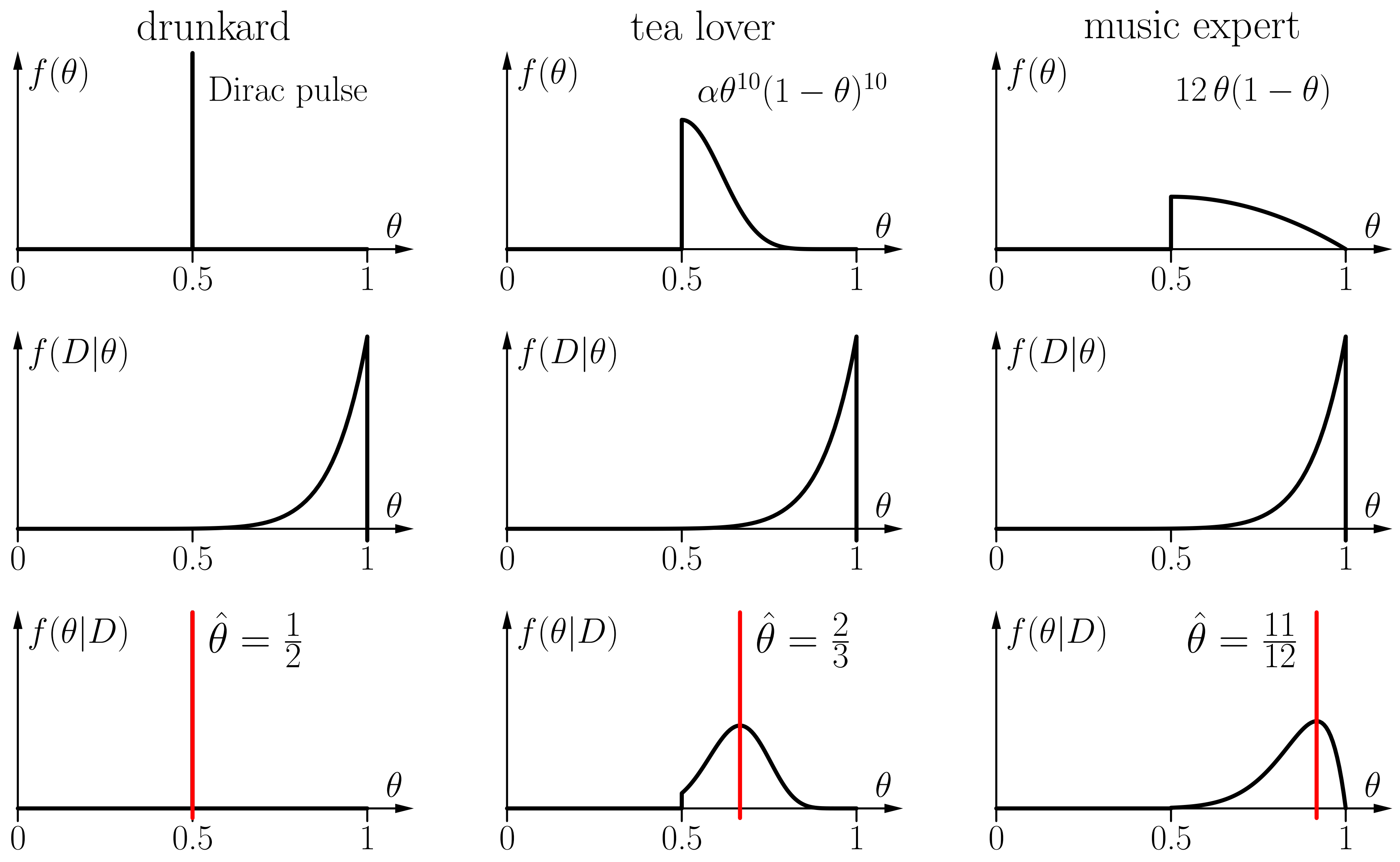
  which satisfies

$$\Gamma(x+1) = x \cdot \Gamma(x), \qquad \Gamma(\tfrac{1}{2}) = \sqrt{\pi}, \qquad \Gamma(1) = 1.$$

- **Example:** the normalization factor α for the binomial distribution prior $f(\theta) = \alpha\,\theta^2(1-\theta)^3$ is

$$\alpha \;=\; \frac{1}{\int_\theta \theta^2 (1-\theta)^3 \mathrm{d}\theta} \;=\; \frac{\Gamma(5+2)}{\Gamma(2+1)\,\Gamma(3+1)} \;=\; \frac{6!}{2!\,3!} \;=\; \frac{720}{12} \;=\; 60.$$

Maximum A-posteriori Estimation: Example



Interval Estimation

- In general the estimated value of a parameter will differ from the true value.
- It is desirable to be able to make an assertion about the possible deviations.
- The simplest possibility is to state not only a point estimate,
  but also the standard deviation of the estimator:

$$t \pm D(T) = t \pm \sqrt{D^2(T)}.$$

- A better possibility is to find intervals that contain the true value with high probability. Formally they can be defined as follows:
  Let $A = g_A(X_1, \ldots, X_n)$ and $B = g_B(X_1, \ldots, X_n)$ be two statistics, such that

$$P(A < \theta < B) = 1 - \alpha, \qquad P(\theta \leq A) = \frac{\alpha}{2}, \qquad P(\theta \geq B) = \frac{\alpha}{2}.$$

  Then the random interval $[A, B]$ (or an instantiation $[a, b]$ of this interval) is called $(1-\alpha) \cdot 100\%$ **confidence interval** for θ. The value $1-\alpha$ is called **confidence level**.

Interval Estimation

- This definition of a confidence interval is not specific enough: A and B are not uniquely determined.
- Common solution: Start from a point estimator T for the unknown parameter θ and define A and B as functions of T:

$$A = h_A(T) \quad \text{and} \quad B = h_B(T).$$

- Instead of $A \le \theta \le B$ consider the corresponding event w.r.t. the estimator T, that is, $A^* \le T \le B^*$.
- Determine $A = h_A(T)$ and $B = h_B(T)$ from the inverse functions $A^* = h_A^{-1}(\theta)$ and $B^* = h_B^{-1}(\theta)$.

$$\begin{array}{rlcl} \text{Procedure:} & P(A^* < T < B^*) & = & 1-\alpha \\ \Rightarrow & P(h_A^{-1}(\theta) < T < h_B^{-1}(\theta)) & = & 1-\alpha \\ \Rightarrow & P(h_A(T) < \theta < h_B(T)) & = & 1-\alpha \\ \Rightarrow & P(A < \theta < B) & = & 1-\alpha. \end{array}$$

Interval Estimation: Example

Given: a family of **uniform distributions** on the interval $[0, \theta]$, i.e.

$$f_X(x;\theta) = \begin{cases} \frac{1}{\theta}, & \text{if } 0 \le x \le \theta, \\ 0, & \text{otherwise.} \end{cases}$$

Desired: a confidence interval for the unknown parameter θ.

- Start from the unbiased point estimator $U = \frac{n+1}{n}\max\{X_1, \dots, X_n\}$:

$$\begin{aligned} P(U \le B^*) &= \int_0^{B^*} f_U(u;\theta)\mathrm{d}u &= \frac{\alpha}{2} \\ P(U \ge A^*) &= \int_{A^*}^{\frac{n+1}{n}\theta} f_U(u;\theta)\mathrm{d}u &= \frac{\alpha}{2} \end{aligned}$$

- From the study of point estimators we know

$$f_U(u;\theta) = \frac{n^{n+1}}{(n+1)^n}\frac{u^{n-1}}{\theta^n}.$$

Interval Estimation: Example

- Solving the integrals gives us

$$B^* = \sqrt[n]{\frac{\alpha}{2}}\,\frac{n+1}{n}\,\theta \qquad \text{and} \qquad A^* = \sqrt[n]{1-\frac{\alpha}{2}}\,\frac{n+1}{n}\,\theta,$$

  that is,

$$P\left(\sqrt[n]{\frac{\alpha}{2}}\,\frac{n+1}{n}\,\theta < U < \sqrt[n]{1-\frac{\alpha}{2}}\,\frac{n+1}{n}\,\theta\right) = 1-\alpha.$$

- Computing the inverse functions leads to

$$P\left(\frac{U}{\sqrt[n]{1-\frac{\alpha}{2}}\frac{n+1}{n}} < \theta < \frac{U}{\sqrt[n]{\frac{\alpha}{2}}\frac{n+1}{n}}\right) = 1-\alpha,$$

  that is,

$$A = \frac{U}{\sqrt[n]{1-\frac{\alpha}{2}}\frac{n+1}{n}} \qquad \text{and} \qquad B = \frac{U}{\sqrt[n]{\frac{\alpha}{2}}\frac{n+1}{n}}.$$

Interval Estimation: Common Misconceptions

- Since A and B are functions of random variables $\vec{X} = (X_1, \ldots, X_n)$ (modelling the underlying random sampling), they are random variables themselves and thus a statement like
$$P(A(\vec{X} < \theta < B(\vec{X})) = 1 - \alpha$$
makes sense.

- If, however, applied to a specific random sample $\vec{x}$ the interval borders $a = A(\vec{x})$ and $b = B(\vec{x})$ become fixed and are not random anymore.

- A probability statement about $a < \theta < b$ would be nonsensical because either $\theta \in [a, b]$ or $\theta \notin [a, b]$.

- Therefore it is incorrect to say:
"The true parameter θ lies with $(1 - \alpha) \cdot 100\%$ probability within the confidence interval."

- Correct: **"This confidence interval has been generated by a procedure which returns for $(1 - \alpha) \cdot 100\%$ of all possible samples $\vec{x}$ an interval that contains the true parameter θ."**

Interval Estimation: Common Misconceptions

Relation to sample size n and confidence level α.

- Width of a confidence interval can be considered a measure of imprecision or inaccuracy, i. e., the smaller the interval the more accurate the estimation. (although the real parameter may not be within the interval at all, of course).
- Increasing n yields a smaller interval.
- Increasing α yields a smaller interval. (Often misunderstood!)
- Example: random variable X with binomial distribution: $b_X(x; p, n)$
- Let x be the number of positive outcomes in the sample of size n. The $(1-\alpha)\cdot 100\%$ confidence interval for p reads:

$$\left[r - z_{1-\frac{\alpha}{2}} \cdot \sqrt{\frac{r(1-r)}{n-1}},\quad r + z_{1-\frac{\alpha}{2}} \cdot \sqrt{\frac{r(1-r)}{n-1}}\right] \quad \text{with} \quad r = \frac{x}{n}$$

(with $z_a = \Phi^{-1}(a)$ and Φ being the standard normal distribution function)

Hypothesis Testing

- A **hypothesis test** is a statistical procedure with which a decision is made between two contrary hypothesis about the process that generated the data.
- The two hypotheses can refer to
    - the value of a parameter **(Parameter Test)**,
    - a distribution assumption **(Goodness-of-Fit Test)**,
    - the dependence of two attributes **(Dependence Test)**.
- One of the two hypothesis is preferred, that is, in case of doubt the decision is made in its favor. (One says that it gets the "benefit of the doubt".)
- The preferred hypothesis is called the **Null Hypothesis** H_0,
  the other hypothesis is called the **Alternative Hypothesis** H_a.
- Intuitively: the null hypothesis H_0 is put on trial.
  Only if the evidence is strong enough, it is convicted (i.e. rejected).
  If there is doubt, however, it is acquitted (i.e. accepted).

Hypothesis Testing

- The test decision is based on a **test statistic**, that is, a function of the sample values.
- The null hypothesis is rejected if the value of the test statistic lies inside the so-called **critical region** C.
- Developing a hypothesis test consists in finding the critical region for a given test statistic and significance level (see below).
- The test decision may be wrong. There are two possible types of errors:

  Type 1: The null hypothesis H_0 is rejected, even though it is correct.
  Type 2: The null hypothesis H_0 is accepted, even though it is false.

- Type 1 errors are considered to be more severe, since the null hypothesis gets “the benefit of the doubt”.
- Therefore it is tried to restrict the probability of a type 1 error to a certain maximum α. This maximum value α is called **significance level**.

Parameter Test

- In a parameter test the contrary hypotheses refer to the value of a parameter, for example (one-sided test):

$$H_0: \quad \theta \geq \theta_0, \qquad\qquad H_a: \quad \theta < \theta_0.$$

- For such a test usually a point estimator T is chosen as the test statistic.
- The null hypothesis H_0 is rejected if the value t of the point estimator does not exceed a certain value c, the so-called **critical value** (i.e. $C = (-\infty, c]$).
- Formally the critical value c is determined as follows: We consider

$$\beta(\theta) = P_\theta(H_0 \text{ is rejected}) = P_\theta(T \in C),$$

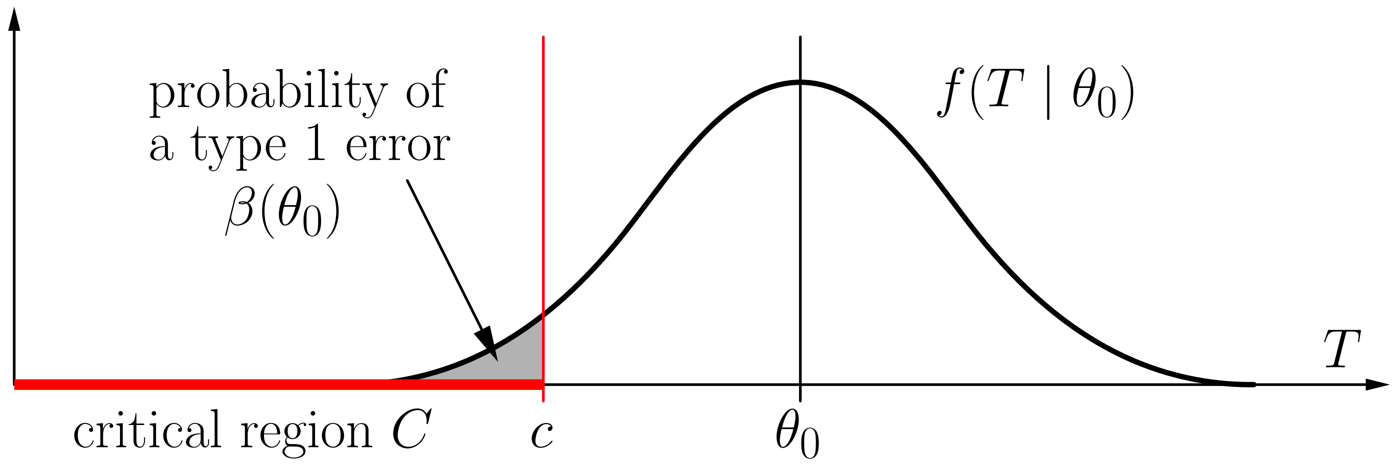
  the so-called **power** β of the test.

- The power must not exceed the significance level α for values θ satisfying H_0:

$$\max_{\theta:\theta \text{ satisfies } H_0} \beta(\theta) \leq \alpha. \qquad\qquad \text{(here:} \qquad \beta(\theta_0) \leq \alpha)$$

Parameter Test: Intuition

- The probability of a type 1 error is the area under the estimator's probability density function $f(T \mid \theta_0)$ to the left of the critical value c. (Note: This example illustrates $H_0 : \theta \geq \theta_0$ and $H_a : \theta < \theta_0$.)

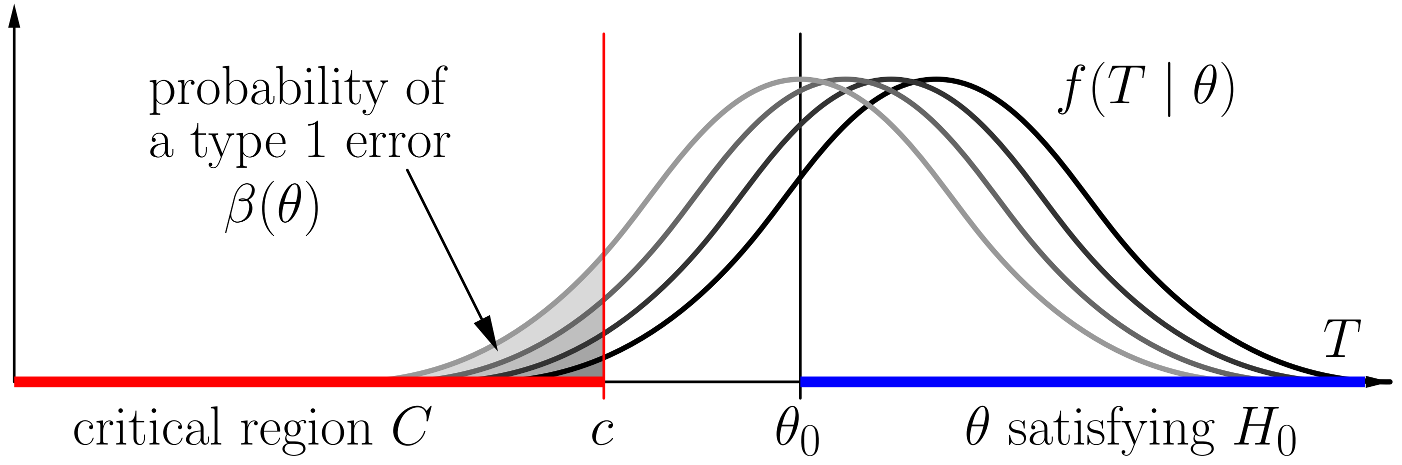


- Obviously the probability of a type 1 error depends on the location of the critical value c: higher values mean a higher error probability.

- Idea: Choose the location of the cricital value so that the maximal probability of a type 1 error equals α, the chosen significance level.

Parameter Test: Intuition

- What is so special about θ_0 that we use $f(T \mid \theta_0)$?



- In principle, all θ satisfying H_0 have to be considered, that is, all density functions $f(T \mid \theta)$ with $\theta \geq \theta_0$.

- Among these values θ, the one with the highest probability of a type 1 error (i.e., the one with the highest power $\beta(\theta)$) determines the critical value.

  Intuitively: we consider the **worst possible case**.

Parameter Test: Example

- We consider a one-sided test of the expected value μ of a normal distribution $N(\mu, \sigma^2)$ with known variance σ^2, i.e., we consider the hypotheses

$$H_0: \quad \mu \geq \mu_0, \qquad\qquad H_a: \quad \mu < \mu_0.$$

- As a test statistic we use the standard point estimator for the expected value

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

  This point estimator has the probability density

$$f_{\bar{X}}(x) = N\left(x; \mu, \frac{\sigma^2}{n}\right).$$

- Therefore it is (with the $N(0,1)$-distributed random variable Z)

$$\alpha = \beta(\mu_0) = P_{\mu_0}(\bar{X} \leq c) = P\left(\frac{\bar{X} - \mu_0}{\sigma/\sqrt{n}} \leq \frac{c - \mu_0}{\sigma/\sqrt{n}}\right) = P\left(Z \leq \frac{c - \mu_0}{\sigma/\sqrt{n}}\right).$$

Parameter Test: Example

- We have as a result that
$$\alpha = \Phi\left(\frac{c-\mu_0}{\sigma/\sqrt{n}}\right),$$
where Φ is the distribution function of the standard normal distribution.

- The distribution function Φ is tabulated, because it cannot be represented in closed form. From such a table we retrieve the value z_α satisfying $\alpha = \Phi(z_\alpha)$.

- Then the critical value is
$$c = \mu_0 + z_\alpha \frac{\sigma}{\sqrt{n}}.$$
(Note that the value of z_α is negative due to the usually small value of α. Typical values are $\alpha = 0.1$, $\alpha = 0.05$ or $\alpha = 0.01$.)

- H_0 is rejected if the value $\bar{x}$ of the point estimator $\bar{X}$ does not exceed c, otherwise it is accepted.

Parameter Test: Example

- Let $\sigma = 5.4$, $n = 25$ and $\bar{x} = 128$. We choose $\mu_0 = 130$ and $\alpha = 0.05$.
- From a standard normal distribution table we retrieve $z_{0.05} \approx -1.645$ and get

$$c_{0.05} \approx 130 - 1.645 \frac{5.4}{\sqrt{25}} \approx 128.22.$$

Since $\bar{x} = 128 < 128.22 = c$, we reject the null hypothesis H_0.

- If, however, we had chosen $\alpha = 0.01$, it would have been (with $z_{0.01} \approx -2.326$):

$$c_{0.01} \approx 130 - 2.326 \frac{5.4}{\sqrt{25}} \approx 127.49$$

Since $\bar{x} = 128 > 127.49 = c$, we would have accepted the null hypothesis H_0.

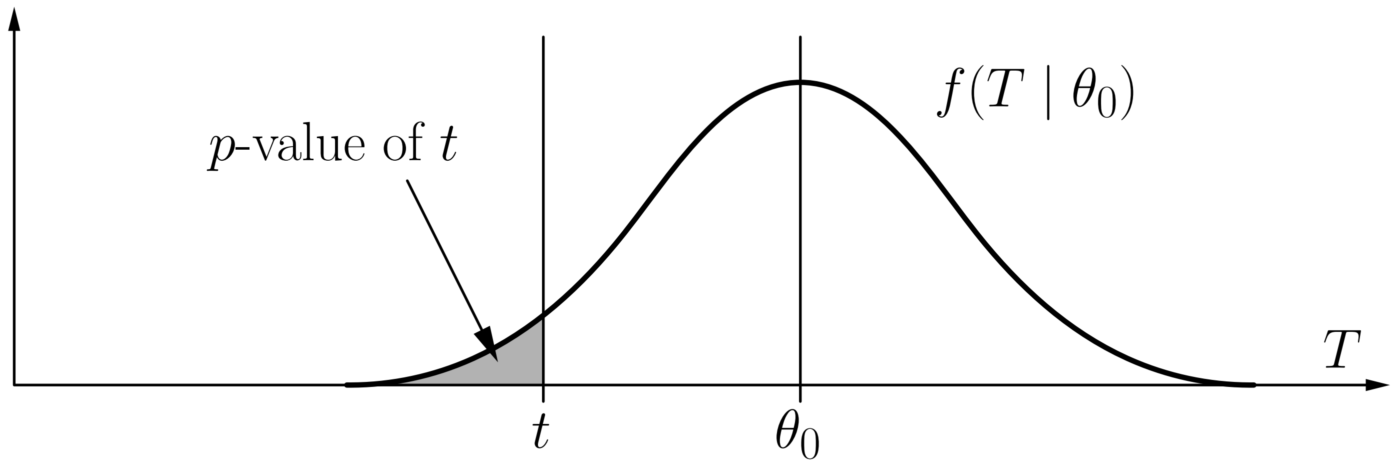
- Instead of fixing a significance level α one may state the so-called **p-value**

$$p = \Phi\left(\frac{128 - 130}{5.4/\sqrt{25}}\right) \approx 0.032.$$

For $\alpha \geq p = 0.032$ the null hypothesis is rejected, for $\alpha < p = 0.032$ accepted.

Parameter Test: p-value

- Let t be the value of the test statistic T that has been computed from a given data set. (Note: This example illustrates $H_0 : \theta \geq \theta_0$ and $H_a : \theta < \theta_0$.)



- The **p-value** is the probability that a value of t or less can be observed for the chosen test statistic T.

- The p-value is a **lower limit for the significance level** α that may have been chosen if we wanted to reject the null hypothesis H_0.

Parameter Test: p-value

Attention: p-values are often misused or misinterpreted!

- A low p-value does **not** mean that the result is very reliable!

  All that matters for the test is whether the computed p-value is **below the chosen significance level or not**.

  (A low p-value could just be a chance event, an accident!)

- The significance level may **not** be chosen **after** computing the p-value, since we tend to choose lower significance levels if we know that they are met.

  Doing so would undermine the reliability of the procedure!

- Stating p-values is only a convenient way of avoiding a fixed significance level. (Since significance levels are a matter of choice and thus user-dependent.)

  However: A significance level must still be chosen **before** a reported p-value is looked at.

Relevance of the Type-2 Error

- Reminder: There are two possible types of errors:

  Type 1: The null hypothesis H_0 is rejected, even though it is correct.

  Type 2: The null hypothesis H_0 is accepted, even though it is false.

- Type-1 errors are considered to be more severe,
  since the null hypothesis gets “the benefit of the doubt”.

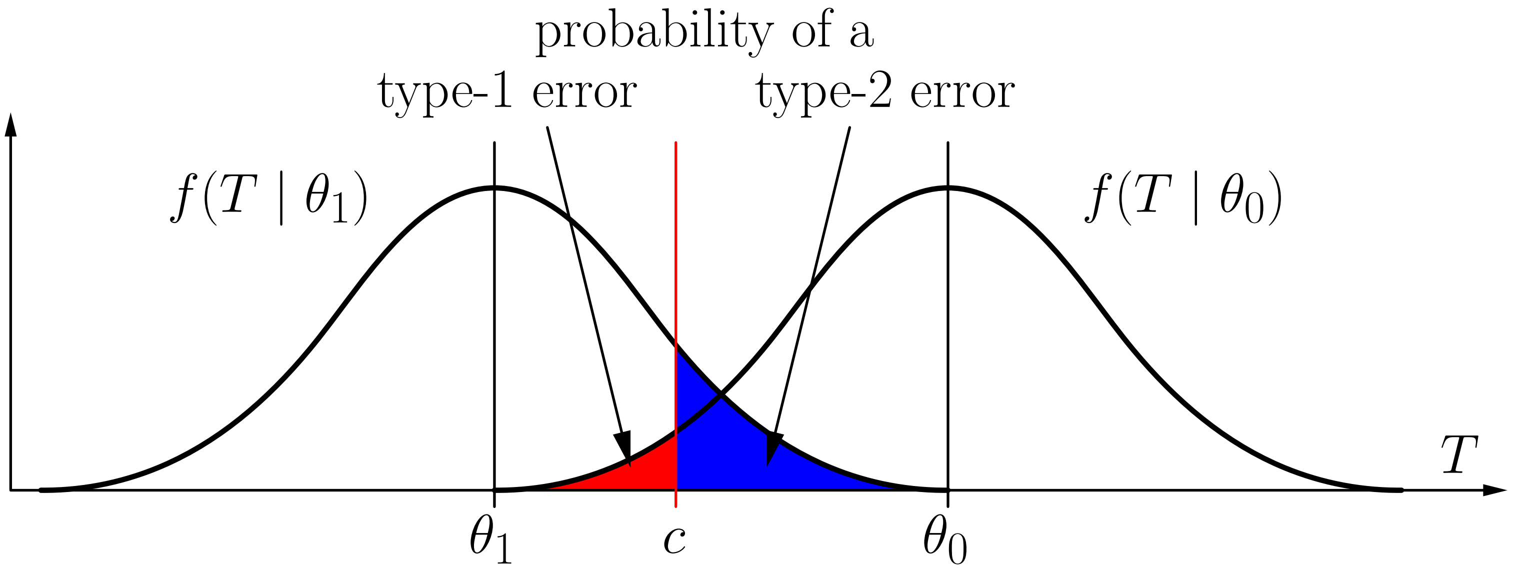
- However, **type-2 errors should not be neglected** completely:
  - It is always possible to achieve a vanishing probability of a type-1 error:
    Simply accept the null hypothesis in all instances, regardless of the data.
  - Unfortunately such an approach maximizes the type-2 error.

- Generally, **type-1 and type-2 errors are complementary quantities**:

  The lower we require the type-1 error to be (the lower the significance level),
  the higher will be the probability of a type-2 error.

Relationship between Type-1 and Type-2 Error

- Suppose there are only two possible parameter values θ_0 and θ_1 with $\theta_1 < \theta_0$. (That is, we have $H_0 : \theta = \theta_0$ and $H_a : \theta = \theta_1$.)



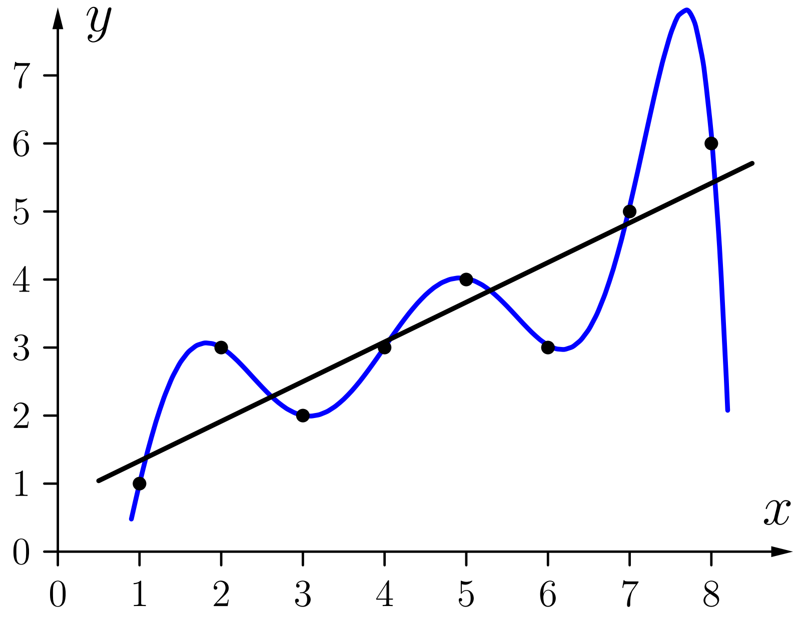
- Lowering the significance level α moves the critical value c to the left: lower type-1 error (red), but higher type-2 error (blue).
- Increasing the significance level α moves the critical value c to the right: higher type-1 error (red), but lower type-2 error (blue).

Model Selection

Model Selection

- Objective: select the model that best fits the data, **taking the model complexity into account**.

  The more complex the model, the better it usually fits the data.



black line:
regression line
(2 free parameters)

blue curve:
7th order regression polynomial
(8 free parameters)

- The blue curve fits the data points perfectly, *but it is not a good model.*

Information Criteria

- There is a **tradeoff between model complexity and fit to the data**.

  Question: How much better must a more complex model fit the data in order to justify the higher complexity?

- One approach to quantify the tradeoff: **Information Criteria**

  Let M be a model and Θ the set of free parameters of M. Then:

$$\mathrm{IC}_\kappa(M, \Theta \mid D) = -2 \ln P(D \mid M, \Theta) + \kappa|\Theta|,$$

  where D are the sample data and κ is a parameter.

  Special cases:

  - **Akaike Information Criterion** (AIC): $\kappa = 2$
  - **Bayesian Information Criterion** (BIC): $\kappa = \ln n$, where n is the sample size.

- The lower the value of these measures, the better the model.

Minimum Description Length

- Idea: Consider the transmission of the data from a sender to a receiver.

  Since the transmission of information is costly,
  the length of the message to be transmitted should be minimized.

- A good model of the data can be used to transmit the data with fewer bits.

  However, the receiver does not know the model the sender used
  and thus cannot decode the message.

  Therefore: if the sender uses a model, he/she has to transmit the model, too.

- **description length** = **length of model description**
  + **length of data description**

  (A more complex model increases the length of the model description,
  but reduces the length of the data description.)

- The model that leads to the smallest total description length is the best.

Minimum Description Length: Example

- Given: a one-dimensional sample from a polynomial distribution.
- Question: are the probabilities of the attribute values sufficiently different to justify a non-uniform distribution model?
- Coding using **no model** (equal probabilities for all values):

$$l_1 = n \log_2 k,$$

  where n is the sample size and k the number of attribute values.

- Coding using a **polynomial distribution model**:

$$l_2 = \underbrace{\log_2 \frac{(n+k-1)!}{n!(k-1)!}}_{\text{model description}} + \underbrace{\log_2 \frac{n!}{x_1! \dots x_k!}}_{\text{data description}}$$

  (Idea: Use a codebook with one page per configuration, i.e. frequency distribution (model) and specific sequence (data), and transmit the page number.)

Minimum Description Length: Example

Some details about the codebook idea:

- **Model Description**:
  There are n objects (the sample cases) that have to be partitioned into k groups (one for each attribute value). (Model: distribute n balls on k boxes.)

  $$\text{Number of possible distributions:} \qquad \frac{(n+k-1)!}{n!(k-1)!}$$

  Idea: number of possible sequences of $n+k-1$ objects (n balls and $k-1$ box walls) of which n (the objects) and $k-1$ (the box walls) are indistinguishable.

- **Data Description**:
  There are k groups of objects with x_i, $i = 1, \ldots, k$, elements in them. (The values of the x_k are known from the model description.)

  $$\text{Number of possible sequences:} \qquad \frac{n!}{x_1! \ldots x_k!}$$

Summary Statistics

Statistics has two main areas:

- **Descriptive Statistics**
  - Display the data in tables or charts.
  - Summarize the data in characteristic measures.
  - Reduce the dimensionality of the data with principal component analysis.
- **Inductive Statistics**
  - Use probability theory to draw inferences
    about the process that generated the data.
  - Parameter Estimation
  - Hypothesis Testing
  - Model Selection

A short script in German can be found at
`http://fuzzy.cs.uni-magdeburg.de/studium/ida/`

Regression

Regression

- **General Idea of Regression**
  - Method of least squares
- **Linear Regression**
  - An illustrative example
- **Polynomial Regression**
  - Generalization to polynomial functional relationships
- **Multivariate Regression**
  - Generalization to more than one function argument
- **Logistic Regression**
  - Generalization to non-polynomial functional relationships
  - An illustrative example
- **Summary**

Regression

Also known as: **Method of Least Squares** (Carl Friedrich Gauß)

Given:
- A data set of data tuples (one or more input values and one output value).
- A hypothesis about the functional relationship between output and input values.

Desired:
- A parameterization of the conjectured function that minimizes the sum of squared errors ("best fit").

Depending on

- the hypothesis about the functional relationship and
- the number of arguments to the conjectured function

different types of regression are distinguished.

Reminder: Function Optimization

Task: Find values $\vec{x} = (x_1, \dots, x_m)$ such that $f(\vec{x}) = f(x_1, \dots, x_m)$ is optimal.

Often feasible approach:

- A necessary condition for a (local) optimum (maximum or minimum) is that the partial derivatives w.r.t. the parameters vanish (Pierre Fermat).
- Therefore: (Try to) solve the equation system that results from setting all partial derivatives w.r.t. the parameters equal to zero.

Example task: Minimize $\quad f(x, y) = x^2 + y^2 + xy - 4x - 5y.$

Solution procedure:

1. Take the partial derivatives of the objective function and set them to zero:

$$\frac{\partial f}{\partial x} = 2x + y - 4 = 0, \qquad \frac{\partial f}{\partial y} = 2y + x - 5 = 0.$$

2. Solve the resulting (here: linear) equation system: $\qquad x = 1, \quad y = 2.$

Linear Regression

- Given: data set $((x_1, y_1), \ldots, (x_n, y_n))$ of n data tuples
- Conjecture: the functional relationship is linear, i.e., $y = g(x) = a + bx$.

Approach: Minimize the sum of squared errors, i.e.

$$F(a,b) = \sum_{i=1}^{n} (g(x_i) - y_i)^2 = \sum_{i=1}^{n} (a + bx_i - y_i)^2.$$

Necessary conditions for a minimum:

$$\begin{aligned} \frac{\partial F}{\partial a} &= \sum_{i=1}^{n} 2(a + bx_i - y_i) &= 0 \qquad \text{and} \\ \frac{\partial F}{\partial b} &= \sum_{i=1}^{n} 2(a + bx_i - y_i)x_i &= 0 \end{aligned}$$

Linear Regression

Result of necessary conditions: System of so-called **normal equations**, i.e.

$$na + \left(\sum_{i=1}^{n} x_i\right) b = \sum_{i=1}^{n} y_i,$$

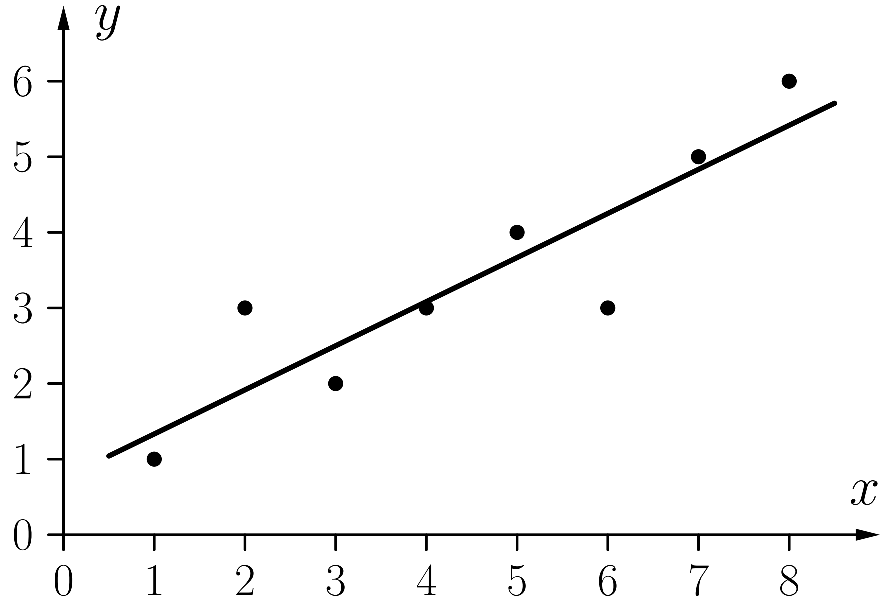
$$\left(\sum_{i=1}^{n} x_i\right) a + \left(\sum_{i=1}^{n} x_i^2\right) b = \sum_{i=1}^{n} x_i y_i.$$

- Two linear equations for two unknowns a and b.
- System can be solved with standard methods from linear algebra.
- Solution is unique unless all x-values are identical.
- The resulting line is called a **regression line**.

Linear Regression: Example

x	1	2	3	4	5	6	7	8
y	1	3	2	3	4	3	5	6

$$y = \frac{3}{4} + \frac{7}{12}x.$$



Least Squares and Maximum Likelihood

A regression line can be interpreted as a **maximum likelihood estimator:**

Assumption: The data generation process can be described well by the model

$$y = a + bx + \xi,$$

where ξ is normally distributed with mean 0 and (unknown) variance σ^2 (σ^2 independent of x, i.e. same dispersion of y for all x).

As a consequence we have

$$f(y \mid x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(y-(a+bx))^2}{2\sigma^2}\right).$$

With this expression we can set up the **likelihood function**

$$\begin{aligned} &L((x_1, y_1), \ldots (x_n, y_n); a, b, \sigma^2) \\ &= \prod_{i=1}^{n} f(x_i) f(y_i \mid x_i) \quad = \prod_{i=1}^{n} f(x_i) \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(y_i-(a+bx_i))^2}{2\sigma^2}\right). \end{aligned}$$

Least Squares and Maximum Likelihood

To simplify taking the derivatives, we compute the natural logarithm:

$$\begin{aligned}
&\ln L((x_1, y_1), \ldots (x_n, y_n); a, b, \sigma^2) \\
&= \ln \prod_{i=1}^{n} f(x_i) \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(y_i - (a + bx_i))^2}{2\sigma^2}\right) \\
&= \sum_{i=1}^{n} \ln f(x_i) \;+\; \sum_{i=1}^{n} \ln \frac{1}{\sqrt{2\pi\sigma^2}} \;-\; \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - (a + bx_i))^2
\end{aligned}$$

From this expression it becomes clear that (provided $f(x)$ is independent of a, b, and σ^2) maximizing the likelihood function is equivalent to minimizing

$$F(a, b) = \sum_{i=1}^{n} (y_i - (a + bx_i))^2.$$

Interpreting the method of least squares as a maximum likelihood estimator works also for the generalizations to polynomials and multilinear functions discussed next.

Polynomial Regression

Generalization to polynomials

$$y = p(x) = a_0 + a_1 x + \ldots + a_m x^m$$

Approach: Minimize the sum of squared errors, i.e.

$$F(a_0, a_1, \ldots, a_m) \;=\; \sum_{i=1}^{n} (p(x_i) - y_i)^2 \;=\; \sum_{i=1}^{n} (a_0 + a_1 x_i + \ldots + a_m x_i^m - y_i)^2$$

Necessary conditions for a minimum: All partial derivatives vanish, i.e.

$$\frac{\partial F}{\partial a_0} = 0, \quad \frac{\partial F}{\partial a_1} = 0, \quad \ldots, \quad \frac{\partial F}{\partial a_m} = 0.$$

Polynomial Regression

System of normal equations for polynomials

$$\begin{array}{rcrcl}
n a_0 + & \left(\sum_{i=1}^{n} x_i\right) a_1 + \ldots + & \left(\sum_{i=1}^{n} x_i^m\right) a_m & = & \sum_{i=1}^{n} y_i \\
\left(\sum_{i=1}^{n} x_i\right) a_0 + & \left(\sum_{i=1}^{n} x_i^2\right) a_1 + \ldots + & \left(\sum_{i=1}^{n} x_i^{m+1}\right) a_m & = & \sum_{i=1}^{n} x_i y_i \\
\vdots & & & & \vdots \\
\left(\sum_{i=1}^{n} x_i^m\right) a_0 + & \left(\sum_{i=1}^{n} x_i^{m+1}\right) a_1 + \ldots + & \left(\sum_{i=1}^{n} x_i^{2m}\right) a_m & = & \sum_{i=1}^{n} x_i^m y_i,
\end{array}$$

- $m + 1$ linear equations for $m + 1$ unknowns $a_0, \ldots, a_m$.
- System can be solved with standard methods from linear algebra.
- Solution is unique unless the points lie exactly on a polynomial of lower degree.

Multilinear Regression

Generalization to more than one argument

$$z = f(x, y) = a + bx + cy$$

Approach: Minimize the sum of squared errors, i.e.

$$F(a, b, c) = \sum_{i=1}^{n} (f(x_i, y_i) - z_i)^2 = \sum_{i=1}^{n} (a + bx_i + cy_i - z_i)^2$$

Necessary conditions for a minimum: All partial derivatives vanish, i.e.

$$\begin{aligned} \frac{\partial F}{\partial a} &= \sum_{i=1}^{n} 2(a + bx_i + cy_i - z_i) &= 0, \\ \frac{\partial F}{\partial b} &= \sum_{i=1}^{n} 2(a + bx_i + cy_i - z_i)x_i &= 0, \\ \frac{\partial F}{\partial c} &= \sum_{i=1}^{n} 2(a + bx_i + cy_i - z_i)y_i &= 0. \end{aligned}$$

Multilinear Regression

System of normal equations for several arguments

$$\begin{aligned}
na + \left(\sum_{i=1}^n x_i\right) b + \left(\sum_{i=1}^n y_i\right) c &= \sum_{i=1}^n z_i \\
\left(\sum_{i=1}^n x_i\right) a + \left(\sum_{i=1}^n x_i^2\right) b + \left(\sum_{i=1}^n x_i y_i\right) c &= \sum_{i=1}^n z_i x_i \\
\left(\sum_{i=1}^n y_i\right) a + \left(\sum_{i=1}^n x_i y_i\right) b + \left(\sum_{i=1}^n y_i^2\right) c &= \sum_{i=1}^n z_i y_i
\end{aligned}$$

- 3 linear equations for 3 unknowns a, b, and c.
- System can be solved with standard methods from linear algebra.
- Solution is unique unless all data points lie on a straight line.

Multilinear Regression

General multilinear case:

$$\vec{y} = f(\vec{x}_1, \dots, \vec{x}_m) = a_0 + \sum_{k=1}^{m} a_k \vec{x}_k$$

Approach: Minimize the sum of squared errors, i.e.

$$F(\vec{a}) = (\mathbf{X}\vec{a} - \vec{y})^\top (\mathbf{X}\vec{a} - \vec{y}),$$

where

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{nm} \end{pmatrix}, \qquad \vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \qquad \text{and} \qquad \vec{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix}$$

Necessary condition for a minimum:

$$\nabla_{\vec{a}} F(\vec{a}) = \nabla_{\vec{a}} (\mathbf{X}\vec{a} - \vec{y})^\top (\mathbf{X}\vec{a} - \vec{y}) = \vec{0}$$

Multilinear Regression

- $\nabla_{\vec{a}}\, F(\vec{a})$ may easily be computed by remembering that the differential operator

$$\nabla_{\vec{a}} = \left(\frac{\partial}{\partial a_0}, \dots, \frac{\partial}{\partial a_m}\right)$$

  behaves formally like a vector that is “multiplied” to the sum of squared errors.

- Alternatively, one may write out the differentiation componentwise.

Reminder: Vector Derivatives

- What is the derivative of $\vec{x}^\top\vec{x}$ w. r. t. $\vec{x}$?

$$\nabla_{\vec{x}}\vec{x}^\top\vec{x} = \left(\frac{\partial\vec{x}^\top\vec{x}}{\partial x_1}, \cdots, \frac{\partial\vec{x}^\top\vec{x}}{\partial x_m}\right)$$

- We get: $k = 1, \ldots, m$

$$\begin{aligned}\frac{\partial\vec{x}^\top\vec{x}}{\partial x_k} &= \frac{\partial}{\partial x_k}\sum_{i=1}^{m} x_i x_i \\ &= \frac{\partial}{\partial x_k}\left(x_1^2 + \cdots + x_k^2 + \cdots + x_m^2\right) \\ &= \frac{\partial}{\partial x_k}x_1^2 + \cdots + \frac{\partial}{\partial x_k}x_k^2 + \cdots + \frac{\partial}{\partial x_k}x_m^2 \\ &= 2x_k\end{aligned}$$

- Therefore we get:

$$\nabla_{\vec{x}}\vec{x}^\top\vec{x} = (2x_1, \ldots, 2x_k, \ldots, 2x_m) = 2\vec{x}$$

Multilinear Regression

With the former method we obtain for the derivative:

$$\begin{aligned}
&\nabla_{\vec{a}}\,(\mathbf{X}\vec{a}-\vec{y})^\top(\mathbf{X}\vec{a}-\vec{y}) \\
&\quad= (\nabla_{\vec{a}}\,(\mathbf{X}\vec{a}-\vec{y}))^\top\,(\mathbf{X}\vec{a}-\vec{y}) + ((\mathbf{X}\vec{a}-\vec{y})^\top\,(\nabla_{\vec{a}}\,(\mathbf{X}\vec{a}-\vec{y})))^\top \\
&\quad= (\nabla_{\vec{a}}\,(\mathbf{X}\vec{a}-\vec{y}))^\top\,(\mathbf{X}\vec{a}-\vec{y}) + (\nabla_{\vec{a}}\,(\mathbf{X}\vec{a}-\vec{y}))^\top\,(\mathbf{X}\vec{a}-\vec{y}) \\
&\quad= 2\mathbf{X}^\top(\mathbf{X}\vec{a}-\vec{y}) \\
&\quad= 2\mathbf{X}^\top\mathbf{X}\vec{a}-2\mathbf{X}^\top\vec{y} \quad = \quad \vec{0}
\end{aligned}$$

Multilinear Regression

Necessary condition for a minimum therefore:

$$\begin{aligned}\nabla_{\vec{a}} F(\vec{a}) &= \nabla_{\vec{a}} (\mathbf{X}\vec{a} - \vec{y})^\top (\mathbf{X}\vec{a} - \vec{y}) \\ &= 2\mathbf{X}^\top \mathbf{X}\vec{a} - 2\mathbf{X}^\top \vec{y} \overset{!}{=} \vec{0}\end{aligned}$$

As a consequence we get the system of **normal equations**:

$$\mathbf{X}^\top \mathbf{X}\vec{a} = \mathbf{X}^\top \vec{y}$$

This system has a unique solution if $\mathbf{X}^\top \mathbf{X}$ is not singular. Then we have

$$\vec{a} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \vec{y}.$$

$(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$ is called the (Moore–Penrose) **pseudoinverse** of the matrix $\mathbf{X}$.

With the matrix-vector representation of the regression problem an extension to **multipolynomial regression** is straighforward:
Simply add the desired products of powers to the matrix $\mathbf{X}$.

Logistic Regression

Generalization to non-polynomial functions

Idea: Find transformation to linear/polynomial case.

Simple example: The function $y = ax^b$

can be transformed into $\ln y = \ln a + b \cdot \ln x$.

Special case: **logistic function**

$$y = \frac{Y}{1 + e^{a+bx}} \qquad \Leftrightarrow \qquad \frac{1}{y} = \frac{1 + e^{a+bx}}{Y} \qquad \Leftrightarrow \qquad \frac{Y - y}{y} = e^{a+bx}.$$

Result: Apply so-called **Logit Transformation**

$$\ln\left(\frac{Y - y}{y}\right) = a + bx.$$

Logistic Regression: Example

x	1	2	3	4	5
y	0.4	1.0	3.0	5.0	5.6

Transform the data with

$$z = \ln\left(\frac{Y - y}{y}\right), \qquad Y = 6.$$

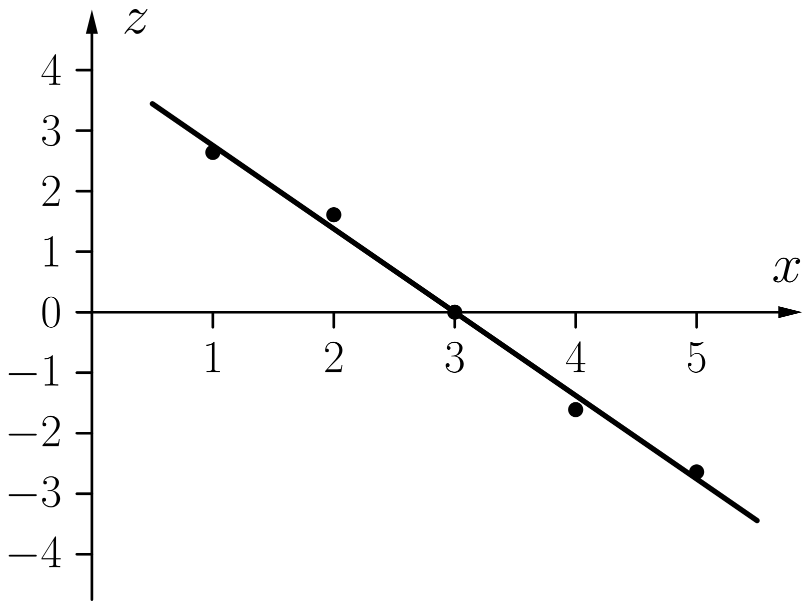
The transformed data points are

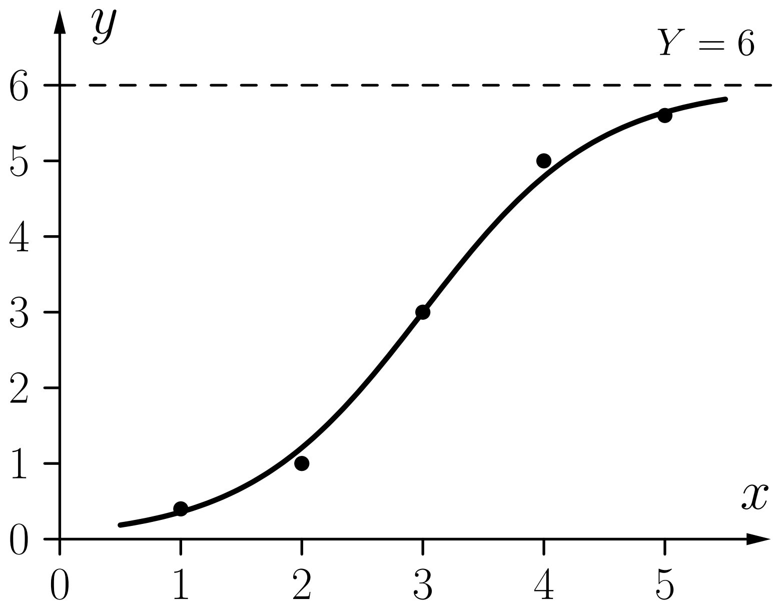
x	1	2	3	4	5
z	2.64	1.61	0.00	-1.61	-2.64

The resulting regression line is

$$z \approx -1.3775x + 4.133.$$

Logistic Regression: Example



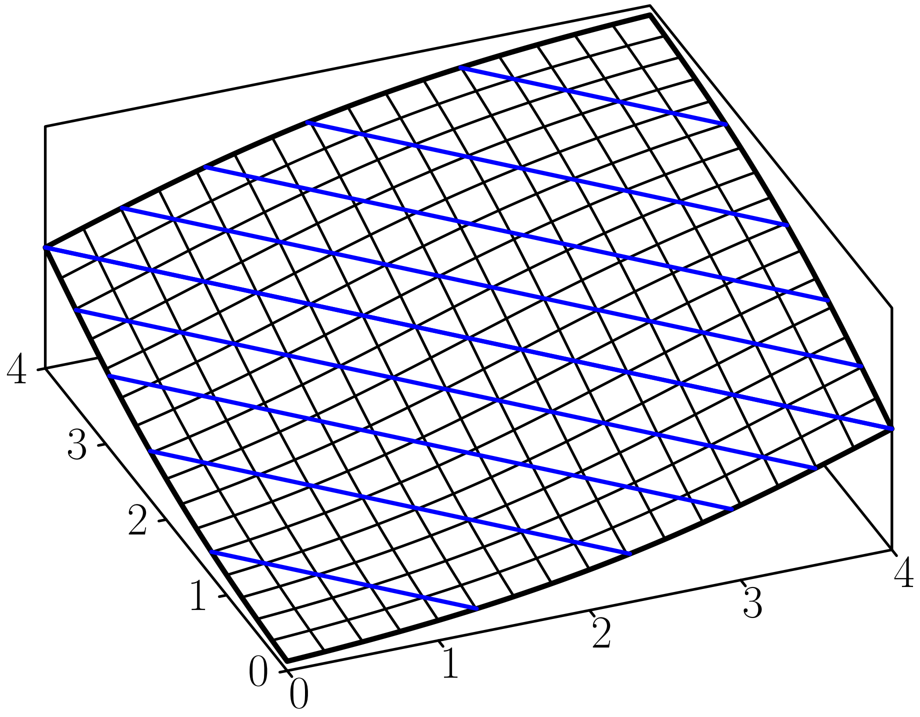


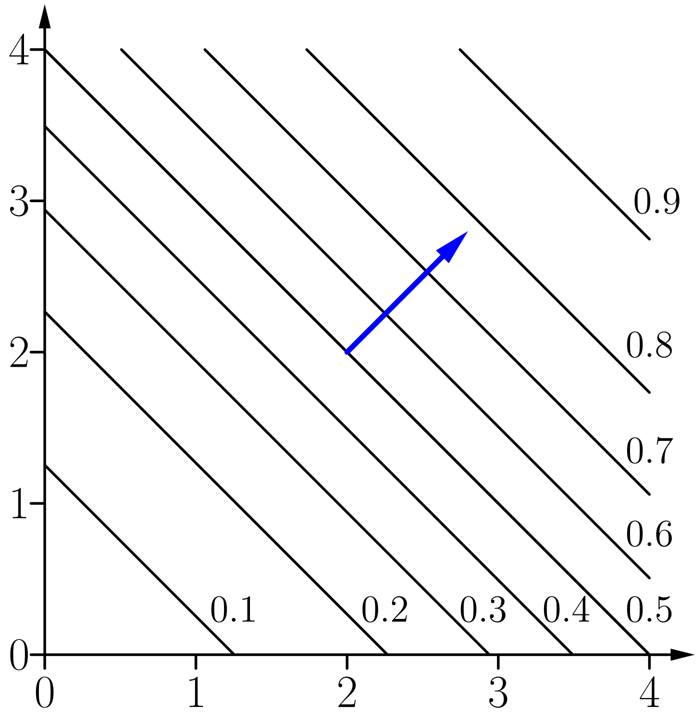
- **Attention:** The sum of squared errors is minimized only in the space the transformation maps to, not in the original space.
- Nevertheless this approach usually leads to very good results.
  The result may be improved by a gradient descent in the original space.

Logistic Regression: Two-dimensional Example

Example logistic function for two arguments x_1 and x_2:

$$y = \frac{1}{1 + \exp(4 - x_1 - x_2)} = \frac{1}{1 + \exp\Big(4 - (1,1)(x_1, x_2)^\top\Big)}$$





Logistic Regression: Two Class Problems

- Let C be a class attribute, $\mathrm{dom}(C) = \{c_1, c_2\}$, and $\vec{X}$ an m-dim. random vector. Let $P(C = c_1 \mid \vec{X} = \vec{x}) = p(\vec{x})$ and $P(C = c_2 \mid \vec{X} = \vec{x}) = 1 - p(\vec{x})$.
- **Given:** A set of data points $\mathbf{X} = \{\vec{x}_1, \ldots, \vec{x}_n\}$ (realizations of $\vec{X}$), each of which belongs to one of the two classes c_1 and c_2.
- **Desired:** A simple description of the function $p(\vec{x})$.
- **Approach:** Describe p by a logistic function:

$$p(\vec{x}) = \frac{1}{1 + e^{a_0 + \vec{a}\vec{x}}} = \frac{1}{1 + \exp\left(a_0 + \sum_{i=1}^{m} a_i x_i\right)}$$

Apply logit transformation to $p(x)$:

$$\ln\left(\frac{1 - p(\vec{x})}{p(\vec{x})}\right) = a_0 + \vec{a}\vec{x} = a_0 + \sum_{i=1}^{m} a_i x_i$$

The values $p(\vec{x}_i)$ may be obtained by kernel estimation.

Kernel Estimation

- **Idea:** Define an "influence function" (kernel), which describes how strongly a data point influences the probability estimate for neighboring points.
- Common choice for the kernel function: **Gaussian function**

$$K(\vec{x},\vec{y}) \;=\; \frac{1}{(2\pi\sigma^2)^{\frac{m}{2}}} \exp\left(-\frac{(\vec{x}-\vec{y})^\top(\vec{x}-\vec{y})}{2\sigma^2}\right)$$

- Kernel estimate of probability density given a data set $\mathcal{X} = \{\vec{x}_1,\ldots,\vec{x}_n\}$:

$$\hat{f}(\vec{x}) \;=\; \frac{1}{n}\sum_{i=1}^{n} K(\vec{x},\vec{x}_i).$$

- Kernel estimation applied to a two class problem:

$$\hat{p}(\vec{x}) \;=\; \frac{\sum_{i=1}^{n} c(\vec{x}_i)K(\vec{x},\vec{x}_i)}{\sum_{i=1}^{n} K(\vec{x},\vec{x}_i)}.$$

(It is $c(\vec{x}_i) = 1$ if x_i belongs to class c_1 and $c(\vec{x}_i) = 0$ otherwise.)

Summary Regression

- **Minimize the Sum of Squared Errors**
  - Write the sum of squared errors
    as a function of the parameters to be determined.
- **Exploit Necessary Conditions for a Minimum**
  - Partial derivatives w.r.t. the parameters to determine must vanish.
- **Solve the System of Normal Equations**
  - The best fit parameters are the solution of the system of normal equations.
- **Non-polynomial Regression Functions**
  - Find a transformation to the multipolynomial case.
  - Logistic regression can be used to solve two class classification problems.

Bayes Classifiers

Bayes Classifiers

- **Probabilistic Classification and Bayes' Rule**
- **Naive Bayes Classifiers**
  - Derivation of the classification formula
  - Probability estimation and Laplace correction
  - Simple examples of naive Bayes classifiers
  - A naive Bayes classifier for the Iris data
- **Full Bayes Classifiers**
  - Derivation of the classification formula
  - Comparison to naive Bayes classifiers
  - A simple example of a full Bayes classifier
  - A full Bayes classifier for the Iris data
- **Summary**

Probabilistic Classification

- A classifier is an algorithm that assigns a class from a predefined set to a case or object, based on the values of descriptive attributes.
- An optimal classifier maximizes the probability of a correct class assignment.
  - Let C be a class attribute with $\text{dom}(C) = \{c_1, \ldots, c_{n_C}\}$, which occur with probabilities p_i, $1 \leq i \leq n_C$.
  - Let q_i be the probability with which a classifier assigns class c_i. ($q_i \in \{0, 1\}$ for a deterministic classifier)
  - The probability of a correct assignment is

$$P(\text{correct assignment}) = \sum_{i=1}^{n_C} p_i q_i.$$

  - Therefore the best choice for the q_i is

$$q_i = \begin{cases} 1, & \text{if } p_i = \max_{k=1}^{n_C} p_k, \\ 0, & \text{otherwise.} \end{cases}$$

Probabilistic Classification

- Consequence: An optimal classifier should assign the **most probable class**.
- This argument does not change if we take descriptive attributes into account.
  - Let $U = \{A_1, \ldots, A_m\}$ be a set of descriptive attributes with domains $\mathrm{dom}(A_k)$, $1 \leq k \leq m$.
  - Let $A_1 = a_1, \ldots, A_m = a_m$ be an instantiation of the descriptive attributes.
  - An optimal classifier should assign the class c_i for which

$$P(C = c_i \mid A_1 = a_1, \ldots, A_m = a_m) = \\ \max\nolimits_{j=1}^{n_C} \; P(C = c_j \mid A_1 = a_1, \ldots, A_m = a_m)$$

- **Problem:** We cannot store a class (or the class probabilities) for every possible instantiation $A_1 = a_1, \ldots, A_m = a_m$ of the descriptive attributes. (The table size grows exponentially with the number of attributes.)
- Therefore: **Simplifying assumptions are necessary.**

Bayes' Rule and Bayes' Classifiers

- Bayes' rule is a formula that can be used to "invert" conditional probabilities: Let X and Y be events, $P(X) > 0$. Then

$$P(Y \mid X) = \frac{P(X \mid Y) \cdot P(Y)}{P(X)}.$$

- Bayes' rule follows directly from the definition of conditional probability:

$$P(Y \mid X) = \frac{P(X \cap Y)}{P(X)} \qquad \text{and} \qquad P(X \mid Y) = \frac{P(X \cap Y)}{P(Y)}.$$

- Bayes' classifiers: Compute the class probabilities as

$$P(C = c_i \mid A_1 = a_1, \ldots, A_m = a_m) =$$
$$\frac{P(A_1 = a_1, \ldots, A_m = a_m \mid C = c_i) \cdot P(C = c_i)}{P(A_1 = a_1, \ldots, A_m = a_m)}.$$

- Looks unreasonable at first sight: Even more probabilities to store.

Naive Bayes Classifiers

Naive Assumption:
The descriptive attributes are conditionally independent given the class.

Bayes' Rule:

$$P(C = c_i \mid \omega) = \frac{P(A_1 = a_1, \ldots, A_m = a_m \mid C = c_i) \cdot P(C = c_i)}{P(A_1 = a_1, \ldots, A_m = a_m)} \quad \leftarrow p_0$$

abbrev. for the normalizing constant

Chain Rule of Probability:

$$P(C = c_i \mid \omega) = \frac{P(C = c_i)}{p_0} \cdot \prod_{k=1}^{m} P(A_k = a_k \mid A_1 = a_1, \ldots, A_{k-1} = a_{k-1}, C = c_i)$$

Conditional Independence Assumption:

$$P(C = c_i \mid \omega) = \frac{P(C = c_i)}{p_0} \cdot \prod_{k=1}^{m} P(A_k = a_k \mid C = c_i)$$

Reminder: Chain Rule of Probability

- Based on the **product rule** of probability:

$$P(A \wedge B) = P(A \mid B) \cdot P(B)$$

  (Multiply definition of conditional probability with $P(B)$.)

- **Multiple application** of the product rule yields:

$$\begin{aligned} P(A_1, \ldots, A_m) &= P(A_m \mid A_1, \ldots, A_{m-1}) \cdot P(A_1, \ldots, A_{m-1}) \\ &= P(A_m \mid A_1, \ldots, A_{m-1}) \\ &\cdot\ P(A_{m-1} \mid A_1, \ldots, A_{m-2}) \cdot P(A_1, \ldots, A_{m-2}) \\ &= \vdots \\ &= \prod_{k=1}^{m} P(A_k \mid A_1, \ldots, A_{k-1}) \end{aligned}$$

- The scheme works also if there is already a condition in the original expression:

$$P(A_1, \ldots, A_m \mid C) = \prod_{i=1}^{m} P(A_k \mid A_1, \ldots, A_{k-1}, C)$$

Conditional Independence

- Reminder: **stochastic independence** (unconditional)

$$P(A \wedge B) = P(A) \cdot P(B)$$

(Joint probability is the product of the individual probabilities.)

- Comparison to the **product rule**

$$P(A \wedge B) = P(A \mid B) \cdot P(B)$$

shows that this is equivalent to

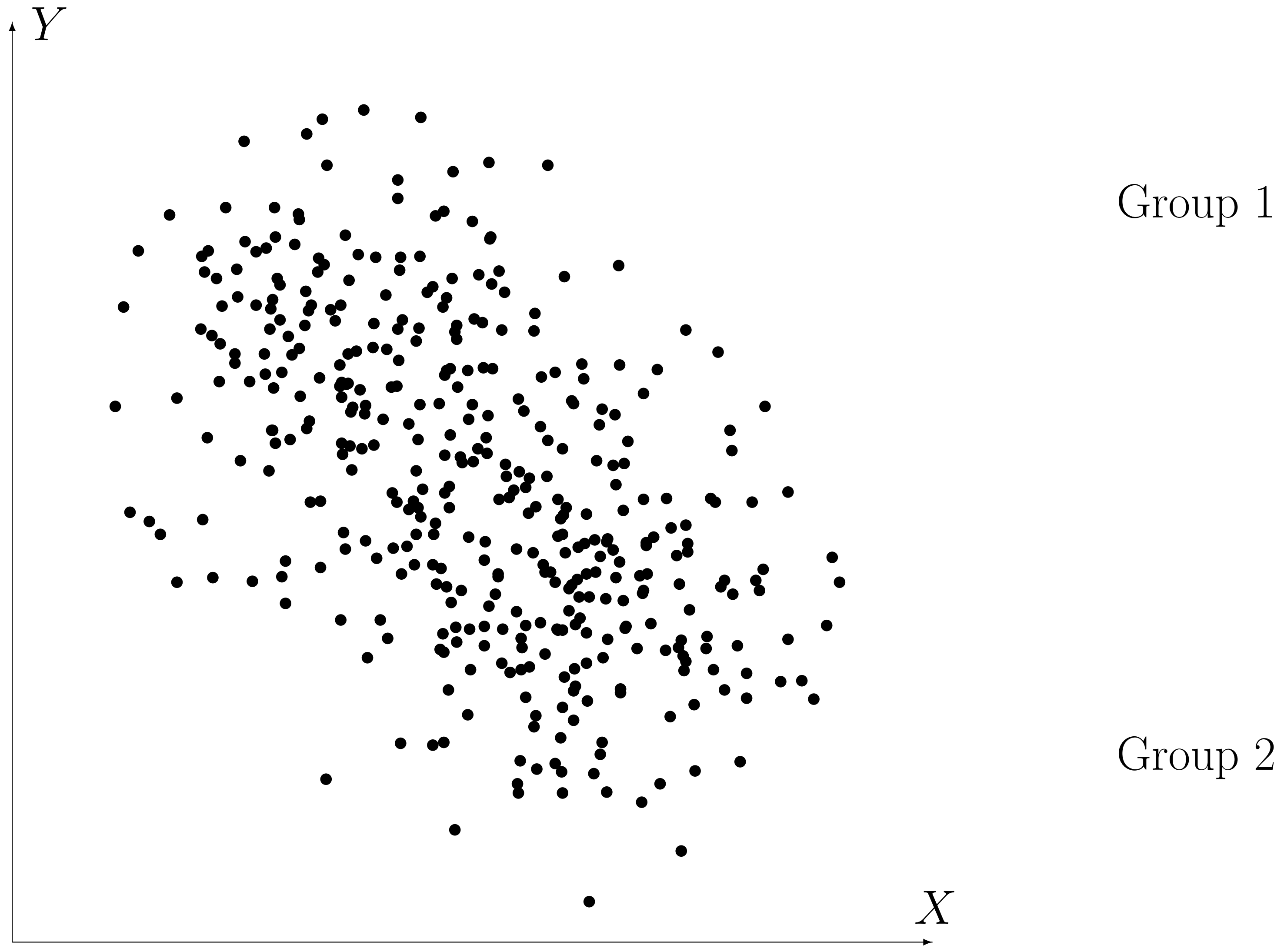
$$P(A \mid B) = P(A)$$

- The same formulae hold conditionally, i.e.

$$\begin{aligned} P(A \wedge B \mid C) &= P(A \mid C) \cdot P(B \mid C) \qquad \text{and} \\ P(A \mid B, C) &= P(A \mid C). \end{aligned}$$

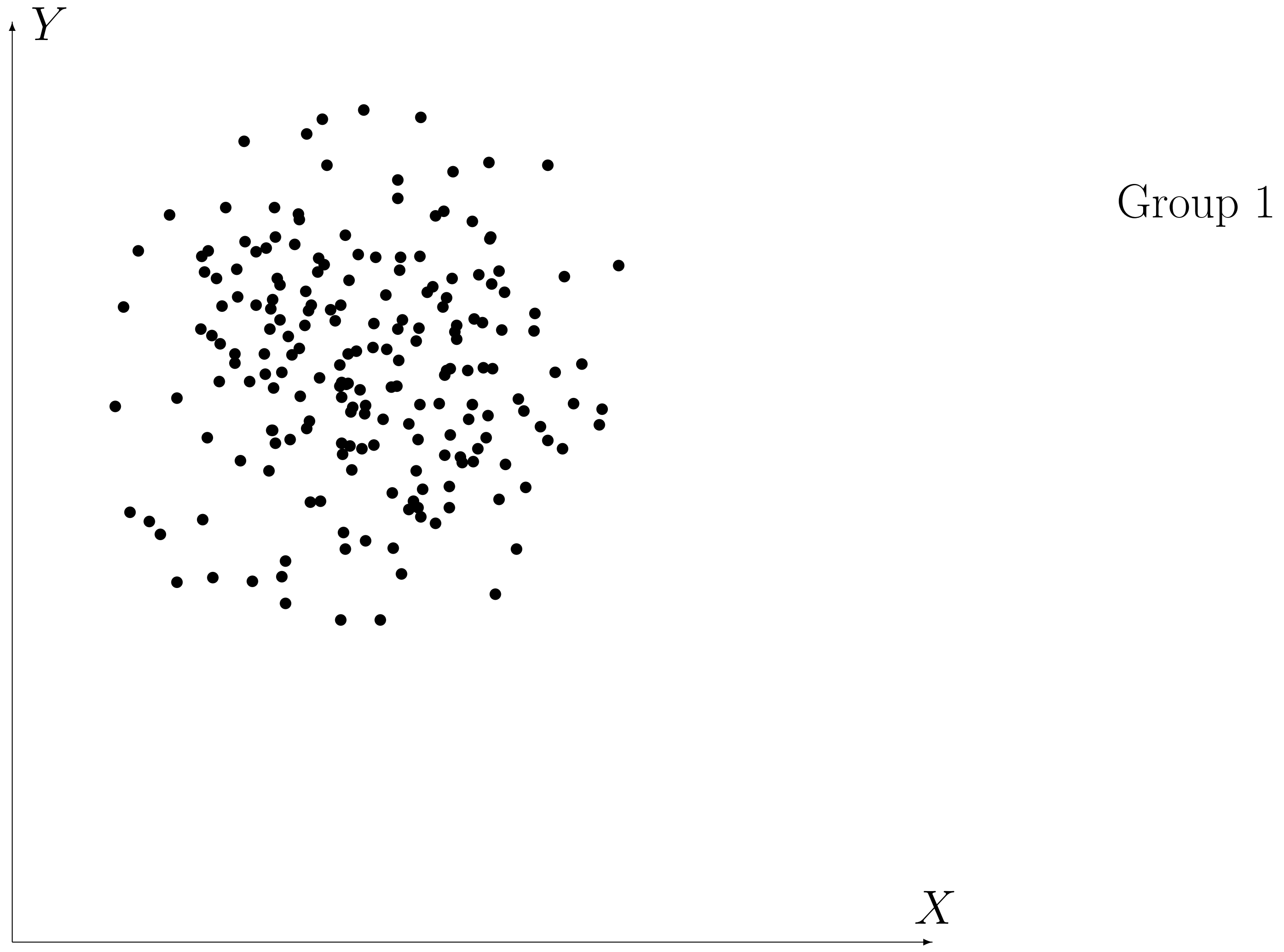
- **Conditional independence allows us to cancel some conditions**.

Conditional Independence: An Example



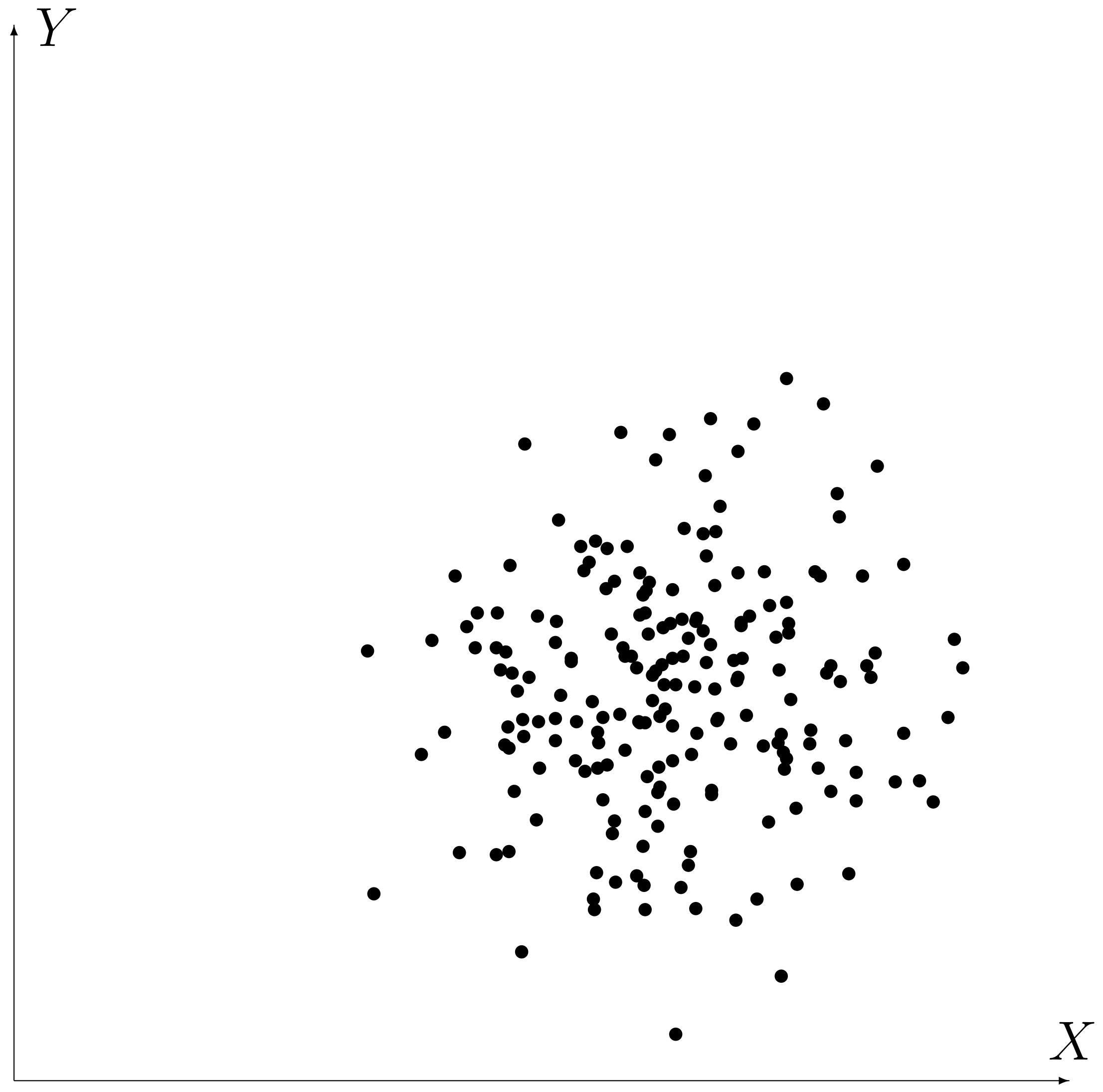
(Weak) Dependence in the entire dataset: X and Y dependent.

Conditional Independence: An Example



No Dependence in Group 1: X and Y conditionally independent given Group 1.

Conditional Independence: An Example



Group 2

No Dependence in Group 2: X and Y conditionally independent given Group 2.

Marginal & Conditional Independence

- The next table shows four 3-dimensional probability distributions (one per row).
- The (in)dependencies are always w. r. t. A and B.
- The condition variable is C.

		a_1				a_2			
		b_1		b_2		b_1		b_2	
marginal	conditional	c_1	c_2	c_1	c_2	c_1	c_2	c_1	c_2
indep.	indep.	0.03	0.01	0.27	0.09	0.006	0.054	0.054	0.486
dep.	dep.	0.01	0.03	0.126	0.234	0.1275	0.3825	0.0315	0.0585
dep.	indep.	0.12	0.085	0.18	0.015	0.024	0.459	0.036	0.081
indep.	dep.	0.008	0.032	0.144	0.216	0.018	0.042	0.054	0.486

- All combinations are possible.

Naive Bayes Classifiers

- Consequence: Manageable amount of data to store.
  Store distributions $P(C = c_i)$ and $\forall 1 \leq k \leq m : P(A_k = a_k \mid C = c_i)$.
- It is not necessary to compute p_0 explicitely, because it can be computed implicitly by normalizing the computed values to sum 1.

Estimation of Probabilities:

- **Nominal/Symbolic Attributes**

$$\hat{P}(A_k = a_k \mid C = c_i) = \frac{\#(A_k = a_k, C = c_i) + \gamma}{\#(C = c_i) + n_{A_k}\gamma}$$

  γ is called **Laplace correction**: Assume for every class c_i some number of hypothetical samples for every value of A_k to prevent the estimate to be 0 if $\#(A_k = a_k, C = c_i) = 0$.

  $\gamma = 0$: Maximum likelihood estimation.

  Common choices: $\gamma = 1$ or $\gamma = \frac{1}{2}$.

Naive Bayes Classifiers

Estimation of Probabilities:

- **Metric/Numeric Attributes:** Assume a normal distribution.

$$P(A_k = a_k \mid C = c_i) = \frac{1}{\sqrt{2\pi}\sigma_k(c_i)} \exp\left(-\frac{(a_k - \mu_k(c_i))^2}{2\sigma_k^2(c_i)}\right)$$

- Estimate of mean value

$$\hat{\mu}_k(c_i) = \frac{1}{\#(C = c_i)} \sum_{j=1}^{\#(C=c_i)} a_k(j)$$

- Estimate of variance

$$\hat{\sigma}_k^2(c_i) = \frac{1}{\xi} \sum_{j=1}^{\#(C=c_i)} (a_k(j) - \hat{\mu}_k(c_i))^2$$

$\xi = \#(C = c_i)$: Maximum likelihood estimation
$\xi = \#(C = c_i) - 1$: Unbiased estimation

Naive Bayes Classifiers: Simple Example 1

No	Sex	Age	Blood pr.	Drug
1	male	20	normal	A
2	female	73	normal	B
3	female	37	high	A
4	male	33	low	B
5	female	48	high	A
6	male	29	normal	A
7	female	52	normal	B
8	male	42	low	B
9	male	61	normal	B
10	female	30	normal	A
11	female	26	low	B
12	male	54	high	A

$P(\text{Drug})$	A	B
	0.5	0.5

$P(\text{Sex} \mid \text{Drug})$	A	B
male	0.5	0.5
female	0.5	0.5

$P(\text{Age} \mid \text{Drug})$	A	B
μ	36.3	47.8
σ^2	161.9	311.0

$P(\text{Blood Pr.} \mid \text{Drug})$	A	B
low	0	0.5
normal	0.5	0.5
high	0.5	0

A simple database and estimated (conditional) probability distributions.

Naive Bayes Classifiers: Simple Example 1

$$
\begin{aligned}
&P(\text{Drug A} \mid \text{male, 61, normal}) \\
&\quad = \; c_1 \cdot P(\text{Drug A}) \cdot P(\text{male} \mid \text{Drug A}) \cdot P(61 \mid \text{Drug A}) \cdot P(\text{normal} \mid \text{Drug A}) \\
&\quad \approx \; c_1 \cdot 0.5 \cdot 0.5 \cdot 0.004787 \cdot 0.5 \quad = \quad c_1 \cdot 5.984 \cdot 10^{-4} \quad = \quad 0.219
\end{aligned}
$$

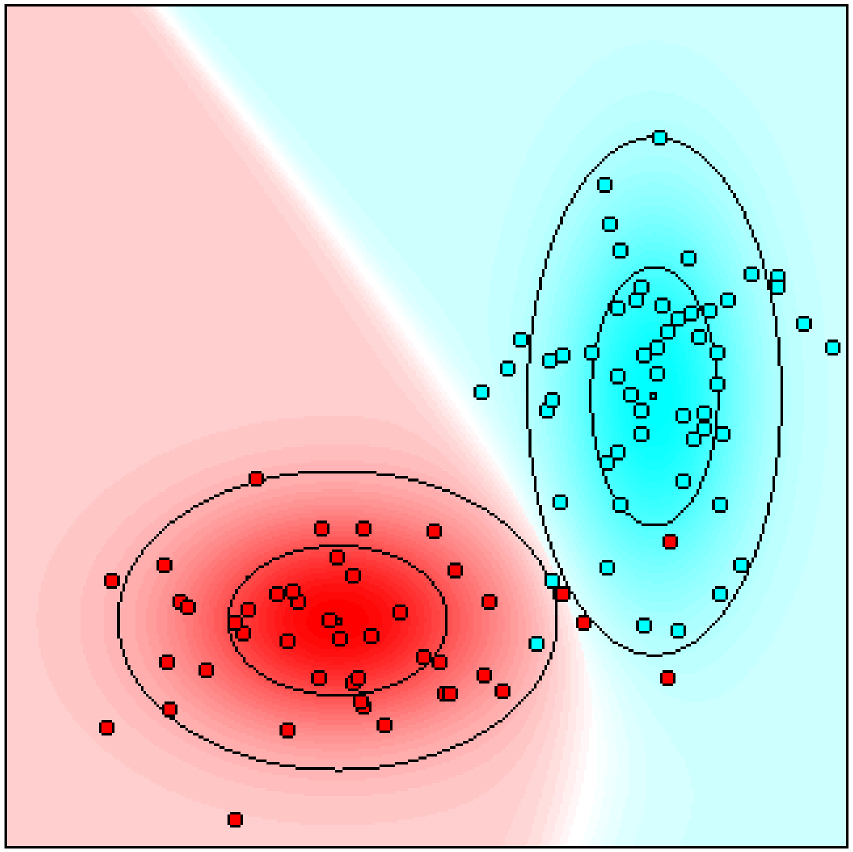
$$
\begin{aligned}
&P(\text{Drug B} \mid \text{male, 61, normal}) \\
&\quad = \; c_1 \cdot P(\text{Drug B}) \cdot P(\text{male} \mid \text{Drug B}) \cdot P(61 \mid \text{Drug B}) \cdot P(\text{normal} \mid \text{Drug B}) \\
&\quad \approx \; c_1 \cdot 0.5 \cdot 0.5 \cdot 0.017120 \cdot 0.5 \quad = \quad c_1 \cdot 2.140 \cdot 10^{-3} \quad = \quad 0.781
\end{aligned}
$$

$$
\begin{aligned}
&P(\text{Drug A} \mid \text{female, 30, normal}) \\
&\quad = \; c_2 \cdot P(\text{Drug A}) \cdot P(\text{female} \mid \text{Drug A}) \cdot P(30 \mid \text{Drug A}) \cdot P(\text{normal} \mid \text{Drug A}) \\
&\quad \approx \; c_2 \cdot 0.5 \cdot 0.5 \cdot 0.027703 \cdot 0.5 \quad = \quad c_2 \cdot 3.471 \cdot 10^{-3} \quad = \quad 0.671
\end{aligned}
$$

$$
\begin{aligned}
&P(\text{Drug B} \mid \text{female, 30, normal}) \\
&\quad = \; c_2 \cdot P(\text{Drug B}) \cdot P(\text{female} \mid \text{Drug B}) \cdot P(30 \mid \text{Drug B}) \cdot P(\text{normal} \mid \text{Drug B}) \\
&\quad \approx \; c_2 \cdot 0.5 \cdot 0.5 \cdot 0.013567 \cdot 0.5 \quad = \quad c_2 \cdot 1.696 \cdot 10^{-3} \quad = \quad 0.329
\end{aligned}
$$

Naive Bayes Classifiers: Simple Example 2

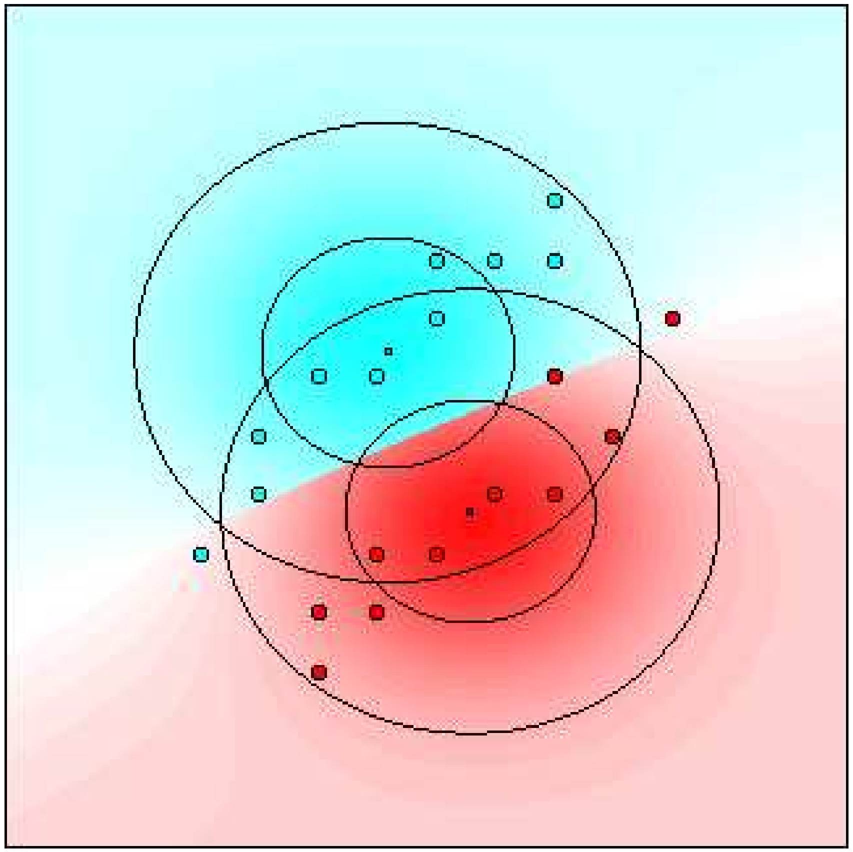
- 100 data points, 2 classes
- Small squares: mean values
- Inner ellipses:
  one standard deviation
- Outer ellipses:
  two standard deviations
- Classes overlap:
  classification is not perfect



Naive Bayes Classifier

Naive Bayes Classifiers: Simple Example 3

- 20 data points, 2 classes
- Small squares: mean values
- Inner ellipses:
  one standard deviation
- Outer ellipses:
  two standard deviations
- Attributes are not conditionally
  independent given the class



Naive Bayes Classifier

Naive Bayes Classifiers: Iris Data

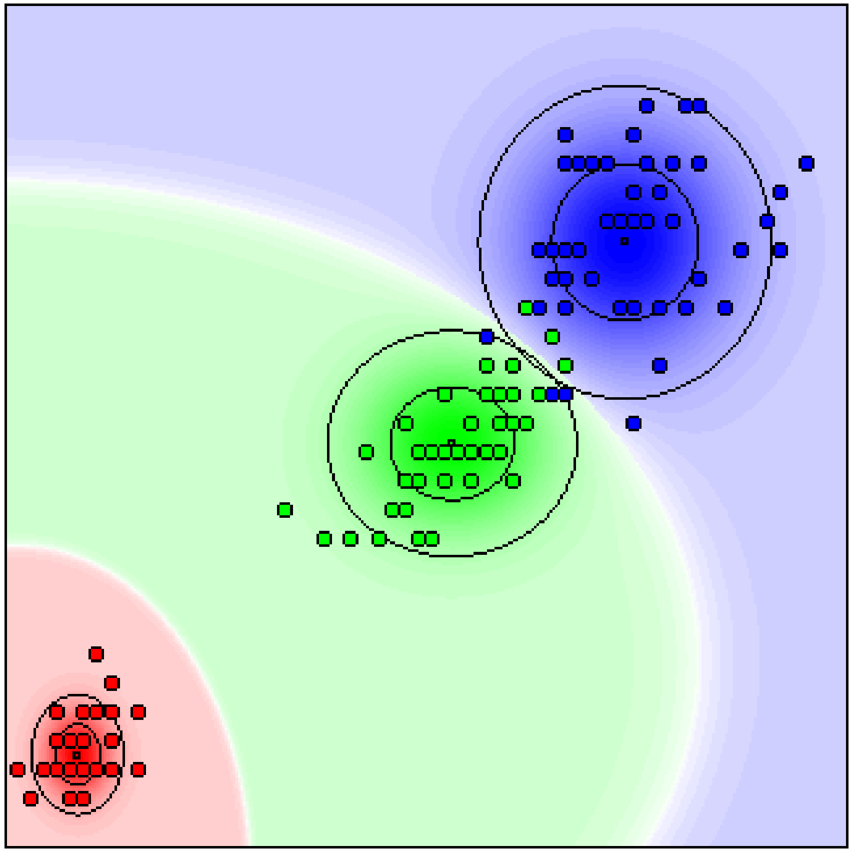
- 150 data points, 3 classes

Iris setosa	(red)
Iris versicolor	(green)
Iris virginica	(blue)

- Shown: 2 out of 4 attributes

sepal length	
sepal width	
petal length	(horizontal)
petal width	(vertical)

- 6 misclassifications
  on the training data
  (with all 4 attributes)



Naive Bayes Classifier

Full Bayes Classifiers

- Restricted to metric/numeric attributes (only the class is nominal/symbolic).
- **Simplifying Assumption:**
  Each class can be described by a multivariate normal distribution.

$$
\begin{aligned}
& f(A_1 = a_1, \ldots, A_m = a_m \mid C = c_i) \\
&= \frac{1}{\sqrt{(2\pi)^m |\mathbf{\Sigma}_i|}} \exp\left(-\frac{1}{2} (\vec{a} - \vec{\mu}_i)^\top \mathbf{\Sigma}_i^{-1} (\vec{a} - \vec{\mu}_i) \right)
\end{aligned}
$$

  $\vec{\mu}_i$: mean value vector for class c_i

  $\mathbf{\Sigma}_i$: covariance matrix for class c_i

- Intuitively: Each class has a bell-shaped probability density.
- Naive Bayes classifiers: Covariance matrices are diagonal matrices.
  (Details about this relation are given below.)

Full Bayes Classifiers

Estimation of Probabilities:

- Estimate of mean value vector

$$\hat{\vec{\mu}}_i = \frac{1}{\#(C = c_i)} \sum_{j=1}^{\#(C=c_i)} \vec{a}(j)$$

- Estimate of covariance matrix

$$\widehat{\mathbf{\Sigma}}_i = \frac{1}{\xi} \sum_{j=1}^{\#(C=c_i)} \left(\vec{a}(j) - \hat{\vec{\mu}}_i\right) \left(\vec{a}(j) - \hat{\vec{\mu}}_i\right)^\top$$

$\xi = \#(C = c_i)$: Maximum likelihood estimation
$\xi = \#(C = c_i) - 1$: Unbiased estimation

$\vec{x}^\top$ denotes the transpose of the vector $\vec{x}$.

$\vec{x}\vec{x}^\top$ is the so-called **outer product** or **matrix product** of $\vec{x}$ with itself.

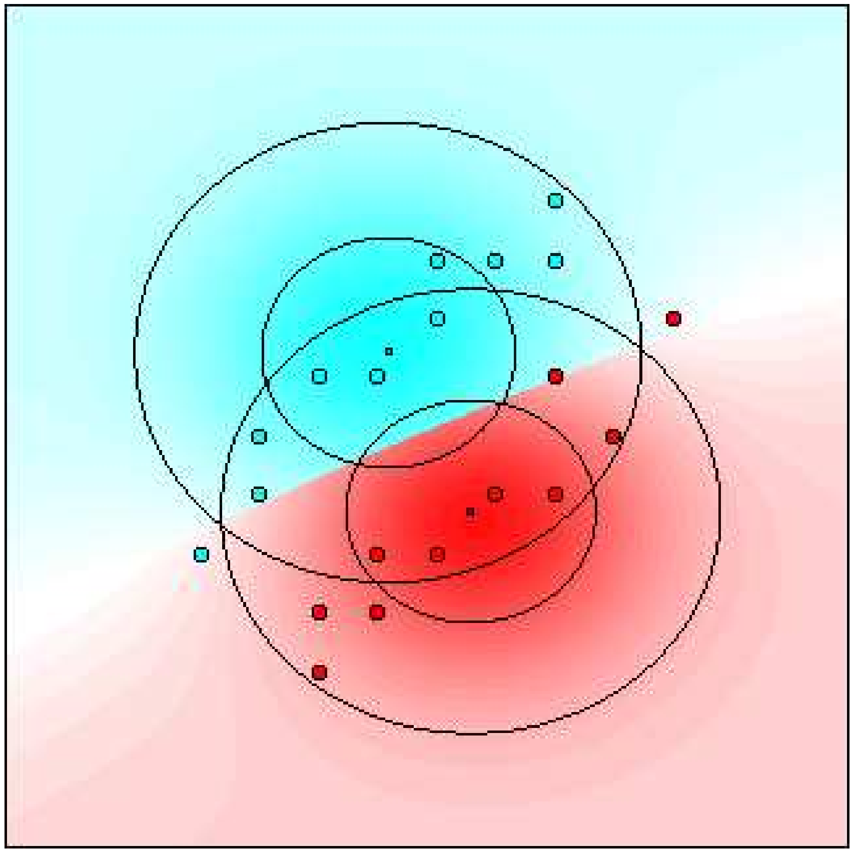
Comparison of Naive and Full Bayes Classifiers

Naive Bayes classifiers for metric/numeric data are equivalent to full Bayes classifiers with diagonal covariance matrices:

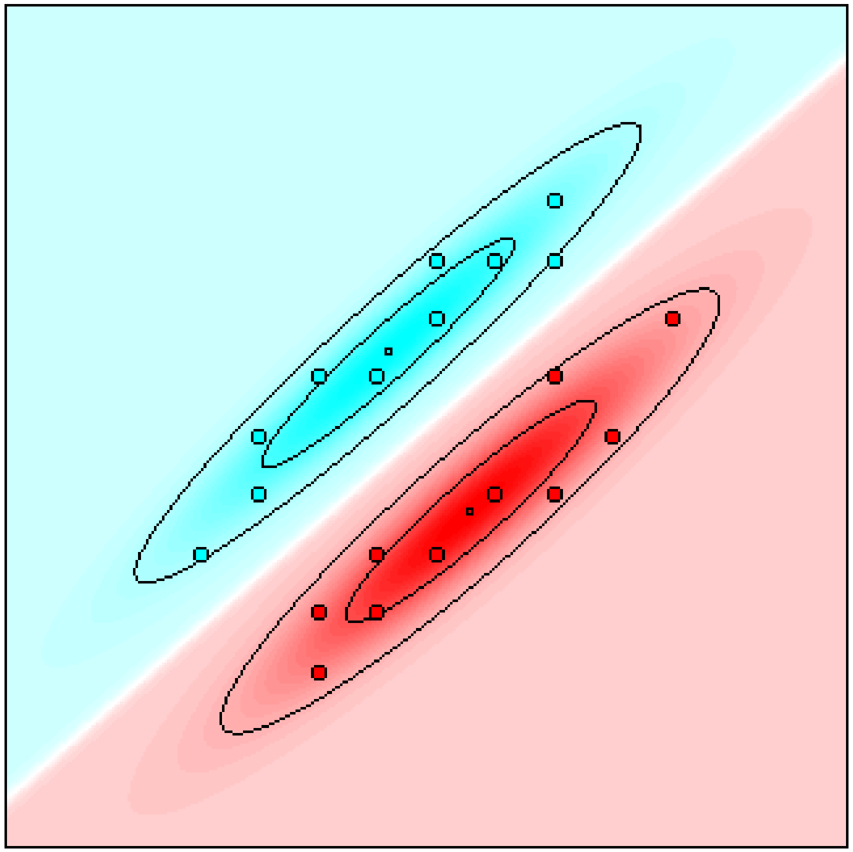
$$\begin{aligned}
&f(A_1 = a_1, \ldots, A_m = a_m \mid C = c_i) \\
&= \frac{1}{\sqrt{(2\pi)^m |\mathbf{\Sigma}_i|}} \cdot \exp\left(-\frac{1}{2}(\vec{a} - \vec{\mu}_i)^\top \mathbf{\Sigma}_i^{-1} (\vec{a} - \vec{\mu}_i)\right) \\
&= \frac{1}{\sqrt{(2\pi)^m \prod_{k=1}^m \sigma_{i,k}^2}} \cdot \exp\left(-\frac{1}{2}(\vec{a} - \vec{\mu}_i)^\top \operatorname{diag}\left(\sigma_{i,1}^{-2}, \ldots, \sigma_{i,m}^{-2}\right) (\vec{a} - \vec{\mu}_i)\right) \\
&= \frac{1}{\prod_{k=1}^m \sqrt{2\pi\sigma_{i,k}^2}} \cdot \exp\left(-\frac{1}{2}\sum_{k=1}^m \frac{(a_k - \mu_{i,k})^2}{\sigma_{i,k}^2}\right) \\
&= \prod_{k=1}^m \frac{1}{\sqrt{2\pi\sigma_{i,k}^2}} \cdot \exp\left(-\frac{(a_k - \mu_{i,k})^2}{2\sigma_{i,k}^2}\right) \;\widehat{=}\; \prod_{k=1}^m f(A_k = a_k \mid C = c_i),
\end{aligned}$$

where $f(A_k = a_k \mid C = c_i)$ are the density functions used by a naive Bayes classifier.

Comparison of Naive and Full Bayes Classifiers



Naive Bayes Classifier



Full Bayes Classifier

Full Bayes Classifiers: Iris Data

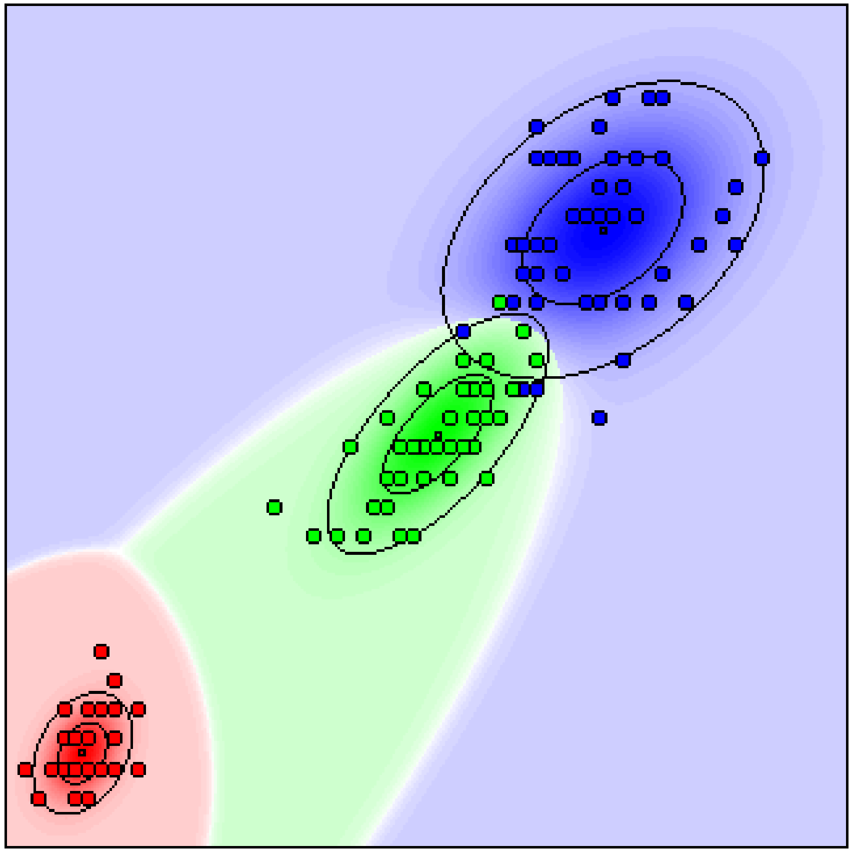
- 150 data points, 3 classes

Iris setosa	(red)
Iris versicolor	(green)
Iris virginica	(blue)

- Shown: 2 out of 4 attributes

sepal length	
sepal width	
petal length	(horizontal)
petal width	(vertical)

- 2 misclassifications on the training data (with all 4 attributes)



Full Bayes Classifier

Summary Bayes Classifiers

- **Probabilistic Classification**: Assign the most probable class.
- **Bayes' Rule**: "Invert" the conditional class probabilities.
- **Naive Bayes Classifiers**
    - Simplifying Assumption:
      Attributes are conditionally independent given the class.
    - Can handle nominal/symbolic as well as metric/numeric attributes.
- **Full Bayes Classifiers**
    - Simplifying Assumption:
      Each class can be described by a multivariate normal distribution.
    - Can handle only metric/numeric attributes.

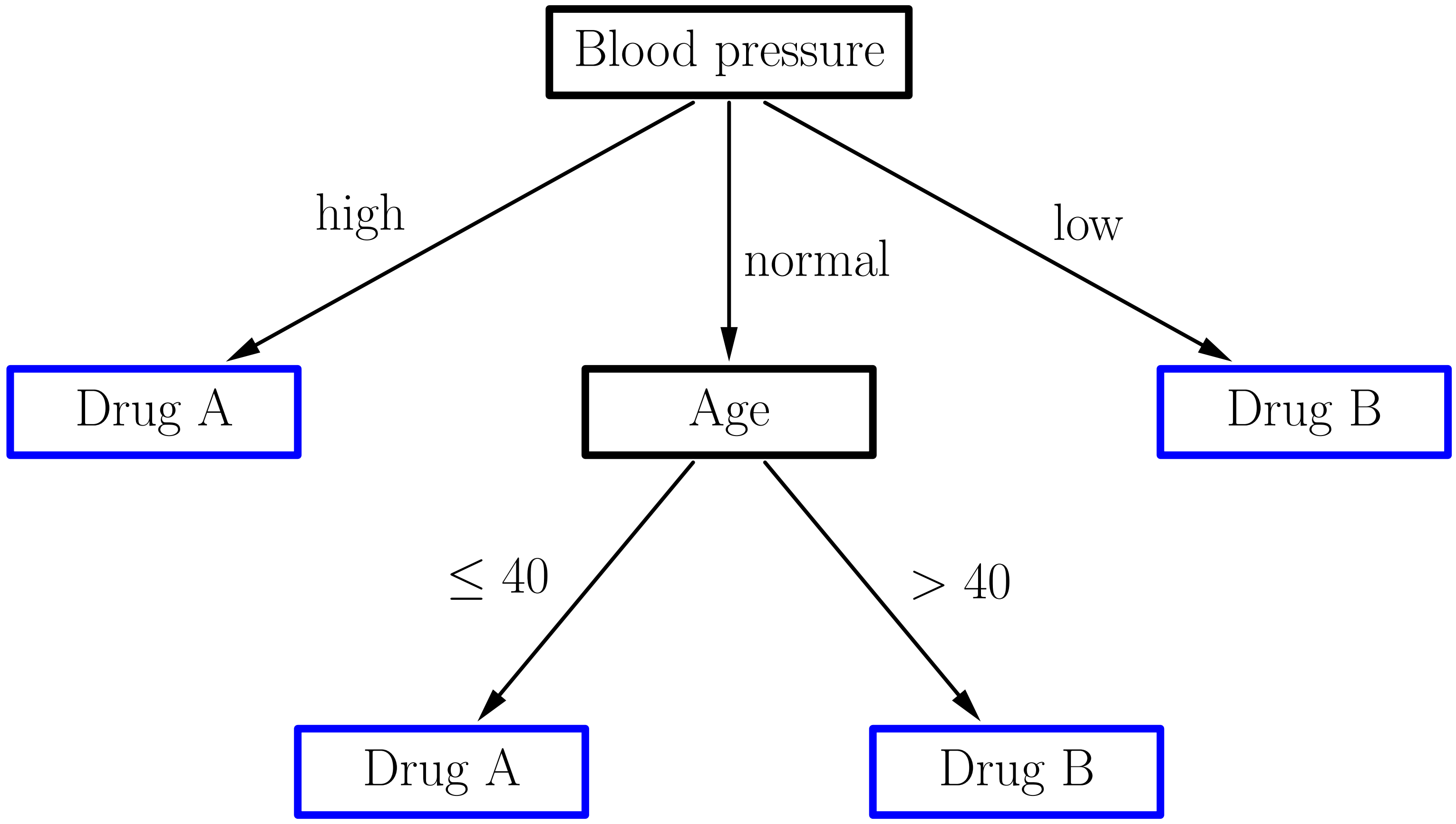
Decision and Regression Trees

Decision and Regression Trees

- **Classification with a Decision Tree**
- **Top-down Induction of Decision Trees**
  - A simple example
  - The general algorithm
  - Attribute selection measures
  - Treatment of numeric attributes and missing values
- **Pruning Decision Trees**
  - General approaches
  - A simple example
- **Regression Trees**
- **Summary**

A Very Simple Decision Tree

Assignment of a drug to a patient:



Classification with a Decision Tree

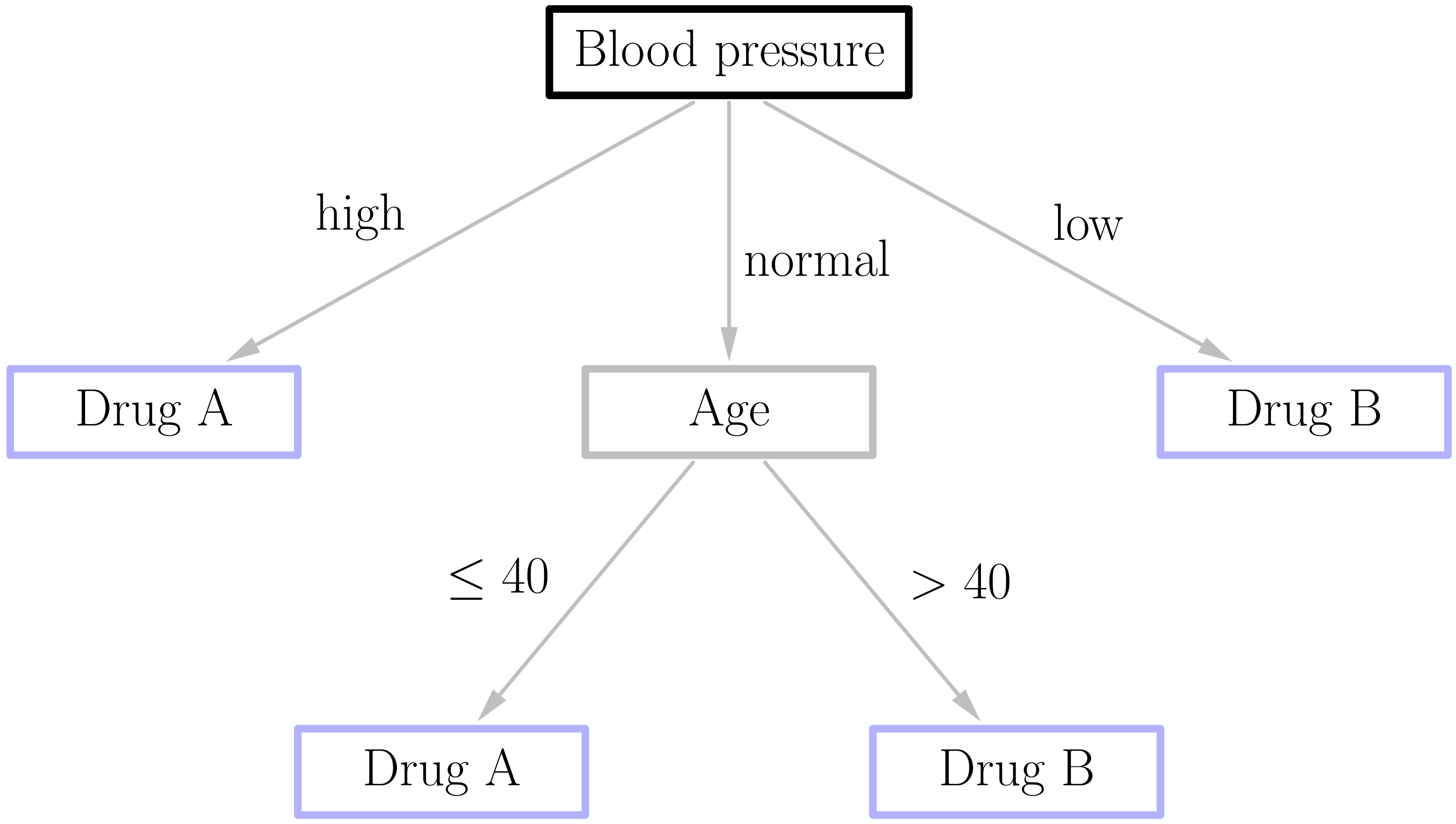
Recursive Descent:

- Start at the root node.
- If the current node is an **leaf node**:
  - Return the class assigned to the node.
- If the current node is an **inner node**:
  - Test the attribute associated with the node.
  - Follow the branch labeled with the outcome of the test.
  - Apply the algorithm recursively.

Intuitively: Follow the path corresponding to the case to be classified.

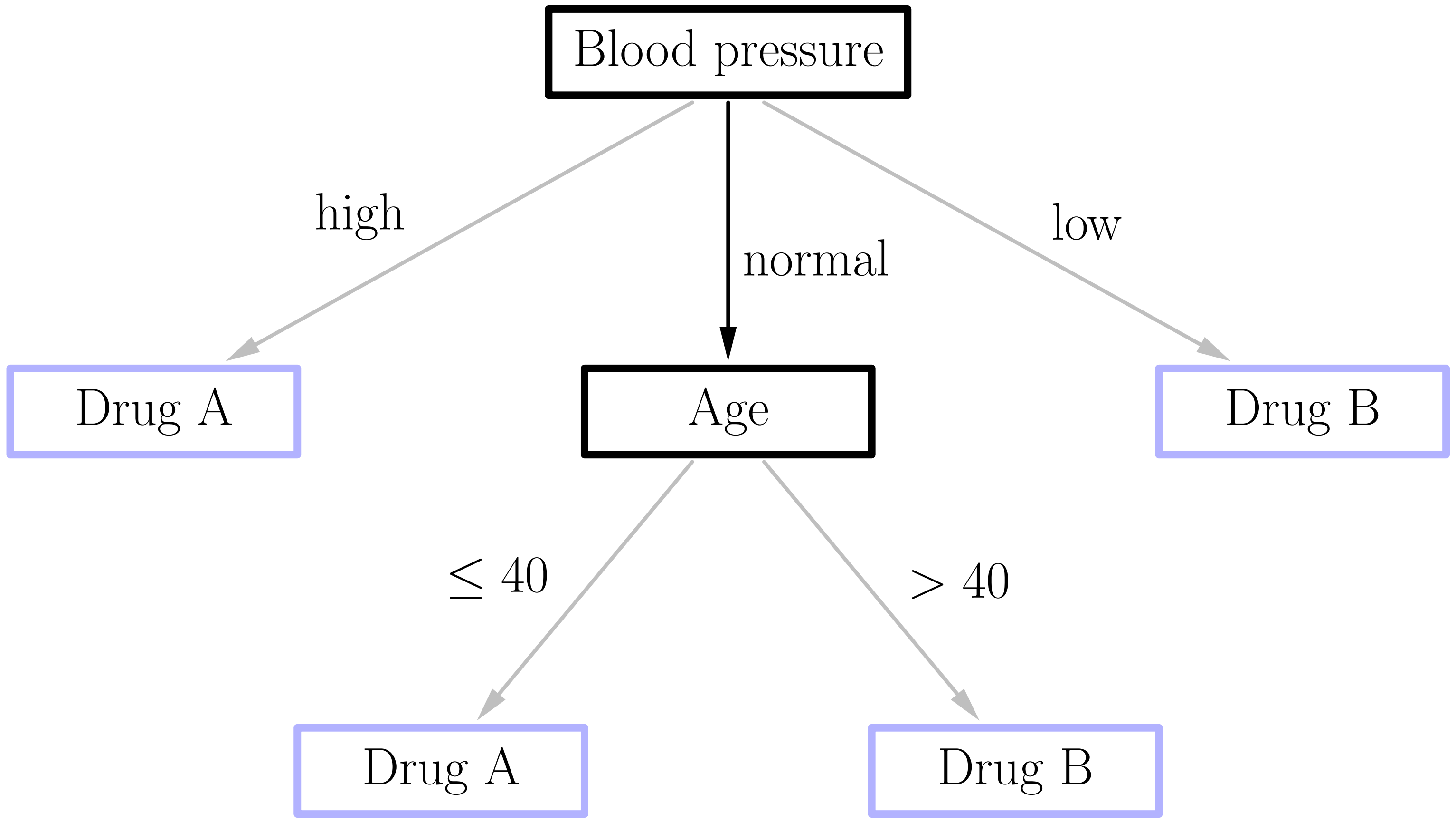
Classification in the Example

Assignment of a drug to a patient:



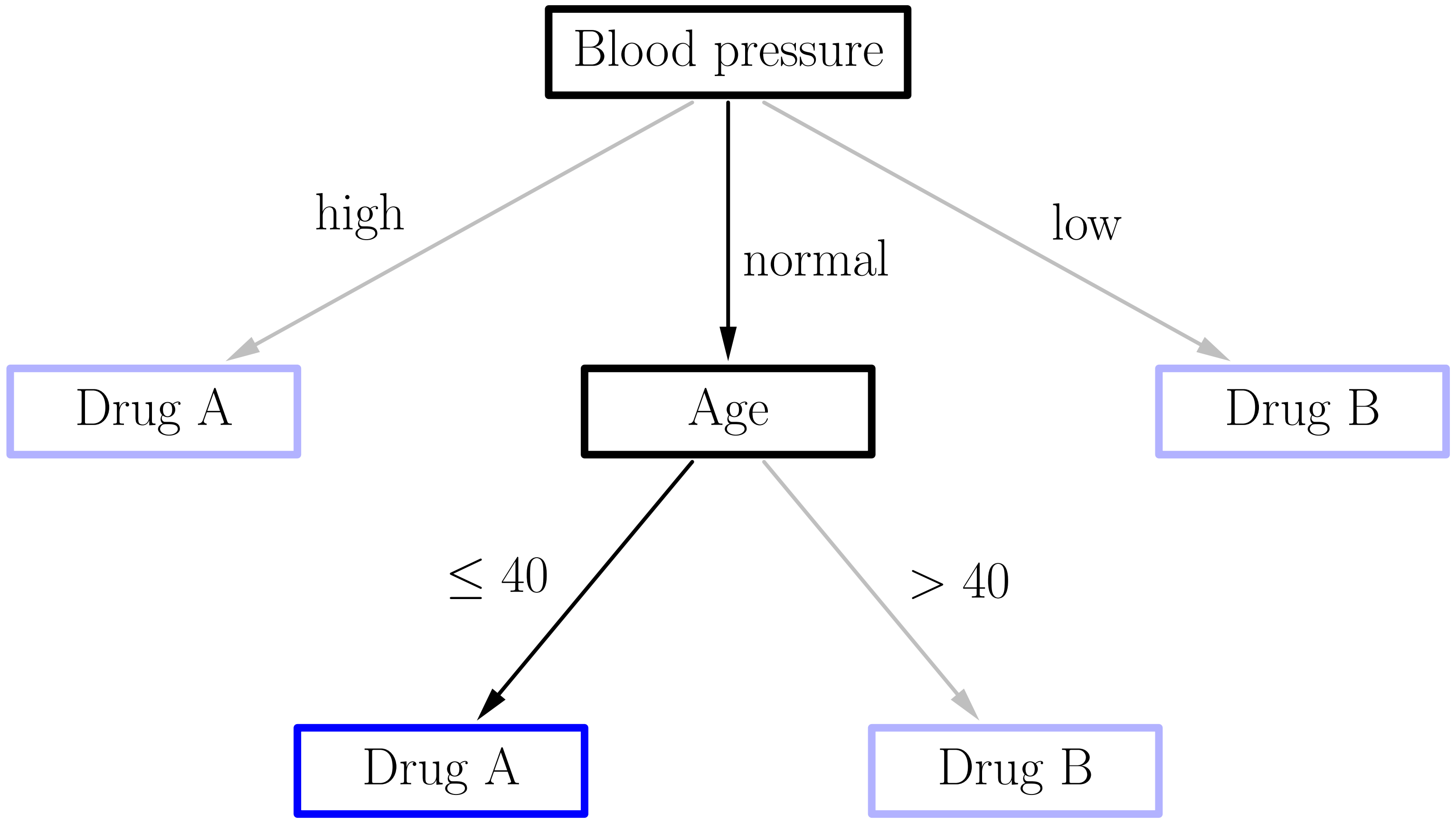
Classification in the Example

Assignment of a drug to a patient:



Classification in the Example

Assignment of a drug to a patient:



Induction of Decision Trees

- **Top-down approach**
  - Build the decision tree from top to bottom (from the root to the leaves).
- **Greedy Selection of a Test Attribute**
  - Compute an evaluation measure for all attributes.
  - Select the attribute with the best evaluation.
- **Divide and Conquer / Recursive Descent**
  - Divide the example cases according to the values of the test attribute.
  - Apply the procedure recursively to the subsets.
  - Terminate the recursion if
    - all cases belong to the same class
    - no more test attributes are available

Induction of a Decision Tree: Example

Patient database

- 12 example cases
- 3 descriptive attributes
- 1 class attribute

No	Sex	Age	Blood pr.	Drug
1	male	20	normal	A
2	female	73	normal	B
3	female	37	high	A
4	male	33	low	B
5	female	48	high	A
6	male	29	normal	A
7	female	52	normal	B
8	male	42	low	B
9	male	61	normal	B
10	female	30	normal	A
11	female	26	low	B
12	male	54	high	A

Assignment of drug

(without patient attributes)

always drug A or always drug B:

50% correct (in 6 of 12 cases)

Induction of a Decision Tree: Example

Sex of the patient

- Division w.r.t. male/female.

Assignment of drug

male:	50% correct	(in 3 of 6 cases)
female:	50% correct	(in 3 of 6 cases)
total:	**50% correct**	(in 6 of 12 cases)

No	Sex	Drug
1	male	A
6	male	A
12	male	A
4	male	B
8	male	B
9	male	B
3	female	A
5	female	A
10	female	A
2	female	B
7	female	B
11	female	B

Induction of a Decision Tree: Example

Age of the patient

- Sort according to age.
- Find best age split.
  here: ca. 40 years

Assignment of drug

≤ 40:	A	67% correct	(in 4 of 6 cases)
> 40:	B	67% correct	(in 4 of 6 cases)
total:		**67% correct**	(in 8 of 12 cases)

No	Age	Drug
1	20	A
11	26	B
6	29	A
10	30	A
4	33	B
3	37	A
8	42	B
5	48	A
7	52	B
12	54	A
9	61	B
2	73	B

Induction of a Decision Tree: Example

Blood pressure of the patient

- Division w.r.t. high/normal/low.

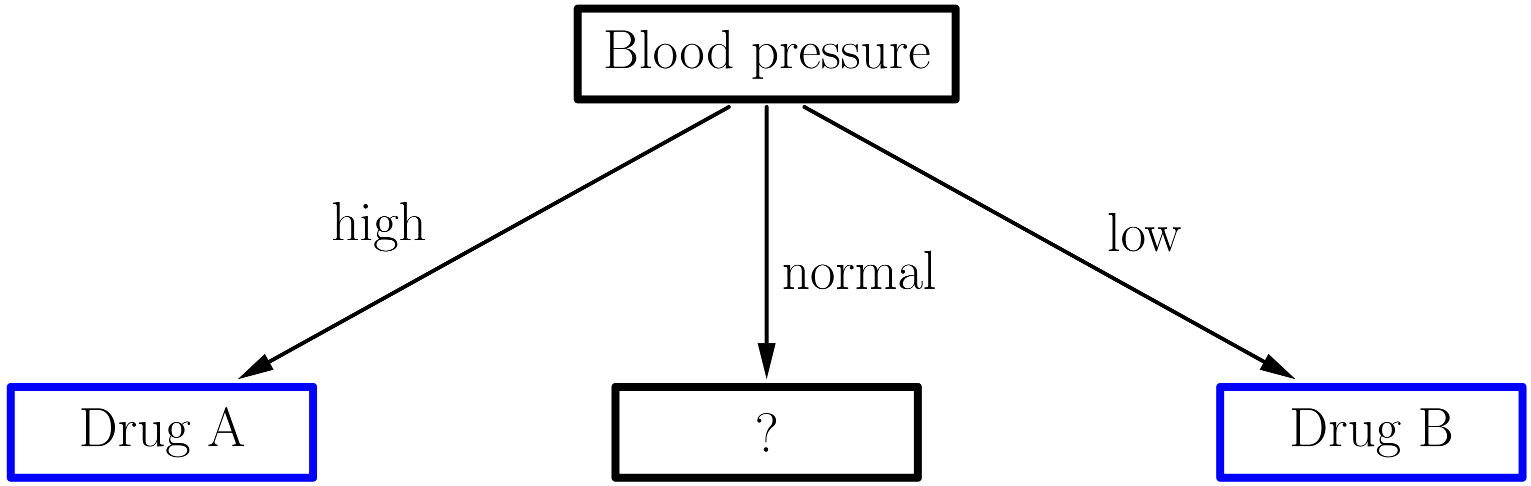
Assignment of drug

high:	A	100% correct	(in 3 of 3 cases)
normal:		50% correct	(in 3 of 6 cases)
low:	B	100% correct	(in 3 of 3 cases)
total:		**75% correct**	(in 9 of 12 cases)

No	Blood pr.	Drug
3	high	A
5	high	A
12	high	A
1	normal	A
6	normal	A
10	normal	A
2	normal	B
7	normal	B
9	normal	B
4	low	B
8	low	B
11	low	B

Induction of a Decision Tree: Example

Current Decision Tree:



Induction of a Decision Tree: Example

Blood pressure and sex

- Only patients with normal blood pressure.
- Division w.r.t. male/female.

Assignment of drug

male:	A	67% correct	(2 of 3)
female:	B	67% correct	(2 of 3)
total:		**67% correct**	(4 of 6)

No	Blood pr.	Sex	Drug
3	high		A
5	high		A
12	high		A
1	normal	male	A
6	normal	male	A
9	normal	male	B
2	normal	female	B
7	normal	female	B
10	normal	female	A
4	low		B
8	low		B
11	low		B

Induction of a Decision Tree: Example

Blood pressure and age

- Only patients with normal blood pressure.
- Sort according to age.
- Find best age split. here: ca. 40 years

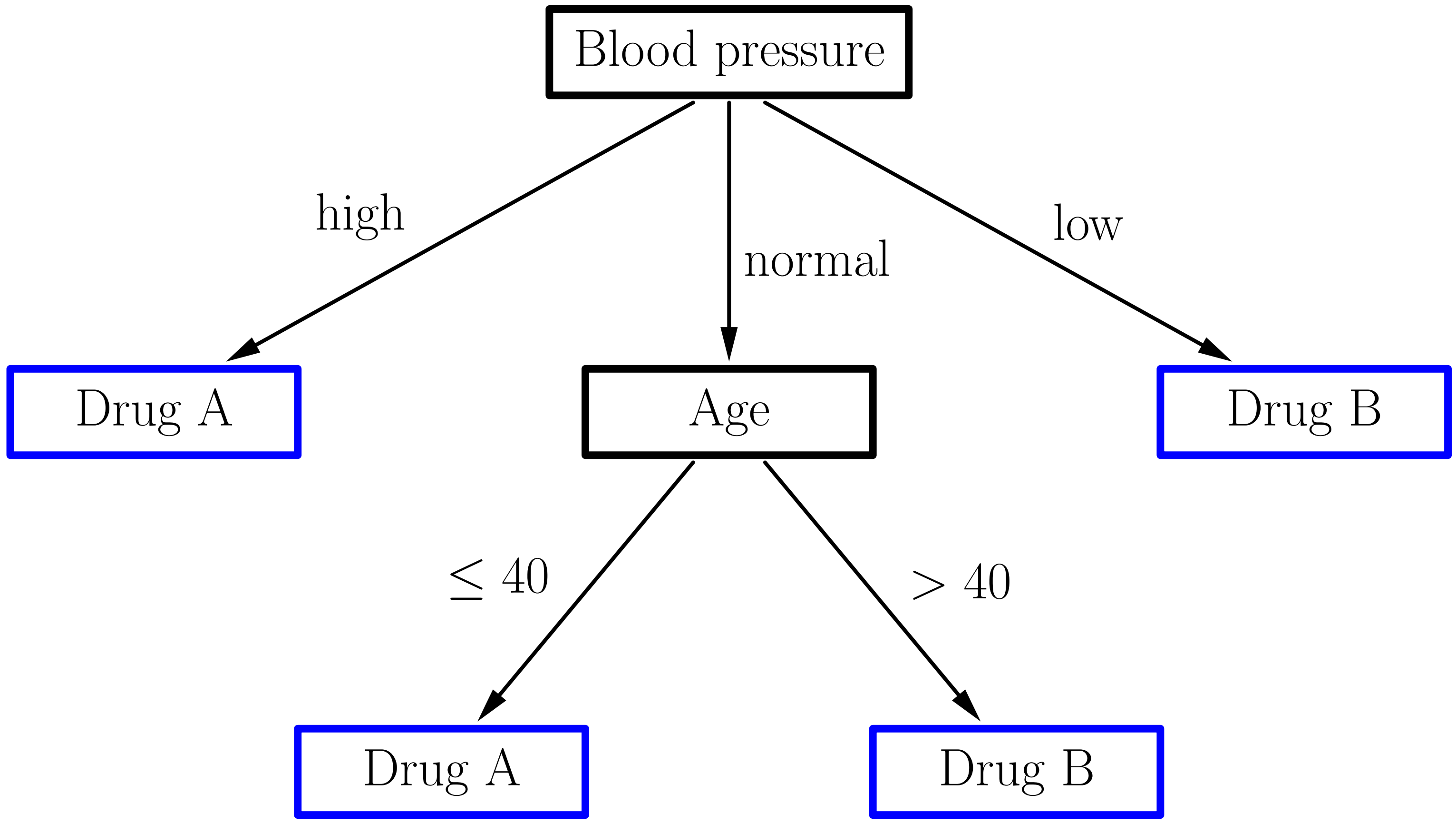
Assignment of drug

≤ 40:	A	100% correct	(3 of 3)
> 40:	B	100% correct	(3 of 3)
total:		**100% correct**	(6 of 6)

No	Blood pr.	Age	Drug
3	high		A
5	high		A
12	high		A
1	normal	20	A
6	normal	29	A
10	normal	30	A
7	normal	52	B
9	normal	61	B
2	normal	73	B
11	low		B
4	low		B
8	low		B

Result of Decision Tree Induction

Assignment of a drug to a patient:



Decision Tree Induction: Notation

S	a set of case or object descriptions
C	the class attribute
$A^{(1)}, \ldots, A^{(m)}$	other attributes (index dropped in the following)
$\mathrm{dom}(C)$	$= \{c_1, \ldots, c_{n_C}\}$, $\quad n_C$: number of classes
$\mathrm{dom}(A)$	$= \{a_1, \ldots, a_{n_A}\}$, $\quad n_A$: number of attribute values
$N_{..}$	total number of case or object descriptions i.e. $N_{..} = \|S\|$
$N_{i.}$	absolute frequency of the class c_i
$N_{.j}$	absolute frequency of the attribute value a_j
N_{ij}	absolute frequency of the combination of the class c_i and the attribute value a_j. It is $N_{i.} = \sum_{j=1}^{n_A} N_{ij}$ and $N_{.j} = \sum_{i=1}^{n_C} N_{ij}$.
$p_{i.}$	relative frequency of the class c_i, $p_{i.} = \frac{N_{i.}}{N_{..}}$
$p_{.j}$	relative frequency of the attribute value a_j, $p_{.j} = \frac{N_{.j}}{N_{..}}$
p_{ij}	relative frequency of the combination of class c_i and attribute value a_j, $p_{ij} = \frac{N_{ij}}{N_{..}}$
$p_{i\|j}$	relative frequency of the class c_i in cases having attribute value a_j, $p_{i\|j} = \frac{N_{ij}}{N_{.j}} = \frac{p_{ij}}{p_{.j}}$

Decision Tree Induction: General Algorithm

```
function grow_tree ($S$ : set of cases) : node;
begin
    $best\_v$ := WORTHLESS;
    for all untested attributes $A$ do
            compute frequencies $N_{ij}$, $N_{i.}$, $N_{.j}$ for $1 \leq i \leq n_C$ and $1 \leq j \leq n_A$;
            compute value $v$ of an evaluation measure using $N_{ij}$, $N_{i.}$, $N_{.j}$;
            if $v > best\_v$ then $best\_v := v$; $best\_A := A$; end;
    end
    if $best\_v$ = WORTHLESS
    then create leaf node $x$;
            assign majority class of $S$ to $x$;
    else    create test node $x$;
            assign test on attribute $best\_A$ to $x$;
            for all $a \in \mathrm{dom}(best\_A)$ do $x$.child[$a$] := grow_tree($S|_{best\_A=a}$); end;
    end;
    return $x$;
end;
```

Evaluation Measures

- Evaluation measure used in the above example:
  rate of correctly classified example cases.
  - Advantage: simple to compute, easy to understand.
  - Disadvantage: works well only for two classes.
- If there are more than two classes, the rate of misclassified example cases **neglects a lot of the available information**.
  - Only the majority class—that is, the class occurring most often in (a subset of) the example cases—is really considered.
  - The distribution of the other classes has no influence. However, a good choice here can be important for deeper levels of the decision tree.
- **Therefore:** Study also other evaluation measures. Here:
  - **Information gain** and its various normalizations.
  - χ^2 **measure** (well-known in statistics).

An Information-theoretic Evaluation Measure

Information Gain (Kullback and Leibler 1951, Quinlan 1986)

Based on Shannon Entropy $H = -\sum_{i=1}^{n} p_i \log_2 p_i$ (Shannon 1948)

$$I_{\text{gain}}(C, A) = \overbrace{-\sum_{i=1}^{n_C} p_{i.} \log_2 p_{i.}}^{H(C)} - \overbrace{\sum_{j=1}^{n_A} p_{.j} \left(-\sum_{i=1}^{n_C} p_{i|j} \log_2 p_{i|j} \right)}^{H(C|A)}$$

$H(C)$	Entropy of the class distribution (C: class attribute)
$H(C\|A)$	*Expected entropy* of the class distribution if the value of the attribute A becomes known
$H(C) - H(C\|A)$	Expected entropy reduction or *information gain*

Inducing the Decision Tree with Information Gain

- Information gain for drug and sex:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Sex}) = \frac{1}{2}\underbrace{\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right)}_{H(\text{Drug}|\text{Sex=male})} + \frac{1}{2}\underbrace{\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right)}_{H(\text{Drug}|\text{Sex=female})} = 1$$

$$I_{\text{gain}}(\text{Drug}, \text{Sex}) = 1 - 1 = 0$$

- No gain at all since the initial the uniform distribution of drug is splitted into two (still) uniform distributions.

Inducing the Decision Tree with Information Gain

- Information gain for drug and age:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Age}) = \frac{1}{2}\underbrace{\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right)}_{H(\text{Drug}|\text{Age}\leq 40)} + \frac{1}{2}\underbrace{\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right)}_{H(\text{Drug}|\text{Age}>40)} \approx 0.9183$$

$$I_{\text{gain}}(\text{Drug}, \text{Age}) = 1 - 0.9183 = 0.0817$$

- Splitting w. r. t. age can reduce the overall entropy.

Inducing the Decision Tree with Information Gain

- Information gain for drug and blood pressure:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Blood\_pr}) = \frac{1}{4}\cdot 0 + \frac{1}{2}\Bigg(\underbrace{-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}}_{H(\text{Drug}|\text{Blood\_pr=normal})} \Bigg) + \frac{1}{4}\cdot 0 = 0.5$$

$$I_{\text{gain}}(\text{Drug}, \text{Blood\_pr}) = 1 - 0.5 = 0.5$$

- Largest information gain, so we first split w. r. t. blood pressure (as in the example with misclassification rate).

Inducing the Decision Tree with Information Gain

- Next level: Subtree blood pressure is normal.
- Information gain for drug and sex:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Sex}) = \frac{1}{2}\underbrace{\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right)}_{H(\text{Drug}|\text{Sex=male})} + \frac{1}{2}\underbrace{\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right)}_{H(\text{Drug}|\text{Sex=female})} = 0.9183$$

$$I_{\text{gain}}(\text{Drug}, \text{Sex}) = 0.0817$$

- Entropy can be decreased.

Inducing the Decision Tree with Information Gain

- Next level: Subtree blood pressure is normal.
- Information gain for drug and age:

$$\begin{aligned}
H(\text{Drug}) &= -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1 \\
H(\text{Drug} \mid \text{Age}) &= \frac{1}{2}\cdot 0 + \frac{1}{2}\cdot 0 = 0 \\
I_{\text{gain}}(\text{Drug}, \text{Age}) &= 1
\end{aligned}$$

- Maximal information gain, that is we result in a perfect classification (again, as in the case of using misclassification rate).

Interpretation of Shannon Entropy

- Let $S = \{s_1, \ldots, s_n\}$ be a finite set of alternatives having positive probabilities $P(s_i)$, $i = 1, \ldots, n$, satisfying $\sum_{i=1}^{n} P(s_i) = 1$.
- **Shannon Entropy:**

$$H(S) = -\sum_{i=1}^{n} P(s_i) \log_2 P(s_i)$$

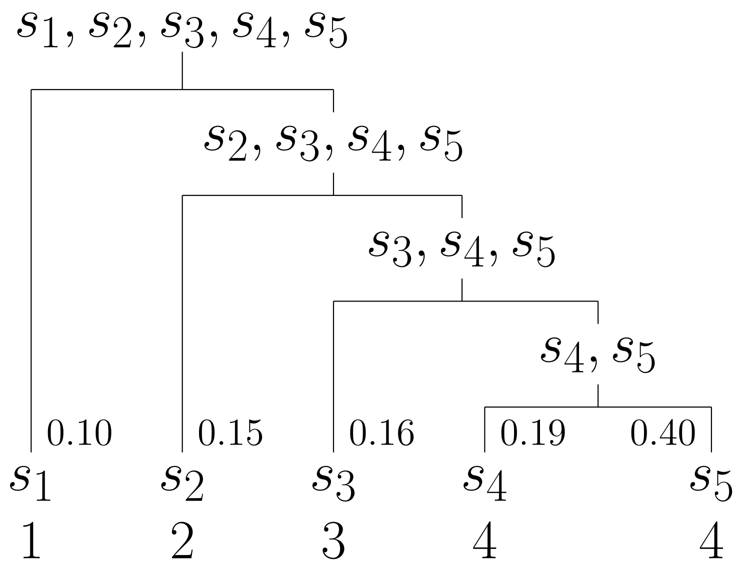
- Intuitively: **Expected number of yes/no questions that have to be asked in order to determine the obtaining alternative.**
    - Suppose there is an oracle, which knows the obtaining alternative, but responds only if the question can be answered with "yes" or "no".
    - A better question scheme than asking for one alternative after the other can easily be found: Divide the set into two subsets of about equal size.
    - Ask for containment in an arbitrarily chosen subset.
    - Apply this scheme recursively $\rightarrow$ number of questions bounded by $\lceil \log_2 n \rceil$.

Question/Coding Schemes

$$P(s_1) = 0.10, \quad P(s_2) = 0.15, \quad P(s_3) = 0.16, \quad P(s_4) = 0.19, \quad P(s_5) = 0.40$$

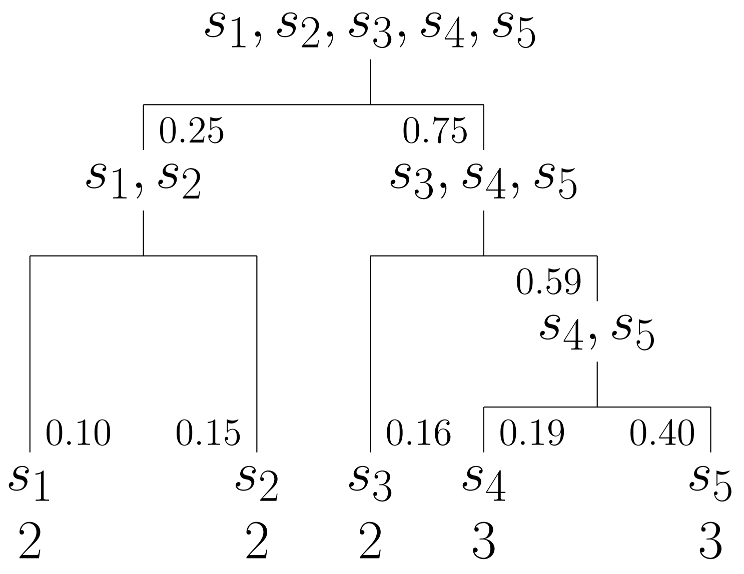
Shannon entropy: $\quad -\sum_i P(s_i) \log_2 P(s_i) = 2.15$ bit/symbol

Linear Traversal



Code length: 3.24 bit/symbol
Code efficiency: 0.664

Equal Size Subsets



Code length: 2.59 bit/symbol
Code efficiency: 0.830

Question/Coding Schemes

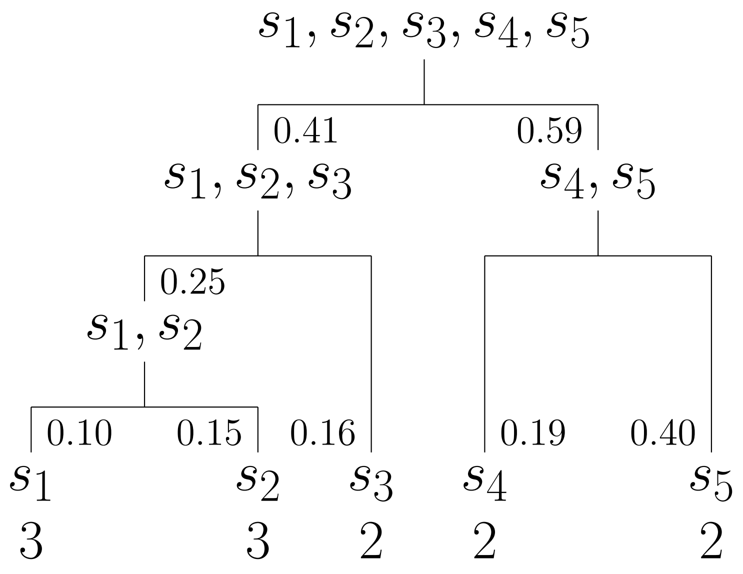
- Splitting into subsets of about equal size can lead to a bad arrangement of the alternatives into subsets $\rightarrow$ high expected number of questions.
- Good question schemes take the probability of the alternatives into account.
- **Shannon-Fano Coding** (1948)
  - Build the question/coding scheme top-down.
  - Sort the alternatives w.r.t. their probabilities.
  - Split the set so that the subsets have about equal *probability* (splits must respect the probability order of the alternatives).
- **Huffman Coding** (1952)
  - Build the question/coding scheme bottom-up.
  - Start with one element sets.
  - Always combine those two sets that have the smallest probabilities.

Question/Coding Schemes

$$P(s_1) = 0.10, \quad P(s_2) = 0.15, \quad P(s_3) = 0.16, \quad P(s_4) = 0.19, \quad P(s_5) = 0.40$$

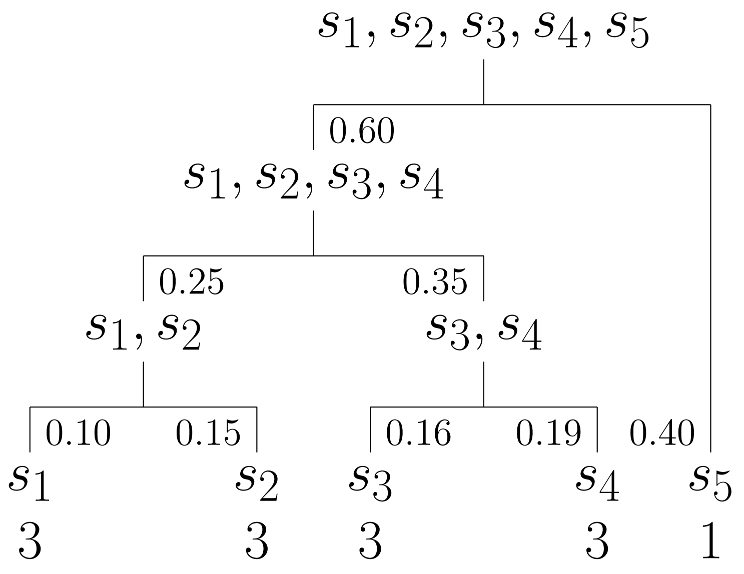
Shannon entropy: $\quad -\sum_i P(s_i) \log_2 P(s_i) = 2.15$ bit/symbol

Shannon–Fano Coding (1948)



Code length: 2.25 bit/symbol
Code efficiency: 0.955

Huffman Coding (1952)



Code length: 2.20 bit/symbol
Code efficiency: 0.977

Question/Coding Schemes

- It can be shown that Huffman coding is optimal if we have to determine the obtaining alternative in a single instance.
  (No question/coding scheme has a smaller expected number of questions.)
- Only if the obtaining alternative has to be determined in a sequence of (independent) situations, this scheme can be improved upon.
- Idea: Process the sequence not instance by instance, but combine two, three or more consecutive instances and ask directly for the obtaining combination of alternatives.
- Although this enlarges the question/coding scheme, the expected number of questions per identification is reduced (because each interrogation identifies the obtaining alternative for several situations).
- However, the expected number of questions per identification cannot be made arbitrarily small. Shannon showed that there is a lower bound, namely the Shannon entropy.

Interpretation of Shannon Entropy

$$P(s_1) = \tfrac{1}{2}, \quad P(s_2) = \tfrac{1}{4}, \quad P(s_3) = \tfrac{1}{8}, \quad P(s_4) = \tfrac{1}{16}, \quad P(s_5) = \tfrac{1}{16}$$

Shannon entropy: $\quad -\sum_i P(s_i) \log_2 P(s_i) = 1.875$ bit/symbol

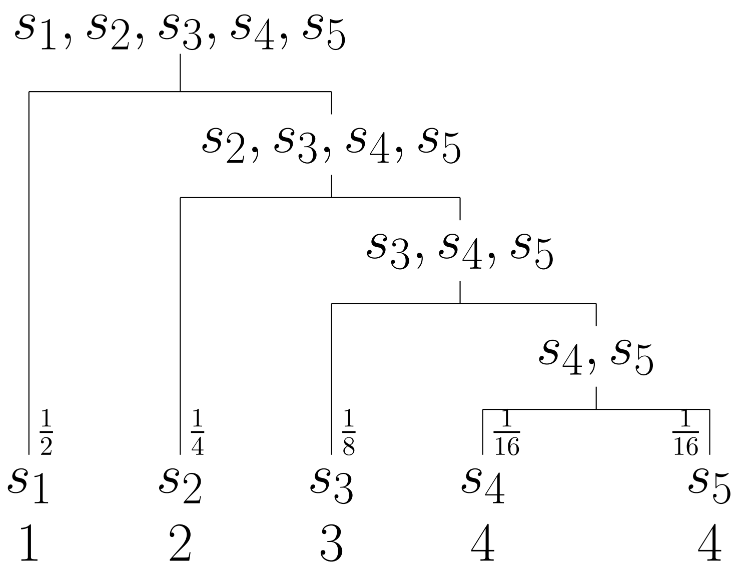
If the probability distribution allows for a perfect Huffman code (code efficiency 1), the Shannon entropy can easily be interpreted as follows:

$$-\sum_i P(s_i) \log_2 P(s_i)$$

$$= \sum_i \underbrace{P(s_i)}_{\substack{\text{occurrence}\\ \text{probability}}} \cdot \underbrace{\log_2 \frac{1}{P(s_i)}}_{\substack{\text{path length}\\ \text{in tree}}} .$$

In other words, it is the expected number of needed yes/no questions.

Perfect Question Scheme



Code length: 1.875 bit/symbol
Code efficiency: 1

Other Information-theoretic Evaluation Measures

Normalized Information Gain

- Information gain is biased towards many-valued attributes.
- Normalization removes / reduces this bias.

Information Gain Ratio (Quinlan 1986 / 1993)

$$I_{\text{gr}}(C, A) = \frac{I_{\text{gain}}(C, A)}{H_A} = \frac{I_{\text{gain}}(C, A)}{-\sum_{j=1}^{n_A} p_{.j} \log_2 p_{.j}}$$

Symmetric Information Gain Ratio (López de Mántaras 1991)

$$I_{\text{sgr}}^{(1)}(C, A) = \frac{I_{\text{gain}}(C, A)}{H_{AC}} \qquad \text{or} \qquad I_{\text{sgr}}^{(2)}(C, A) = \frac{I_{\text{gain}}(C, A)}{H_A + H_C}$$

Bias of Information Gain

- **Information gain is biased towards many-valued attributes**, i.e., of two attributes having about the same information content it tends to select the one having more values.

- The reasons are quantization effects caused by the finite number of example cases (due to which only a finite number of different probabilities can result in estimations) in connection with the following theorem:

- **Theorem:** Let A, B, and C be three attributes with finite domains and let their joint probability distribution be strictly positive, i.e., $\forall a \in \mathrm{dom}(A) : \forall b \in \mathrm{dom}(B) : \forall c \in \mathrm{dom}(C) : P(A = a, B = b, C = c) > 0$. Then

$$I_{\mathrm{gain}}(C, AB) \;\geq\; I_{\mathrm{gain}}(C, B),$$

  with equality obtaining only if the attributes C and A are conditionally independent given B, i.e., if $P(C = c \mid A = a, B = b) = P(C = c \mid B = b)$.

  (A detailed proof of this theorem can be found, for example, in [Borgelt and Kruse 2002], p. 311ff.)

A Statistical Evaluation Measure

χ^2 Measure

- Compares the actual joint distribution
  with a **hypothetical independent distribution**.
- Uses absolute comparison.
- Can be interpreted as a difference measure.

$$\chi^2(C,A) = \sum_{i=1}^{n_C} \sum_{j=1}^{n_A} N_{..} \frac{(p_{i.}p_{.j} - p_{ij})^2}{p_{i.}p_{.j}}$$

- Side remark: Information gain can also be interpreted as a difference measure.

$$I_{\text{gain}}(C,A) = \sum_{i=1}^{n_C} \sum_{j=1}^{n_A} p_{ij} \log_2 \frac{p_{ij}}{p_{i.}p_{.j}}$$

Treatment of Numeric Attributes

General Approach: Discretization

- **Preprocessing I**
    - Form equally sized or equally populated intervals.
- **During the tree construction**
    - Sort the example cases according to the attribute's values.
    - Construct a binary symbolic attribute for every possible split (values: "$\leq$ threshold" and "$>$ threshold").
    - Compute the evaluation measure for these binary attributes.
    - Possible improvements: Add a penalty depending on the number of splits.
- **Preprocessing II / Multisplits during tree construction**
    - Build a decision tree using only the numeric attribute.
    - Flatten the tree to obtain a multi-interval discretization.

Treatment of Missing Values

Induction

- Weight the evaluation measure with the fraction of cases with known values.
  - Idea: The attribute provides information only if it is known.
- Try to find a surrogate test attribute with similar properties (CART, Breiman *et al.* 1984)
- Assign the case to all branches, weighted in each branch with the relative frequency of the corresponding attribute value (C4.5, Quinlan 1993).

Classification

- Use the surrogate test attribute found during induction.
- Follow all branches of the test attribute, weighted with their relative number of cases, aggregate the class distributions of all leaves reached, and assign the majority class of the aggregated class distribution.

Pruning Decision Trees

Pruning serves the purpose

- to simplify the tree (improve interpretability),
- to avoid overfitting (improve generalization).

Basic ideas:

- Replace “bad” branches (subtrees) by leaves.
- Replace a subtree by its largest branch if it is better.

Common approaches:

- Reduced error pruning
- Pessimistic pruning
- Confidence level pruning
- Minimum description length pruning

Reduced Error Pruning

- Classify a set of new example cases with the decision tree.
  (These cases must not have been used for the induction!)
- Determine the number of errors for all leaves.
- The number of errors of a subtree is the sum of the errors of all of its leaves.
- Determine the number of errors for leaves that replace subtrees.
- If such a leaf leads to the same or fewer errors than the subtree,
  replace the subtree by the leaf.
- If a subtree has been replaced,
  recompute the number of errors of the subtrees it is part of.

Advantage:	Very good pruning, effective avoidance of overfitting.
Disadvantage:	Additional example cases needed.

Pessimistic Pruning

- Classify a set of example cases with the decision tree.
  (These cases may or may not have been used for the induction.)
- Determine the number of errors for all leaves and
  increase this number by a fixed, user-specified amount r.
- The number of errors of a subtree is the sum of the errors of all of its leaves.
- Determine the number of errors for leaves that replace subtrees
  (also increased by r).
- If such a leaf leads to the same or fewer errors than the subtree,
  replace the subtree by the leaf and recompute subtree errors.

Advantage:	No additional example cases needed.
Disadvantage:	Number of cases in a leaf has no influence.

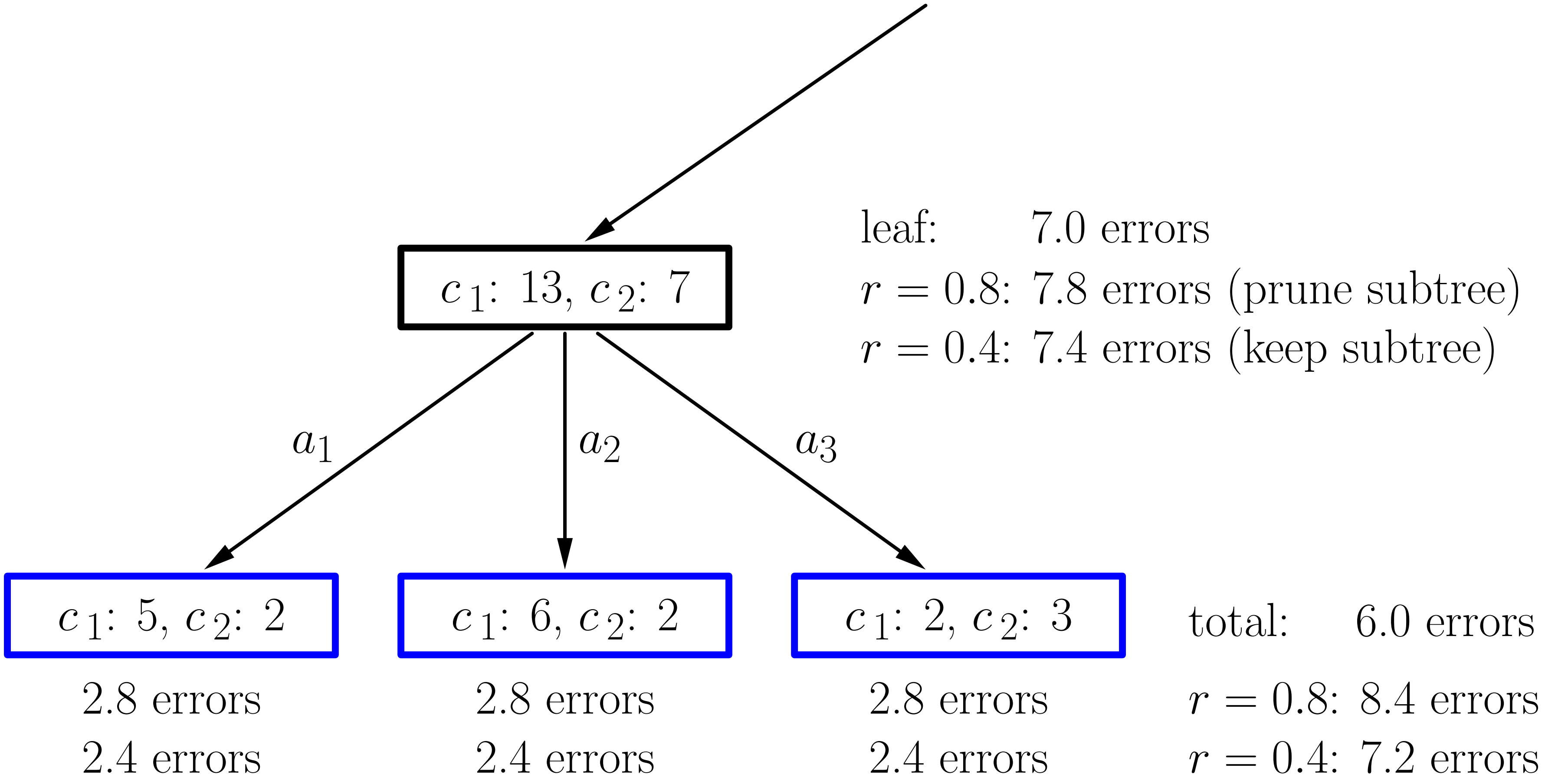
Confidence Level Pruning

- Like pessimistic pruning, but the number of errors is computed as follows:
  - See classification in a leaf as a Bernoulli experiment (error / no error).
  - Estimate an interval for the error probability based on a user-specified confidence level α.
    (use approximation of the binomial distribution by a normal distribution)
  - Increase error number to the upper level of the confidence interval
    times the number of cases assigned to the leaf.
  - Formal problem: Classification is not a random experiment.

Advantage:	No additional example cases needed, good pruning.
Disadvantage:	Statistically dubious foundation.

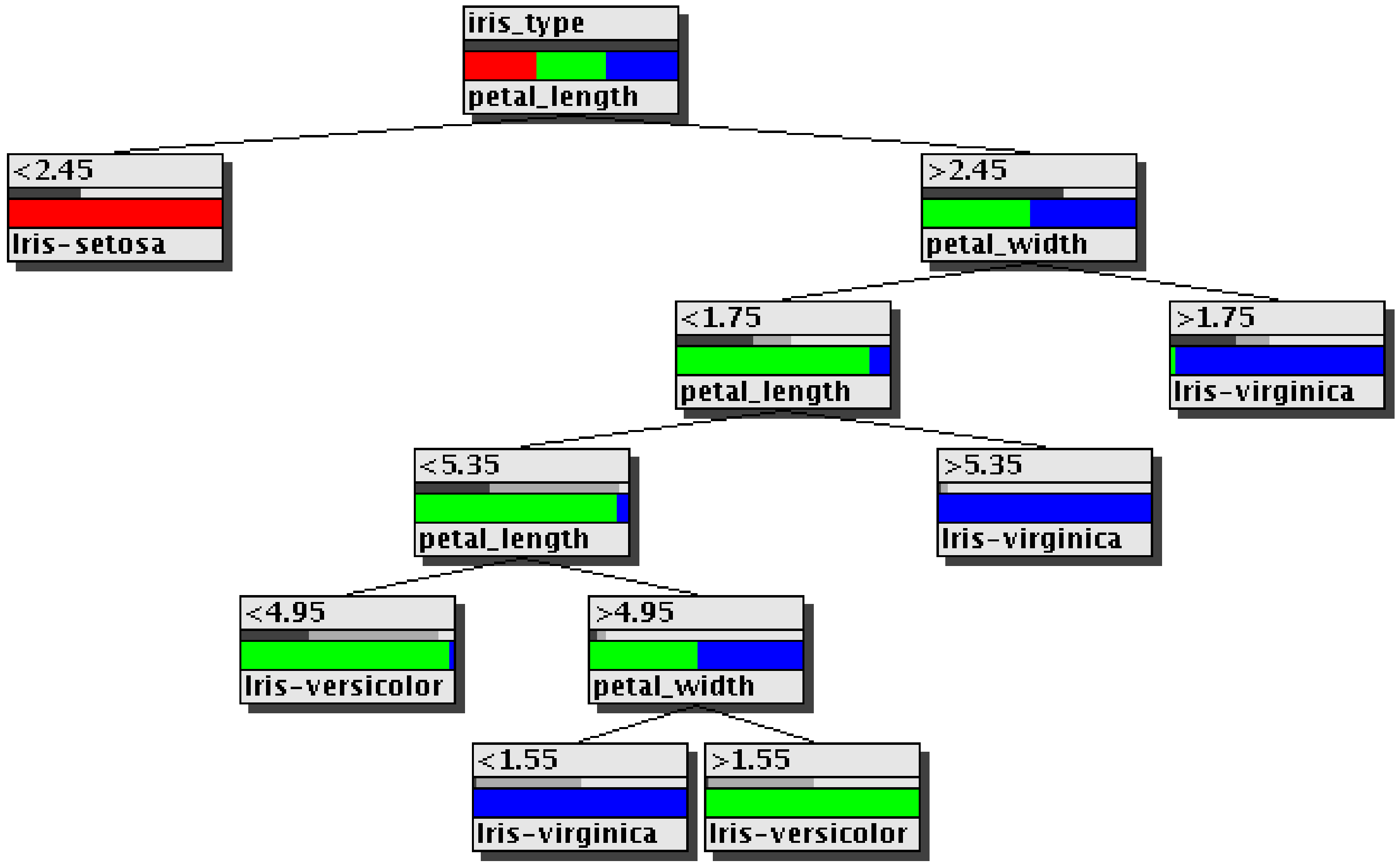
Pruning a Decision Tree: A Simple Example

Pessimistic Pruning with $r = 0.8$ and $r = 0.4$:



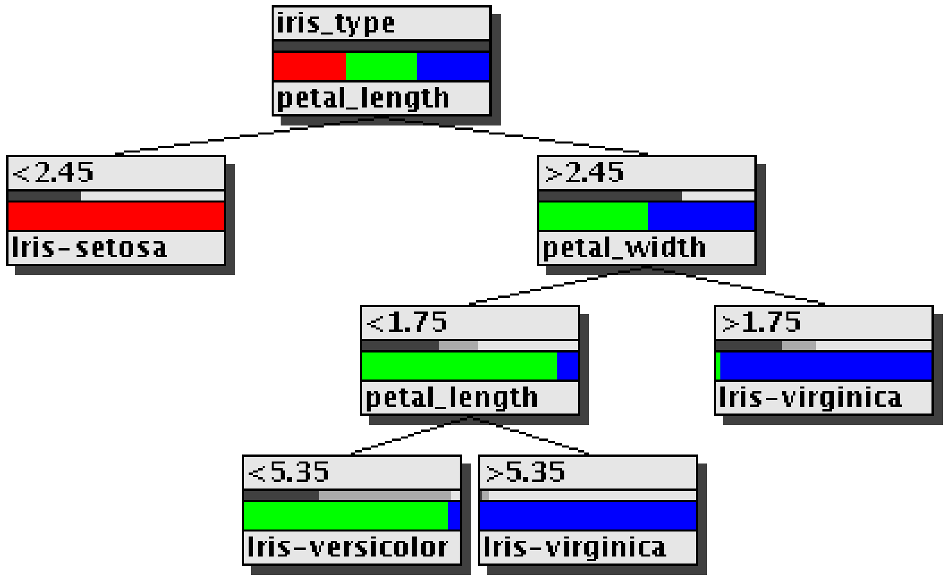
Decision Trees: An Example

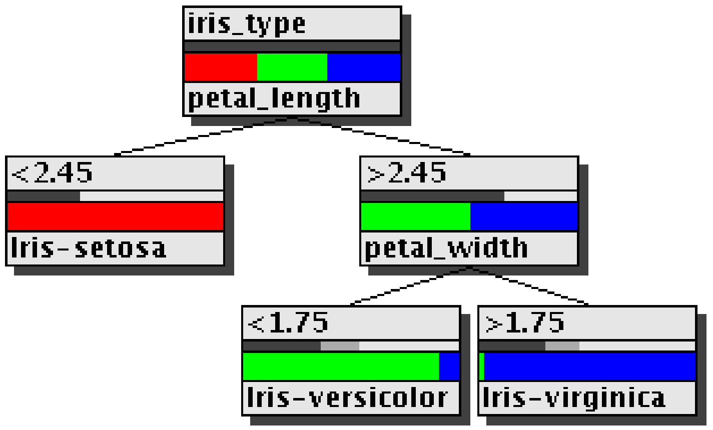
A decision tree for the Iris data
(induced with information gain ratio, unpruned)



Decision Trees: An Example

A decision tree for the Iris data
(pruned with confidence level pruning, $\alpha = 0.8$, and pessimistic pruning, $r = 2$)

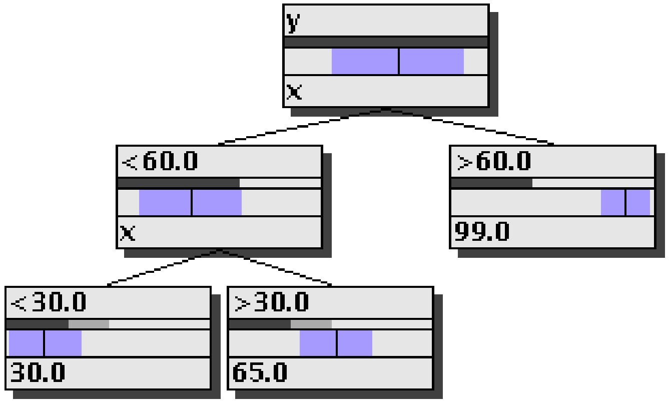




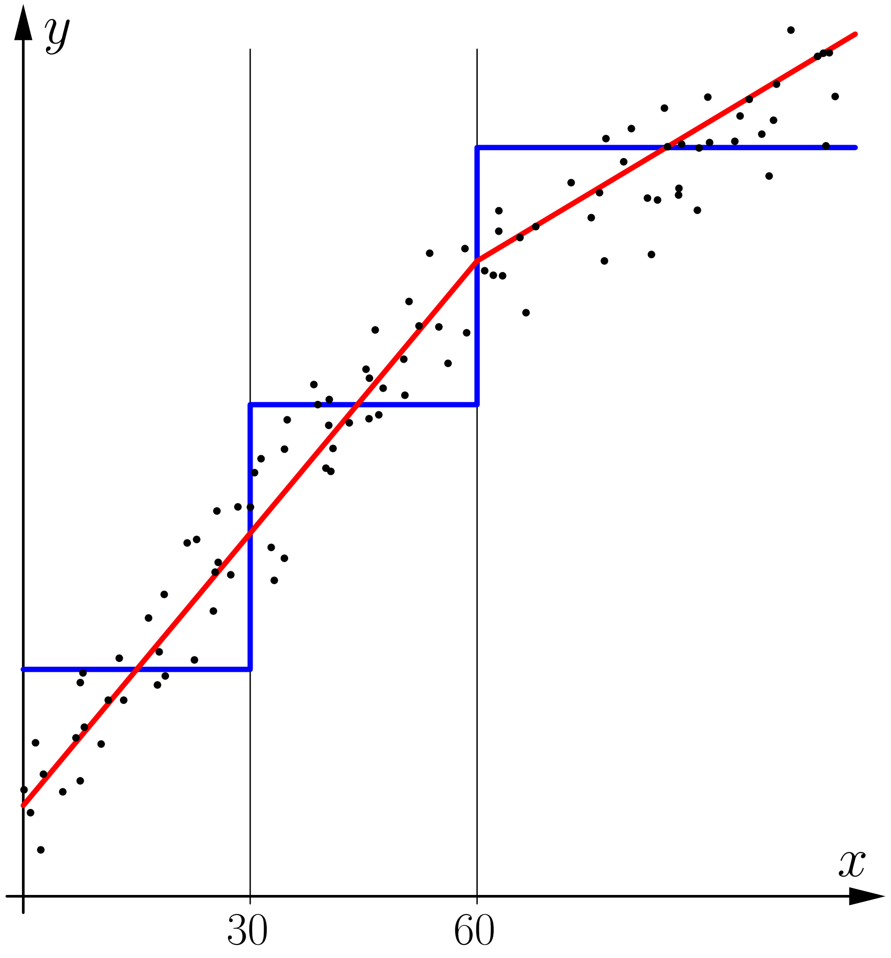
- Left: 7 instead of 11 nodes, 4 instead of 2 misclassifications.
- Right: 5 instead of 11 nodes, 6 instead of 2 misclassifications.
- The right tree is “minimal” for the three classes.

Regression Trees

- Target variable is not a class, but a numeric quantity.
- Simple regression trees: predict constant values in leaves. (blue lines)

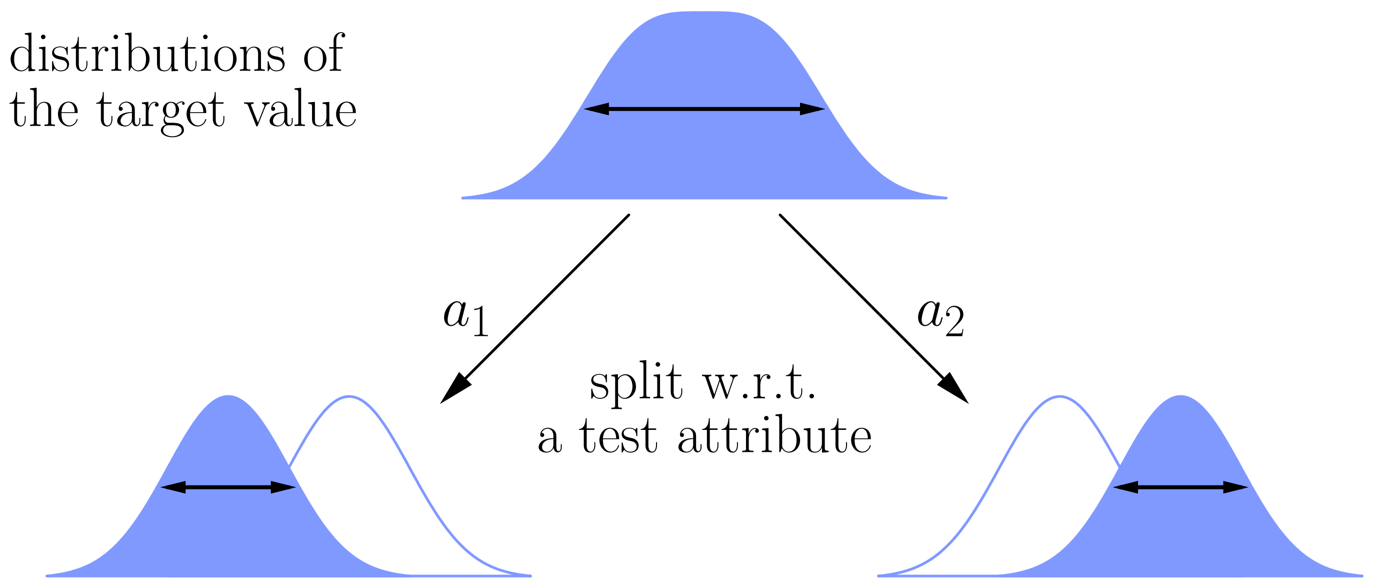


- More complex regression trees: predict linear functions in leaves. (red line)



x: input variable, y: target variable

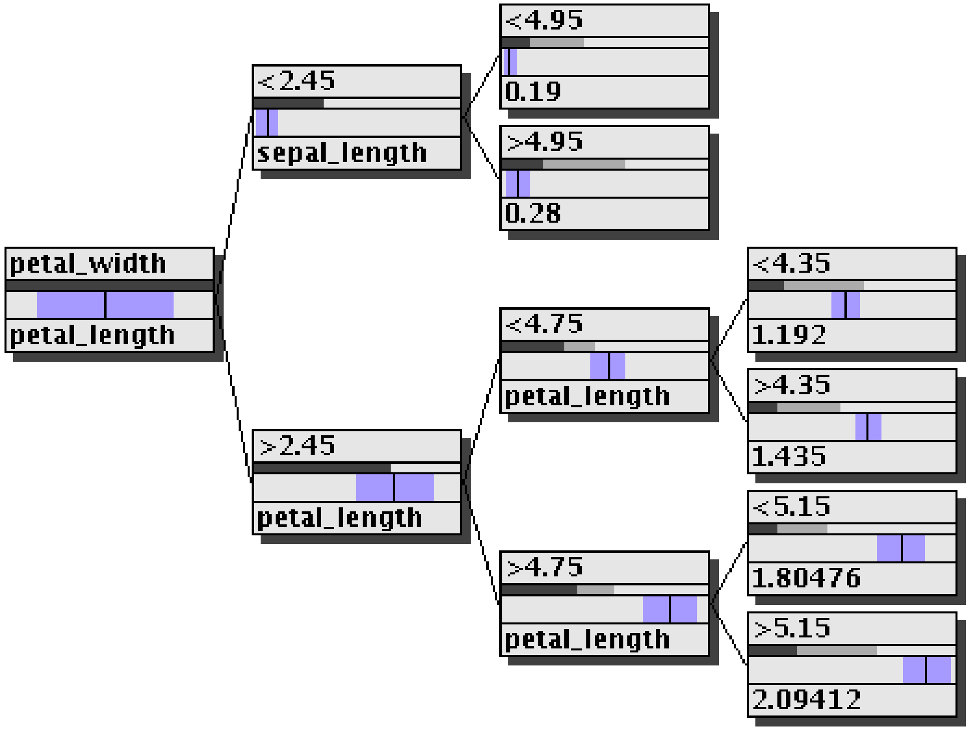
Regression Trees: Attribute Selection

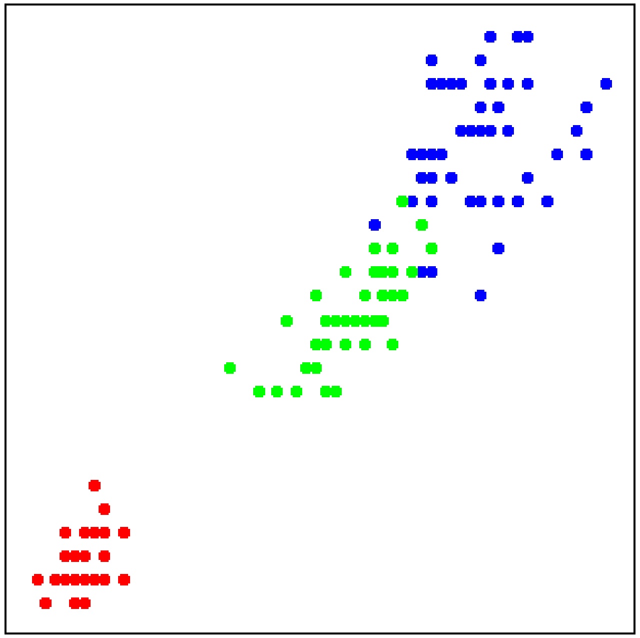

distributions of the target value
a_1
a_2
split w.r.t. a test attribute

- The variance / standard deviation is compared to the variance / standard deviation in the branches.
- The attribute that yields the highest reduction is selected.

Regression Trees: An Example

A regression tree for the Iris data (petal width)
(induced with reduction of sum of squared errors)





Summary Decision and Regression Trees

- **Decision Trees are Classifiers with Tree Structure**
  - Inner node: Test of a descriptive attribute
  - Leaf node: Assignment of a class
- **Induction of Decision Trees from Data**
  (Top-Down Induction of Decision Trees, TDIDT)
  - *Divide and conquer* approach / *recursive descent*
  - *Greedy* selection of the test attributes
  - Attributes are selected based on an *evaluation measure*, e.g. information gain, χ^2 measure
  - Recommended: *Pruning* of the decision tree
- **Numeric Target: Regression Trees**

Classification Evaluation: Cross Validation

- General method to evaluate / predict the performance of classifiers.
- Serves the purpose to estimate the error rate on new example cases.
- Procedure of cross validation:
    - Split the given data set into n so-called *folds* of equal size (n-fold cross validation).
    - Combine $n - 1$ folds into a training data set, build a classifier, and test it on the n-th fold.
    - Do this for all n possible selections of $n - 1$ folds and average the error rates.
- Special case: Leave-1-out cross validation. (use as many folds as there are example cases)
- Final classifier is learned from the full data set.

Clustering

Clustering

- **General Idea of Clustering**
  - Similarity and distance measures
- **Prototype-based Clustering**
  - Classical c-means clustering
  - Learning vector quantization
  - Fuzzy c-means clustering
  - Expectation maximization for Gaussian mixtures
- **Hierarchical Agglomerative Clustering**
  - Merging clusters: Dendrograms
  - Measuring the distance of clusters
  - Choosing the clusters
- **Summary**

General Idea of Clustering

- Goal: Arrange the given data tuples into **classes** or **clusters**.
- Data tuples assigned to the same cluster should be as similar as possible.
- Data tuples assigned to different clusters should be as dissimilar as possible.
- Similarity is most often measured with the help of a distance function.
  (The smaller the distance, the more similar the data tuples.)
- Often: restriction to data points in $\mathbb{R}^m$ (although this is not mandatory).

$d : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_0^+$ is a **distance function** if it satisfies $\forall \vec{x}, \vec{y}, \vec{z} \in \mathbb{R}^m$:

$$\begin{array}{rll}
(i) & d(\vec{x}, \vec{y}) = 0 \quad \Leftrightarrow \quad \vec{x} = \vec{y}, & \\
(ii) & d(\vec{x}, \vec{y}) = d(\vec{y}, \vec{x}) & \text{(symmetry)}, \\
(iii) & d(\vec{x}, \vec{z}) \leq d(\vec{x}, \vec{y}) + d(\vec{y}, \vec{z}) & \text{(triangle inequality)}.
\end{array}$$

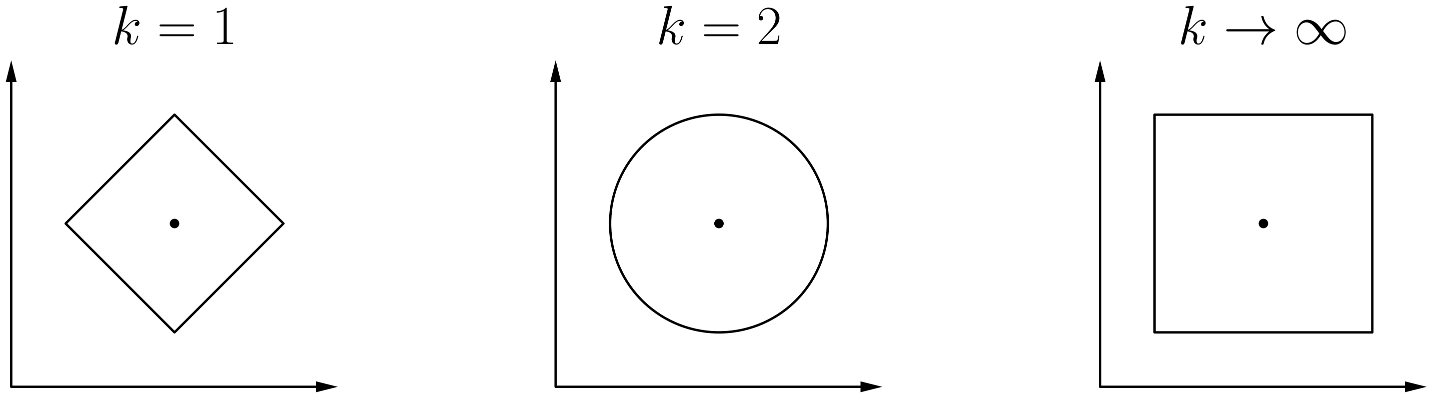
Distance Functions

Illustration of distance functions: Minkowski Family

$$d_k(\vec{x},\vec{y}) = \left(\sum_{i=1}^{n}(x_i - y_i)^k\right)^{\frac{1}{k}}$$

Well-known special cases from this family are:

$k = 1:$	Manhattan or city block distance,
$k = 2:$	Euclidean distance,
$k \to \infty:$	maximum distance, i.e. $d_\infty(\vec{x},\vec{y}) = \max_{i=1}^{n}\lvert x_i - y_i\rvert$.

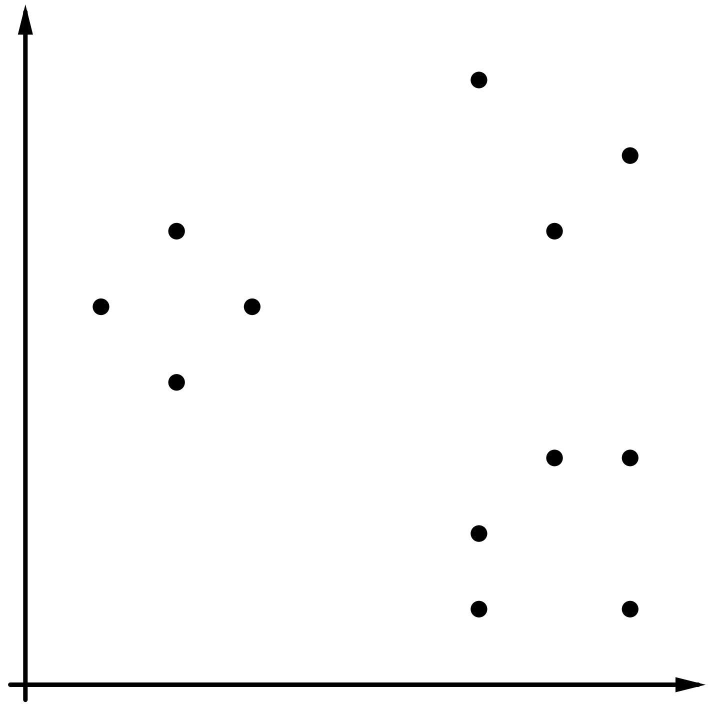


c-Means Clustering

- Choose a number c of clusters to be found (user input).
- Initialize the cluster centers randomly
  (for instance, by randomly selecting c data points).
- **Data point assignment**:
  Assign each data point to the cluster center that is closest to it
  (i.e. closer than any other cluster center).
- **Cluster center update**:
  Compute new cluster centers as the mean vectors of the assigned data points.
  (Intuitively: center of gravity if each data point has unit weight.)
- Repeat these two steps (data point assignment and cluster center update)
  until the clusters centers do not change anymore.

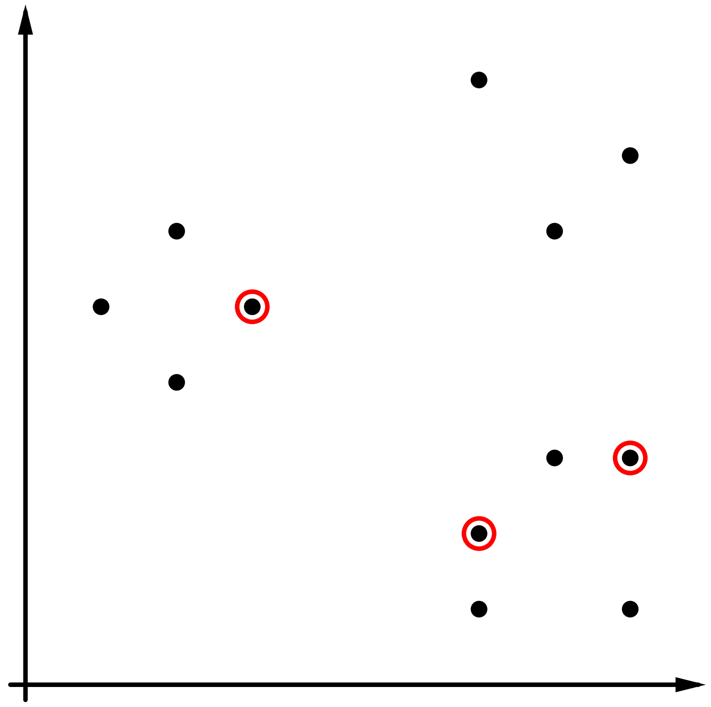
- It can be shown that this scheme must converge,
  i.e., the update of the cluster centers cannot go on forever.

c-Means Clustering: Example



Data set to cluster.

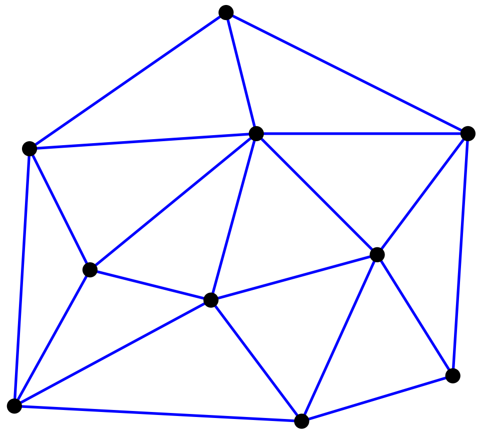
Choose $c = 3$ clusters.
(From visual inspection, can be difficult to determine in general.)

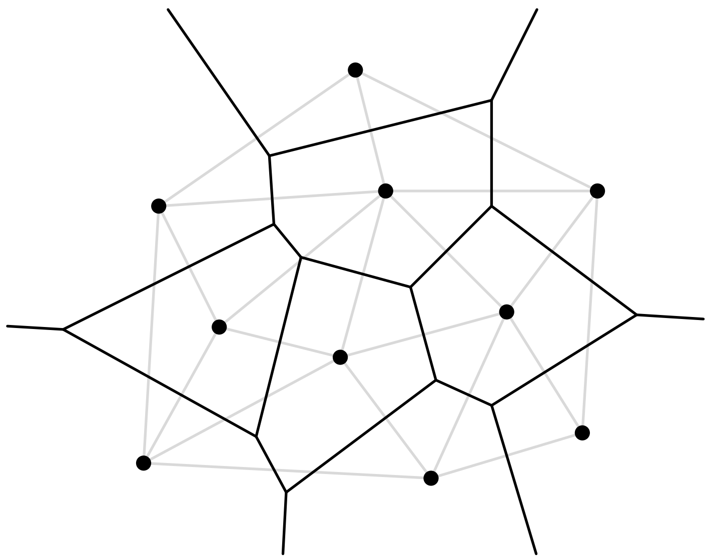


Initial position of cluster centers.

Randomly selected data points.
(Alternative methods include e.g. latin hypercube sampling)

Delaunay Triangulations and Voronoi Diagrams

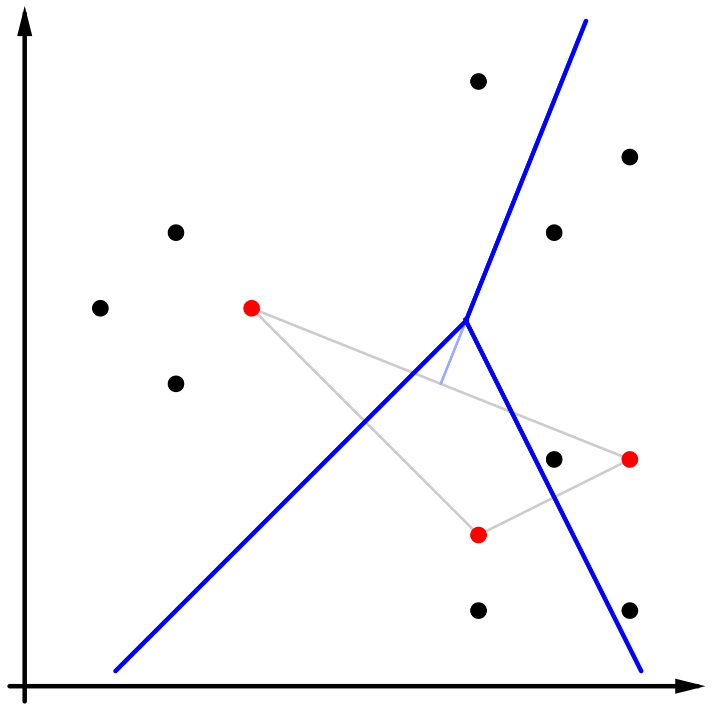


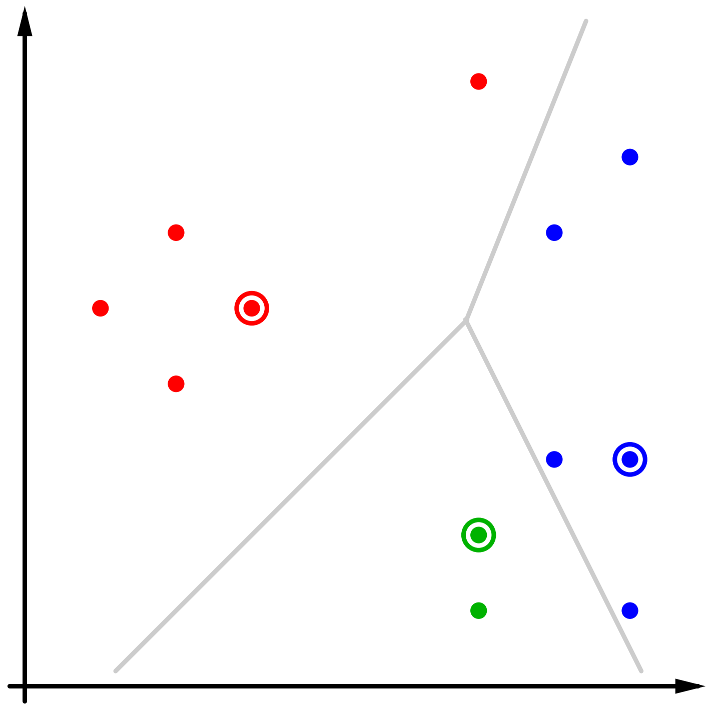


- Dots represent cluster centers (quantization vectors).
- Left: **Delaunay Triangulation**
  (The circle through the corners of a triangle does not contain another point.)
- Right: **Voronoi Diagram**
  (Midperpendiculars of the Delaunay triangulation: boundaries of the regions of points that are closest to the enclosed cluster center (Voronoi cells)).

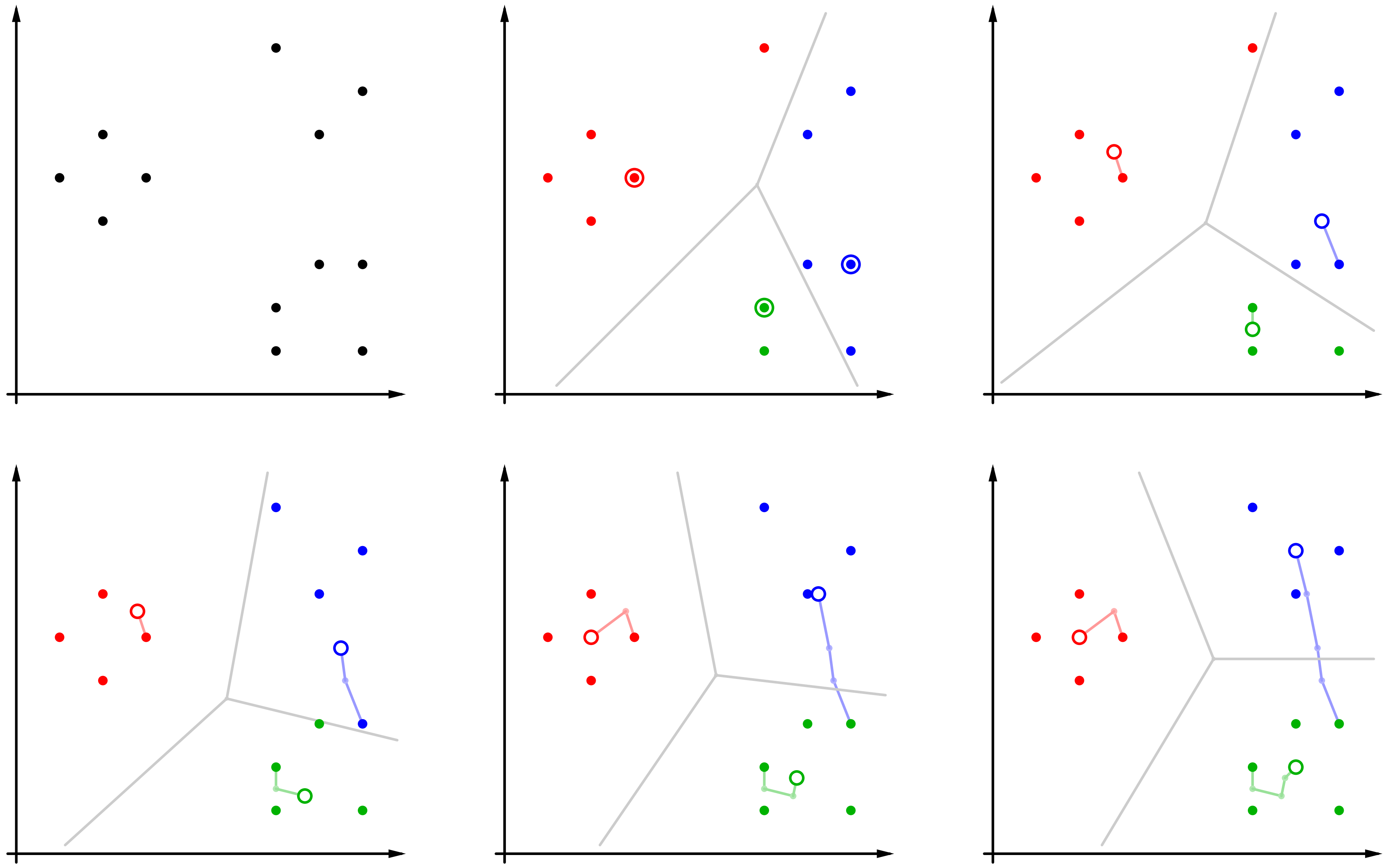
Delaunay Triangulations and Voronoi Diagrams

- **Delaunay Triangulation:** simple triangle (shown in grey on the left)
- **Voronoi Diagram:** midperpendiculars of the triangle's edges (shown in blue on the left, in grey on the right)





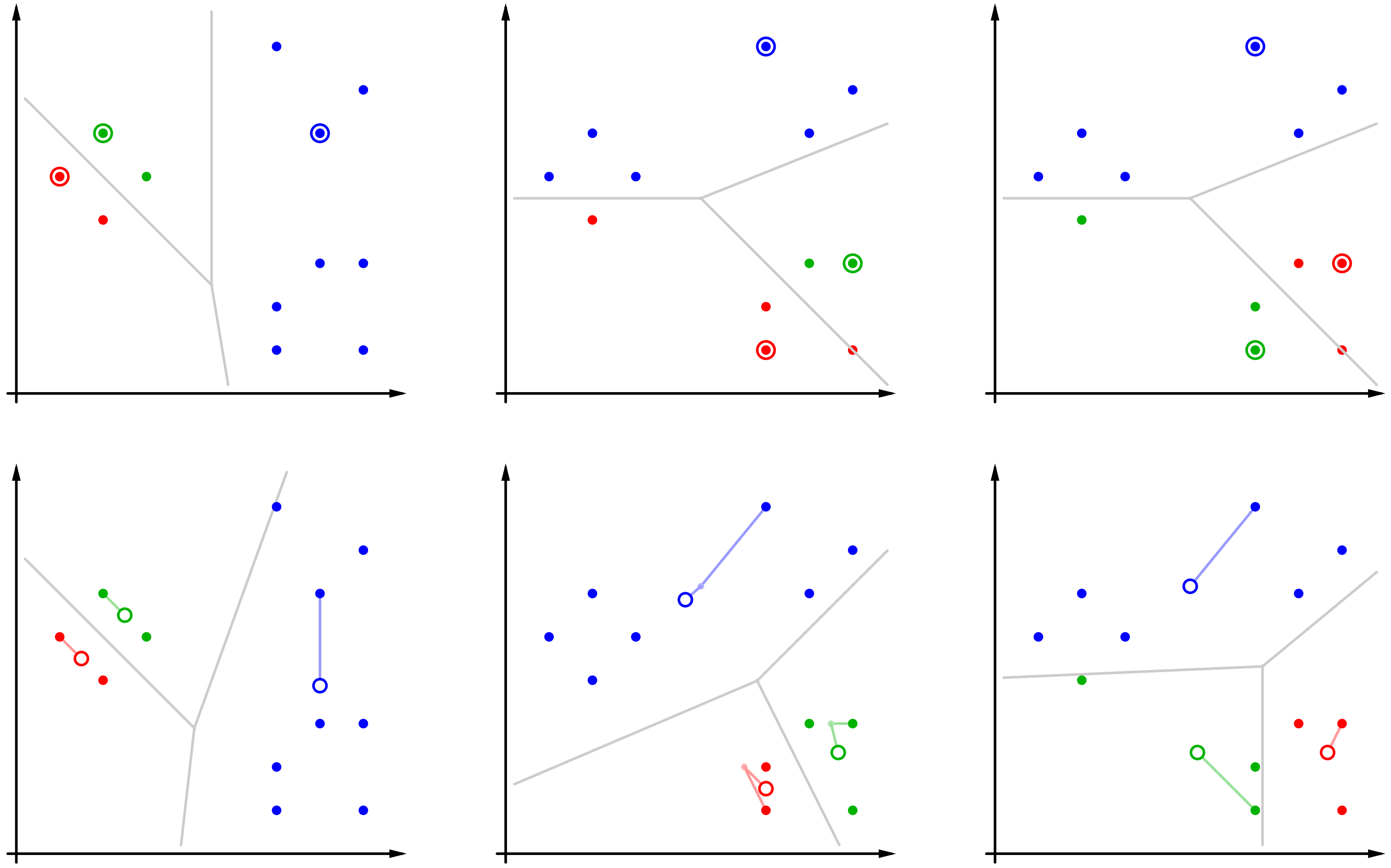
c-Means Clustering: Example



c-Means Clustering: Local Minima

- Clustering is successful in this example:
  The clusters found are those that would have been formed intuitively.
- Convergence is achieved after only 5 steps.
  (This is typical: convergence is usually very fast.)
- However: The clustering result is fairly **sensitive to the initial positions** of the cluster centers.
- With a bad initialization clustering may fail
  (the alternating update process gets stuck in a local minimum).
- Fuzzy c-means clustering and the estimation of a mixture of Gaussians are much more robust (to be discussed later).
- Research issue: Can we determine the number of clusters automatically?
  (Some approaches exists, but none of them is too successful.)

c-Means Clustering: Local Minima



Learning Vector Quantization

Adaptation of reference vectors / codebook vectors

- Like “online” c-means clustering (update after each data point).
- For each training pattern find the closest reference vector.
- Adapt only this reference vector (winner neuron).
- For classified data the class may be taken into account.
  (reference vectors are assigned to classes)

Attraction rule (data point and reference vector have same class)

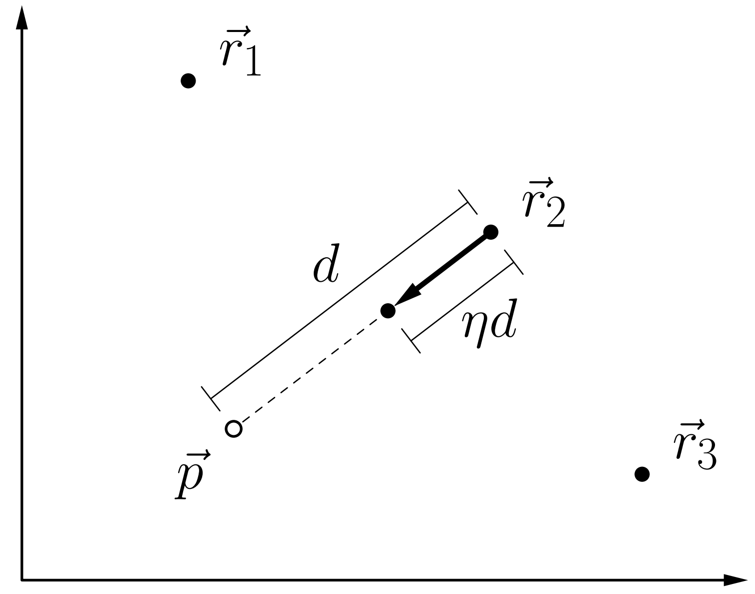
$$\vec{r}^{\,(\text{new})} = \vec{r}^{\,(\text{old})} + \eta\Big(\vec{p} - \vec{r}^{\,(\text{old})}\Big),$$

Repulsion rule (data point and reference vector have different class)

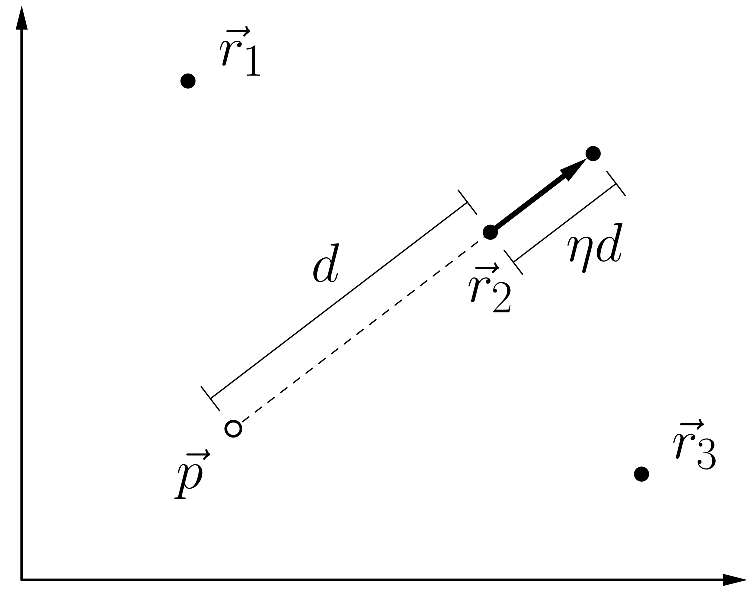
$$\vec{r}^{\,(\text{new})} = \vec{r}^{\,(\text{old})} - \eta\Big(\vec{p} - \vec{r}^{\,(\text{old})}\Big).$$

Learning Vector Quantization

Adaptation of reference vectors / codebook vectors



attraction rule

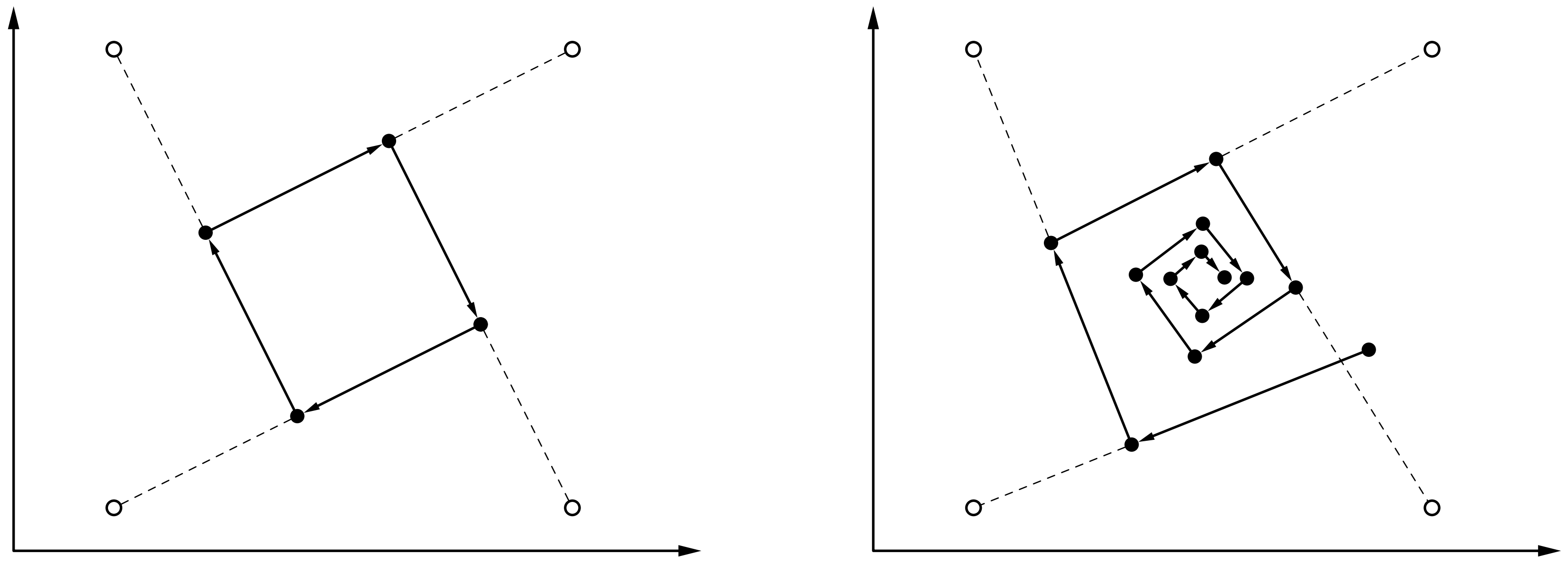


repulsion rule

- $\vec{p}$: data point, $\vec{r}_i$: reference vector
- $\eta = 0.4$ (learning rate)

Learning Vector Quantization: Learning Rate Decay

Problem: fixed learning rate can lead to oscillations

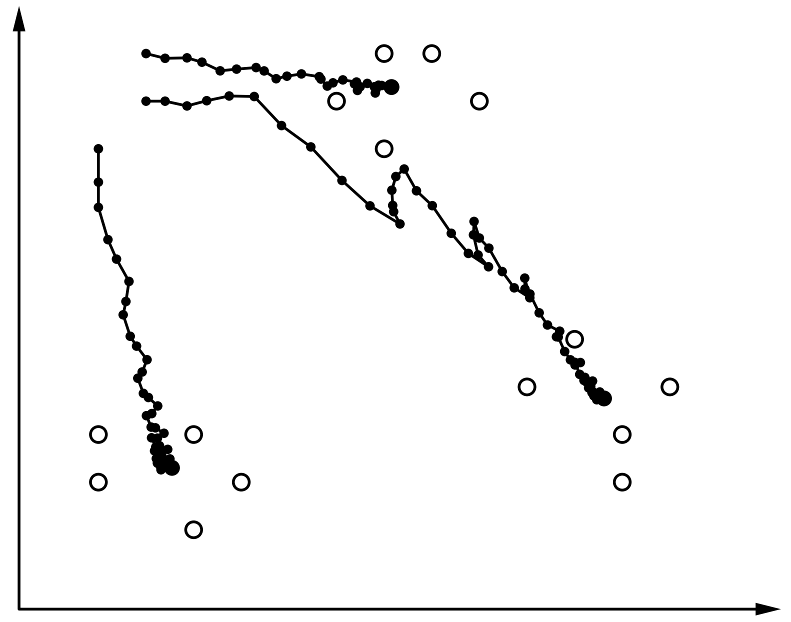


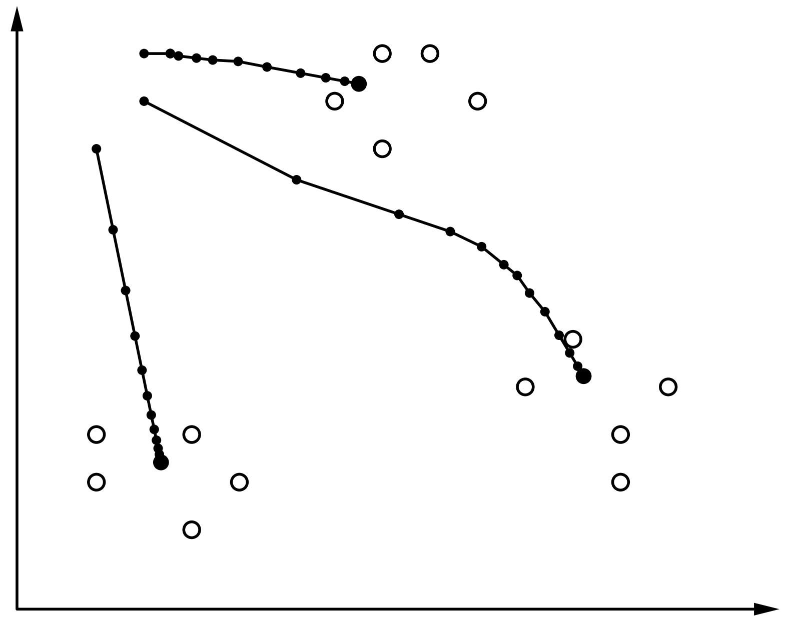
Solution: **time dependent learning rate**

$$\eta(t) = \eta_0 \alpha^t, \quad 0 < \alpha < 1, \qquad \text{or} \qquad \eta(t) = \eta_0 t^\kappa, \quad \kappa < 0.$$

Learning Vector Quantization: Example

Adaptation of reference vectors / codebook vectors





- Left: Online training with learning rate $\eta = 0.1$,
- Right: Batch training with learning rate $\eta = 0.05$.

Excursus: Short Introduction to Fuzzy Theory

- **Classical Logic:** only truth values *true* and *false*.
  Classical Algebra: either *is element* or *is not element*.
- The bivalence of the classical theories is often not appropriate.

  Illustrating example: **Sorites Paradoxon** (Greek *sorites*: heap)

  - One billion grains of sand is a heap of sand. *(true)*
  - If you remove a grain of sand from a heap of sand,
    a heap of sand remains. *(true)*

  Thus it follows:

  - 999 999 999 grains are a heap of sand. *(true)*

  Multiple repetition of the same conclusion finally leads to

  - 1 grain of sand is a heap of sand. *(false!)*

  At which number of grains the conclusion does not preserve the truth?

Excursus: Short Introduction to Fuzzy Theory

- Obviously: There is no exact determined number of grains of sand, where the conclusion to the next smaller number is wrong.
- Problem: Terms of natural language (e.g., "heap of sand", "bald headed", "warm", "fast", "high pressure", "light" etc.) are **vague**.
- Note: Though vague terms may be *inexact*, but not *useless*.
    - Also for vague terms there exist situations/objects, where they are *for sure applicable* and where they are *for sure not applicable*.
    - Between this a **penumbra** (Latin *semi-shade*) of situations is located, where it is unclear, whether the terms are applicable, or whether they are only applicable with restrictions ("small heap of sand").
    - The fuzzy theory tries to model this penumbra mathematically ("soft transition" between *applicable* and *not applicable*).

Excursus: Fuzzy Logic

- **Fuzzy logic** is an extension of the classical logic to intermediate values between *true* and *false*.
- As truth value, any value from the real interval $[0, 1]$ can appear, whereas $0 \mathrel{\widehat{=}}$ *false* and $1 \mathrel{\widehat{=}}$ *true*.
- Thus necessary: **Extension of the logical operators**

◦ Negation	classical: $\neg a$,	fuzzy: $\sim a$	Fuzzy Negation
◦ Conjunction	classical: $a \wedge b$,	fuzzy: $\top(a, b)$	t-Norm
◦ Disjunction	classical: $a \vee b$,	fuzzy: $\bot(a, b)$	t-Conorm

- **Basic principles** of the extension:
  - For the extreme values 0 and 1, the operators shall behave the same way like the classical examples (marginal/edge constraints).
  - For the intermediate values, the behavior shall be monotone.
  - As far as possible the laws of the classical logical shall be preserved.

Fuzzy Clustering

- Allow degrees of membership of a datum to different clusters.
  (Classical c-means clustering assigns data crisply.)

Objective Function: (to be minimized)

$$J(\mathbf{X}, \mathbf{B}, \mathbf{U}) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} d^{\,2}(\vec{\beta}_i, \vec{x}_j)$$

- $\mathbf{U} = [u_{ij}]$ is the $c \times n$ fuzzy partition matrix,
  $u_{ij} \in [0, 1]$ is the membership degree of the data point $\vec{x}_j$ to the i-th cluster.
- $\mathbf{B} = \{\vec{\beta}_1, \ldots, \vec{\beta}_c\}$ is the set of cluster prototypes.
- w is the so-called "fuzzifier" (the higher w, the softer the cluster boundaries).
- Constraints:
  $\forall i \in \{1, \ldots, c\}: \quad \sum_{j=1}^{n} u_{ij} > 0 \quad$ and $\quad \forall j \in \{1, \ldots, n\}: \quad \sum_{i=1}^{c} u_{ij} = 1.$

Fuzzy and Hard Clustering

Relation to Classical c-Means Clustering:

- c-means clustering can be seen as optimizing the objective function

$$J(\mathbf{X}, \mathbf{B}, \mathbf{U}) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}\, d^2(\vec{\beta}_i, \vec{x}_j),$$

  where $\forall i, j : u_{ij} \in \{0, 1\}$ (i.e. hard assignment of the data points) and the cluster prototypes $\vec{\beta}_i$ consist only of cluster centers.

- To obtain a fuzzy assignment of the data points, it is not enough to extend the range of values for the u_{ij} to the unit interval $[0, 1]$: The objective function J is optimized for a hard assignment (each data point is assigned to the closest cluster center).

- **Necessary for degrees of membership:**
  Apply a convex function $h : [0, 1] \to [0, 1]$ to the membership degrees u_{ij}.
  Most common choice: $h(u) = u^w$, usually with $w = 2$.

Reminder: Function Optimization

Task: Find values $\vec{x} = (x_1, \dots, x_m)$ such that $f(\vec{x}) = f(x_1, \dots, x_m)$ is optimal.

Often feasible approach:

- A necessary condition for a (local) optimum (maximum or minimum) is that the partial derivatives w. r. t. the parameters vanish (Pierre Fermat).
- Therefore: (Try to) solve the equation system that results from setting all partial derivatives w. r. t. the parameters equal to zero.

Example task: Minimize $\quad f(x, y) = x^2 + y^2 + xy - 4x - 5y.$

Solution procedure:

1. Take the partial derivatives of the objective function and set them to zero:

$$\frac{\partial f}{\partial x} = 2x + y - 4 = 0, \qquad \frac{\partial f}{\partial y} = 2y + x - 5 = 0.$$

2. Solve the resulting (here: linear) equation system: $\qquad x = 1, \quad y = 2.$

Function Optimization with Constraints

Often a function has to be optimized subject to certain **constraints**.

Here: restriction to k **equality constraints** $C_i(\vec{x}) = 0$, $i = 1, \ldots, k$.

Note: the equality constraints describe a subspace of the domain of the function.

Problem of optimization with constraints:

- The gradient of the objective function f may vanish outside the constrained subspace, leading to an unacceptable solution (violating the constraints).
- At an optimum *in the constrained subspace* the derivatives need not vanish.

One way to handle this problem are **generalized coordinates**:

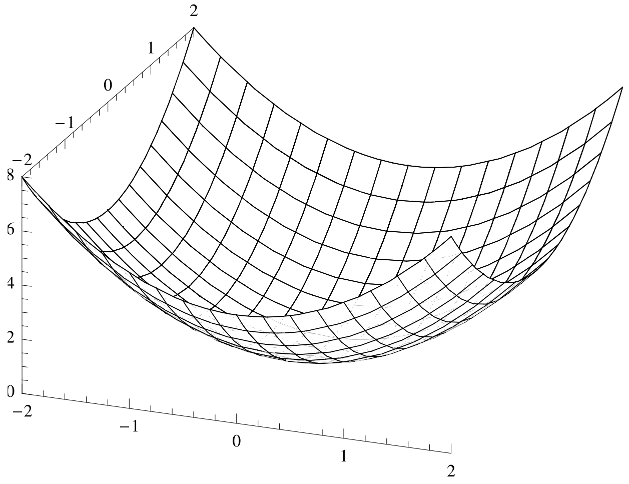
- Exploit the dependence between the parameters specified in the constraints to express some parameters in terms of the others and thus reduce the set $\vec{x}$ to a set $\vec{x}'$ of independent parameters (*generalized coordinates*).
- Problem: Can be clumsy and cumbersome, if possible at all, because the form of the constraints may not allow for expressing some parameters as proper functions of the others.

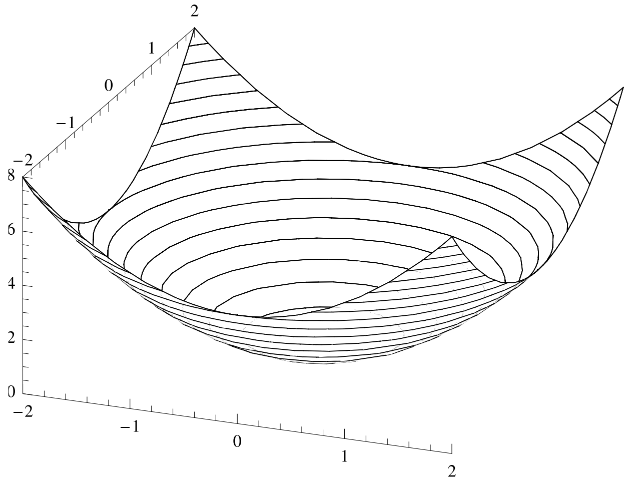
Contour Lines of a Function

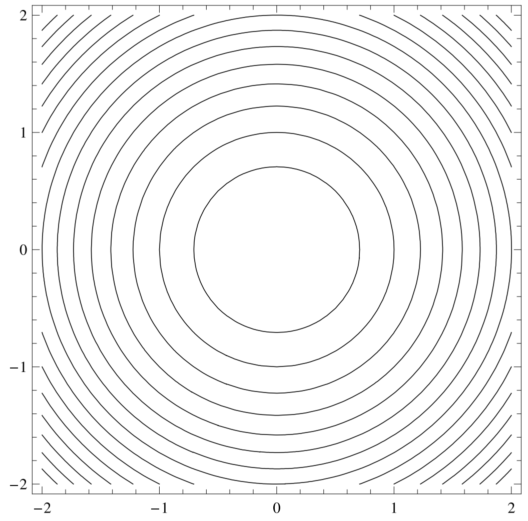
Contour Lines: Given a function $f : \mathbb{R}^2 \to \mathbb{R}$, the contour plot is obtained by drawing the contour sets for equidistant levels, i. e., plot the following sets of points:

$$M_{kc} = \{(x_1, x_2) \in \mathbb{R}^2 \mid f(x_1, x_2) = kc\}$$
$$\text{for } k \in \mathbb{N} \text{ and fixed } c \in \mathbb{R}_{\geq 0}$$

Example: $f(x_1, x_2) = x_1^2 + x_2^2$







Gradient Field of a Function

- The gradient of a function $f\colon \mathbb{R}^n \to \mathbb{R}$ consists of the vector of its partial derivatives w. r. t. the arguments:

$$\nabla_{\vec{x}}\, f \;=\; \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right)^\top$$

- The gradient evaluated at a point $\vec{x}^*$, written as

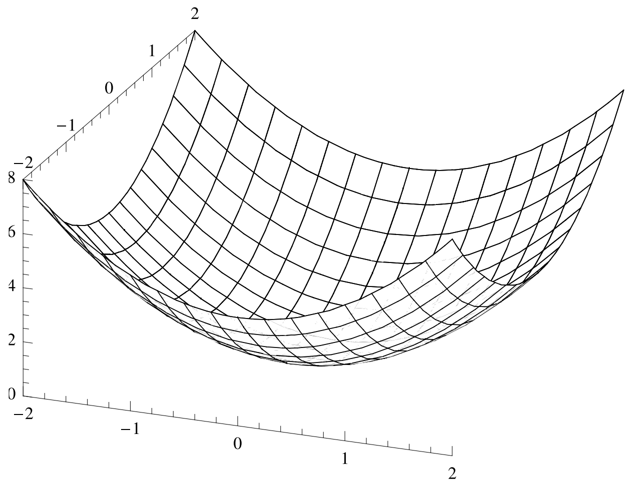
$$\nabla_{\vec{x}}\, f|_{\vec{x}^*} \;=\; \left(\left.\frac{\partial f}{\partial x_1}\right|_{x_1^*}, \ldots, \left.\frac{\partial f}{\partial x_n}\right|_{x_n^*}\right)^\top,$$

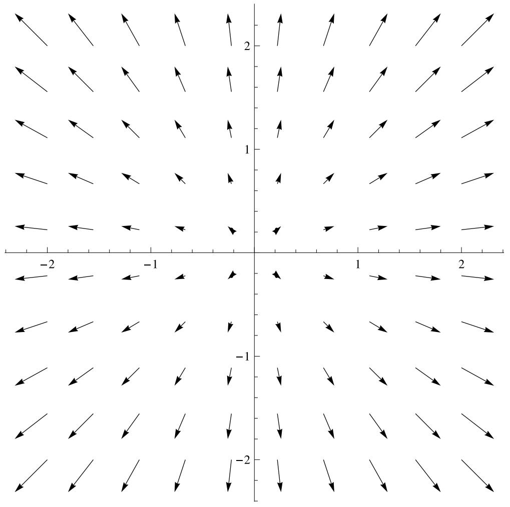
  points into the direction of largest increase of f.

- Formally, the gradient of a function with domain $\mathbb{R}^n$ has n dimensions although it is often depicted as an $n+1$-dimensional vector.

Gradient Field of a Function: Examples

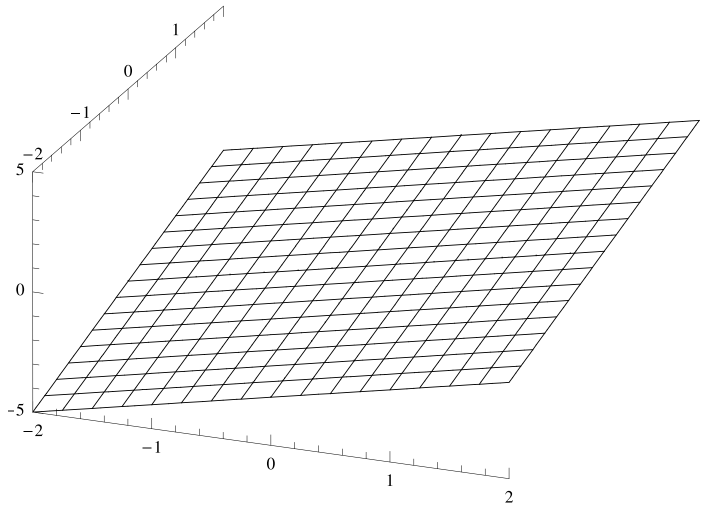
$$f_1(x_1, x_2) = x_1^2 + x_2^2$$

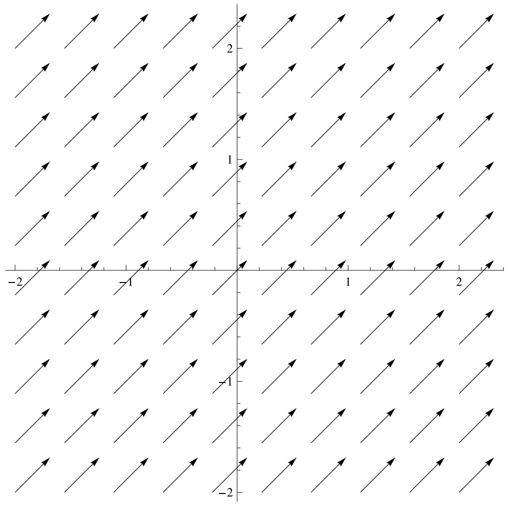




$$\nabla_{\vec{x}} f_1 = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix}$$

$$f_2(x_1, x_2) = x_1 + x_2 - 1$$





$$\nabla_{\vec{x}} f_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Function Optimization with Constraints

Problem: If the global optimum of f lies outside the feasible region the gradient does not vanish at the constrained optimum $\vec{x}^*$.

Which criteria do hold at the constrained optimum?

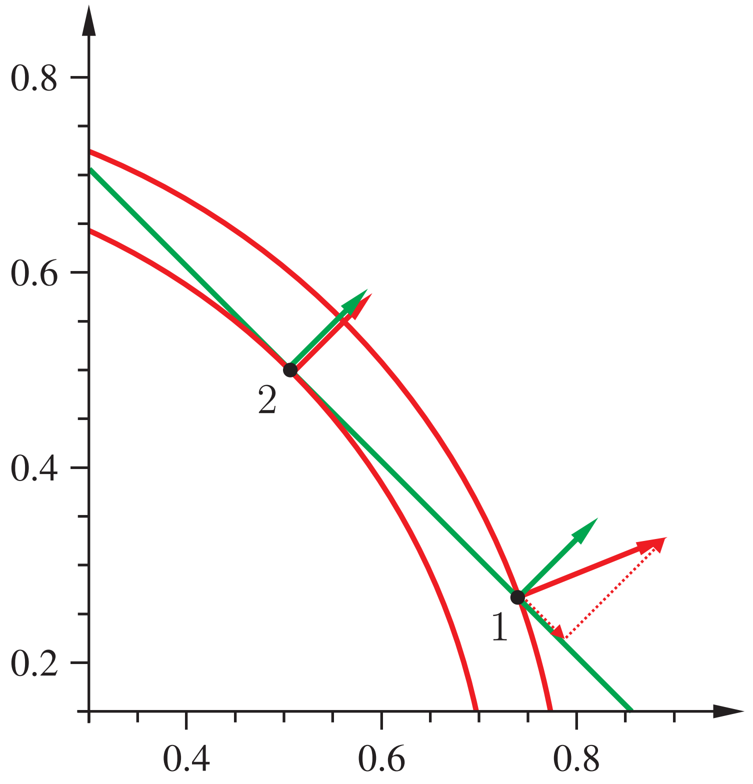
- Assume we move $\vec{x}^*$ throughout the feasible region to find the optimum "manually". If we *cross* a contour line of f, the crossing point cannot be an optimum: because crossing a contour line means descending or ascending.
- However, if we *touch* a contour line we have found an optimum because stepping backward or forward will increase (or decrease) the value.
- At the "touching point" $\vec{x}^*$ the gradient of f and the gradient of g are parallel.

$$\nabla f = \lambda \nabla g$$

- We only need both gradients to be parallel. Since they can have opposite directions and different lengths λ is used to rescale ∇g.

Example

Task: Minimize $f(x_1, x_2) = x_1^2 + x_2^2$ subject to $g: x + y = 1$.



- **Crossing a contour line**: Point 1 cannot be a constrained minimum because ∇f has a non-zero component in the constrained space. Walking in opposite direction to this component can further decrease f.

- **Touching a contour line**: Point 2 is a constrained minimum: both gradients are parallel, hence there is no component of ∇f in the constrained space that might lead us to a lower value of f.

Function Optimization with Constraints

- Therefore, at the constrained optimum $\vec{x}^*$ we require:

$$\nabla f(\vec{x}^*) = \lambda \nabla g(\vec{x}^*) \qquad \text{and} \qquad g(\vec{x}^*) = 0$$

- More compact representation:

$$L(\vec{x}, \lambda) = f(\vec{x}) - \lambda g(\vec{x}) \qquad \text{and} \qquad \nabla L = 0$$

- Taking the partial derivatives reveals the initial conditions:

$$\begin{aligned} \frac{\partial}{\partial \vec{x}} L(\vec{x}, \lambda) &= \nabla f(\vec{x}) - \lambda \nabla g(\vec{x}) = 0 \\ \nabla f(\vec{x}) &= \lambda \nabla g(\vec{x}) \end{aligned}$$

$$\frac{\partial}{\partial \lambda} L(\vec{x}, \lambda) = g(\vec{x}) = 0$$

- The negative sign in the Lagrange function L can be incorporated into λ, i. e. we will from now on replace it by a positive sign.

Lagrange Theory: Example 1

Example task: Minimize $\quad f(x, y) = x^2 + y^2 \quad$ subject to $\quad x + y = 1.$

Solution procedure:

1. Rewrite the constraint, so that one side gets zero: $x + y - 1 = 0.$
2. Construct the Lagrange function by incorporating the constraint into the objective function with a Lagrange multiplier λ:
$$L(x, y, \lambda) = x^2 + y^2 + \lambda(x + y - 1).$$
3. Take the partial derivatives of the Lagrange function and set them to zero (necessary conditions for a minimum):
$$\frac{\partial L}{\partial x} = 2x + \lambda = 0, \qquad \frac{\partial L}{\partial y} = 2y + \lambda = 0, \qquad \frac{\partial L}{\partial \lambda} = x + y - 1 = 0.$$
4. Solve the resulting (here: linear) equation system:
$$\lambda = -1, \qquad x = y = \tfrac{1}{2}.$$

Summary: Function Optimization with Constraints

Let $\vec{x}^*$ be a (local) optimum of $f(\vec{x})$ *in the constrained subspace.* Then:

- The gradient $\nabla_{\vec{x}} f(\vec{x}^*)$, if it does not vanish, must be **perpendicular** to the constrained subspace. (If $\nabla_{\vec{x}} f(\vec{x}^*)$ had a component in the constrained subspace, $\vec{x}^*$ would not be a (local) optimum in this subspace.)
- The gradients $\nabla_{\vec{x}}\, g_j(\vec{x}^*)$, $1 \le j \le k$, must all be **perpendicular** to the constrained subspace, because they are constant, namely 0, in this subspace. Together they span the subspace perpendicular to the constrained subspace.
- Therefore it must be possible to find values λ_j, $1 \le j \le k$, such that

$$\nabla_{\vec{x}} f(\vec{x}^*) + \sum_{j=1}^{s} \lambda_j \nabla_{\vec{x}}\, g_j(\vec{x}^*) = 0.$$

  If the constraints (and thus their gradients) are linearly independent, the values λ_j are uniquely determined. This equation can be used to **compensate the gradient** of $f(\vec{x}^*)$ so that it vanishes at $\vec{x}^*$.

General Principle: Lagrange Theory

As a consequence of these insights we obtain the

Method of Lagrange Multipliers:

Given:
- a function $f(\vec{x})$, which is to be optimized,
- k equality constraints $g_j(\vec{x}) = 0$, $1 \leq j \leq k$.

Procedure:

1. Construct the so-called **Lagrange function** by incorporating the equality constraints g_i, $i = 1, \ldots, k$, with (unknown) **Lagrange multipliers** λ_i:

$$L(\vec{x}, \lambda_1, \ldots, \lambda_k) = f(\vec{x}) + \sum_{i=1}^{k} \lambda_i g_i(\vec{x}).$$

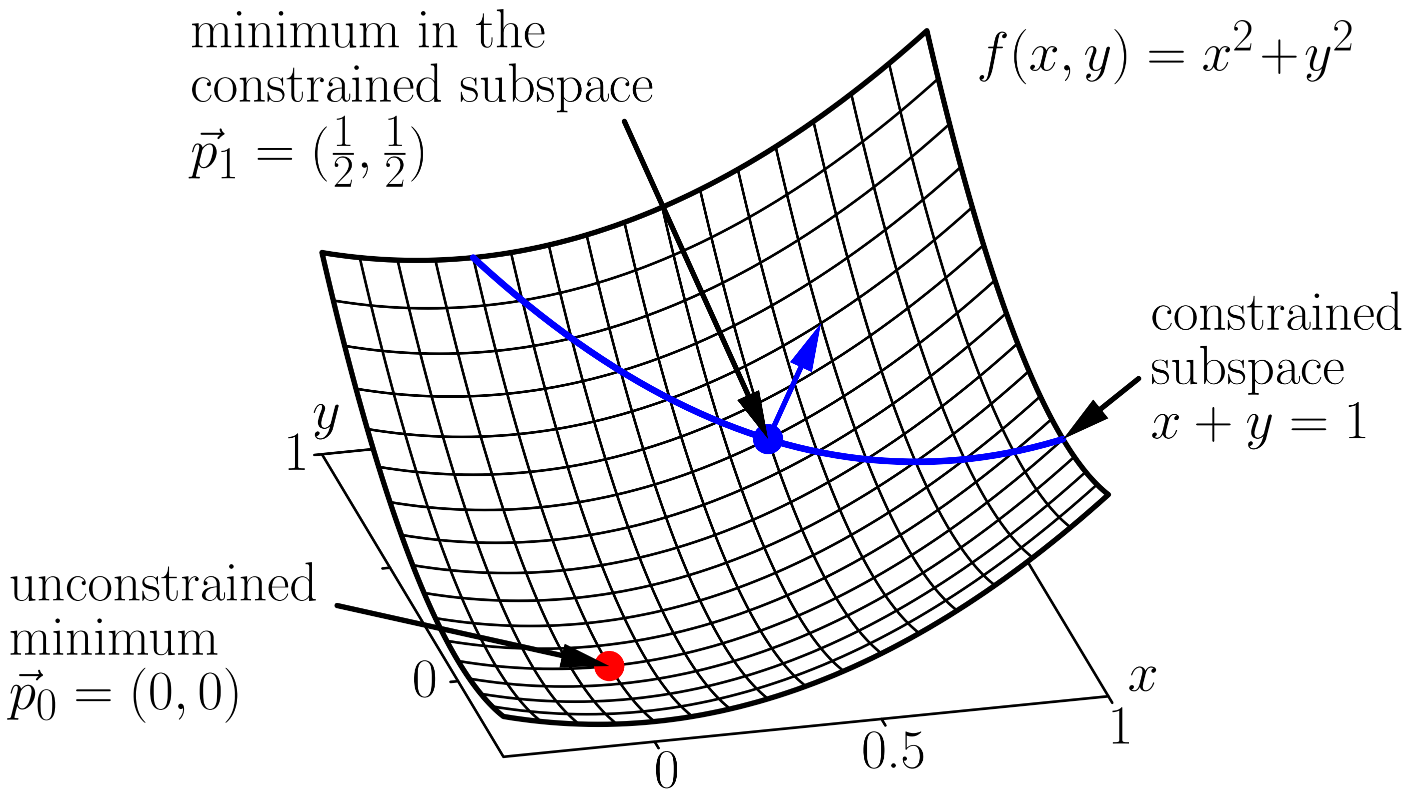
2. Set the partial derivatives of the Lagrange function equal to zero:

$$\frac{\partial L}{\partial x_1} = 0, \quad \ldots, \quad \frac{\partial L}{\partial x_m} = 0, \qquad \frac{\partial L}{\partial \lambda_1} = 0, \quad \ldots, \quad \frac{\partial L}{\partial \lambda_k} = 0.$$

3. (Try to) solve the resulting equation system.

Lagrange Theory: Revisited Example 1

Example task: Minimize $f(x, y) = x^2 + y^2$ subject to $x + y = 1$.

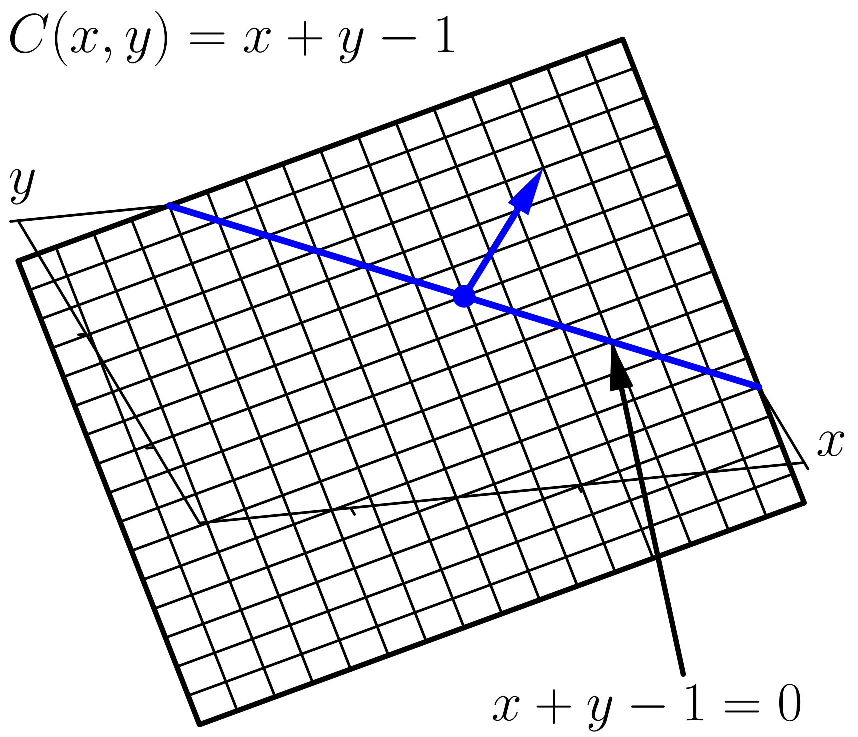


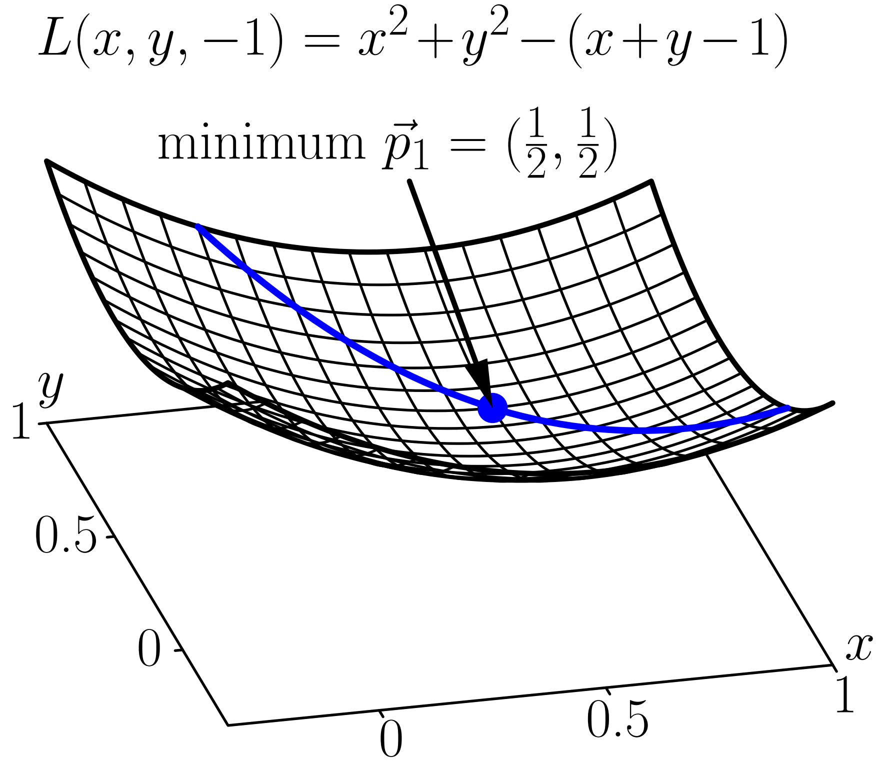
minimum in the constrained subspace $\vec{p}_1 = (\frac{1}{2}, \frac{1}{2})$
$f(x, y) = x^2 + y^2$
constrained subspace $x + y = 1$
unconstrained minimum $\vec{p}_0 = (0, 0)$
y
1
0
x
0
0.5
1

The unconstrained minimum is not in the constrained subspace, and at the minimum in the constrained subspace the gradient does not vanish.

Lagrange Theory: Revisited Example 1

Example task: Minimize $f(x, y) = x^2 + y^2$ subject to $x + y = 1$.





The gradient of the constraint is perpendicular to the constrained subspace.
The (unconstrained) minimum of the Lagrange function $L(x, y, \lambda)$
is the minimum of the objective function $f(x, y)$ in the constrained subspace.

Lagrange Theory: Example 2

Example task: Find the side lengths x, y, z of a box with maximum volume for a given area S of the surface.

Formally: Maximize $f(x, y, z) = xyz$
subject to $2xy + 2xz + 2yz = S$.

Solution procedure:

1. The constraint is $C(x, y, z) = 2xy + 2xz + 2yz - S = 0$.
2. The Lagrange function is
$$L(x, y, z, \lambda) = xyz + \lambda(2xy + 2xz + 2yz - S).$$
3. Taking the partial derivatives yields (in addition to the constraint):
$$\frac{\partial L}{\partial x} = yz+2\lambda(y+z) = 0, \quad \frac{\partial L}{\partial y} = xz+2\lambda(x+z) = 0, \quad \frac{\partial L}{\partial y} = xy+2\lambda(x+y) = 0.$$
4. The solution is: $\lambda = -\frac{1}{4}\sqrt{\frac{S}{6}}$, $\quad x = y = z = \sqrt{\frac{S}{6}}$ (i.e., the box is a cube).

Function Optimization: Lagrange Theory

Observations:

- Due to the representation of the gradient of $f(\vec{x})$ at a local optimum $\vec{x}^*$ in the constrained subspace (see above) the gradient of L w.r.t. $\vec{x}$ vanishes at $\vec{x}^*$.

  $\rightarrow$ The standard approach works again!

- If the constraints are satisfied, the additional terms have no influence.

  $\rightarrow$ The original task is not modified (same objective function).

- Taking the partial derivative w.r.t. a Lagrange multiplier reproduces the corresponding equality constraint:

$$\forall j; 1 \leq j \leq k: \qquad \frac{\partial}{\partial \lambda_j} L(\vec{x}, \lambda_1, \ldots, \lambda_k) = C_j(\vec{x}),$$

  $\rightarrow$ Constraints enter the equation system to solve in a natural way.

Remark:

- **Inequality** constraints can be handled with the **Kuhn–Tucker theory**.

Fuzzy Clustering: Alternating Optimization

Objective function: (to be minimized)

$$J(\mathbf{X}, \mathbf{B}, \mathbf{U}) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} d^{\,2}(\vec{x}_j, \vec{\beta}_i)$$

Constraints:

$$\forall i \in \{1, \ldots, c\} : \quad \sum_{j=1}^{n} u_{ij} > 0 \qquad \text{and} \qquad \forall j \in \{1, \ldots, n\} : \quad \sum_{i=1}^{c} u_{ij} = 1.$$

- **Problem:** The objective function J cannot be minimized directly.
- Therefore: **Alternating Optimization**
  - Optimize membership degrees for fixed cluster parameters.
  - Optimize cluster parameters for fixed membership degrees.

    (Update formulae are derived by differentiating the objective function J)
  - Iterate until convergence (checked, e.g., by change of cluster center).

Fuzzy Clustering: Alternating Optimization

First Step: Fix the cluster parameters.

Introduce Lagrange multipliers λ_j, $0 \leq j \leq n$, to incorporate the constraints $\forall j; 1 \leq j \leq n : \sum_{i=1}^{c} u_{ij} = 1$. This yields the Lagrange function (to be minimized)

$$L(\mathbf{X}, \mathbf{B}, \mathbf{U}, \Lambda) = \underbrace{\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w}\, d_{ij}^{2}}_{=J(\mathbf{X},\mathbf{B},\mathbf{U})} + \sum_{j=1}^{n} \lambda_j \left(1 - \sum_{i=1}^{c} u_{ij}\right),$$

A necessary condition for the minimum is that the partial derivatives of the Lagrange function w.r.t. the membership degrees vanish, i.e.,

$$\frac{\partial}{\partial u_{kl}} L(\mathbf{X}, \mathbf{B}, \mathbf{U}, \Lambda) = w\, u_{kl}^{w-1}\, d_{kl}^{2} - \lambda_l \overset{!}{=} 0,$$

which leads to

$$\forall i; 1 \leq i \leq c : \forall j; 1 \leq j \leq n : \qquad u_{ij} = \left(\frac{\lambda_j}{w\, d_{ij}^{2}}\right)^{\frac{1}{w-1}}.$$

Fuzzy Clustering: Alternating Optimization

Summing these equations over the clusters (in order to be able to exploit the corresponding constraints on the membership degrees), we get

$$1 = \sum_{i=1}^{c} u_{ij} = \sum_{i=1}^{c} \left(\frac{\lambda_j}{w\, d_{ij}^2} \right)^{\frac{1}{w-1}}.$$

Consequently the λ_j, $1 \leq j \leq n$, are

$$\lambda_j = \left(\sum_{i=1}^{c} \left(w\, d_{ij}^2 \right)^{\frac{1}{1-w}} \right)^{1-w}.$$

Inserting this into the equation for the membership degrees yields

$$\forall i; 1 \leq i \leq c : \forall j; 1 \leq j \leq n : \qquad u_{ij} = \frac{d_{ij}^{\frac{2}{1-w}}}{\sum_{k=1}^{c} d_{kj}^{\frac{2}{1-w}}}.$$

This update formula results regardless of the distance measure.

Standard Fuzzy Clustering Algorithms

Fuzzy C-Means Algorithm: Euclidean distance

$$d^2_{\text{fcm}}(\vec{x}_j, \vec{\beta}_i) = (\vec{x}_j - \vec{\mu}_i)^\top (\vec{x}_j - \vec{\mu}_i) \quad \text{with} \quad \vec{\beta}_i = (\vec{\mu}_i)$$

Necessary condition for a minimum: gradients w.r.t. cluster centers vanish.

$$\begin{aligned}
\nabla_{\vec{\mu}_k} J_{\text{fcm}}(\mathbf{X}, \mathbf{B}, \mathbf{U}) &= \nabla_{\vec{\mu}_k} \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^w \, (\vec{x}_j - \vec{\mu}_i)^\top (\vec{x}_j - \vec{\mu}_i) \\
&= \sum_{j=1}^{n} u_{kj}^w \, \nabla_{\vec{\mu}_k} (\vec{x}_j - \vec{\mu}_k)^\top (\vec{x}_j - \vec{\mu}_k) \\
&= -2 \sum_{j=1}^{n} u_{kj}^w \, (\vec{x}_j - \vec{\mu}_k) \overset{!}{=} \vec{0}
\end{aligned}$$

Resulting update rule for the cluster centers (**second step** of alt. optimization):

$$\forall i; 1 \le i \le c: \qquad \vec{\mu}_i = \frac{\sum_{j=1}^n u_{ij}^w \vec{x}_j}{\sum_{j=1}^n u_{ij}^w}$$

Standard Fuzzy Clustering Algorithms

Gustafson–Kessel Algorithm: Mahalanobis distance

$$d_{\mathrm{gk}}^2(\vec{x}_j, \vec{\beta}_i) = (\vec{x}_j - \vec{\mu}_i)^\top \mathbf{C}_i^{-1} (\vec{x}_j - \vec{\mu}_i) \quad \text{with} \quad \vec{\beta}_i = (\vec{\mu}_i, \mathbf{\Sigma}_i)$$

Additional constraints: $|\mathbf{C}_i| = 1$ (all cluster have unit size).
These constraints are incorporated again by Lagrange multipliers.

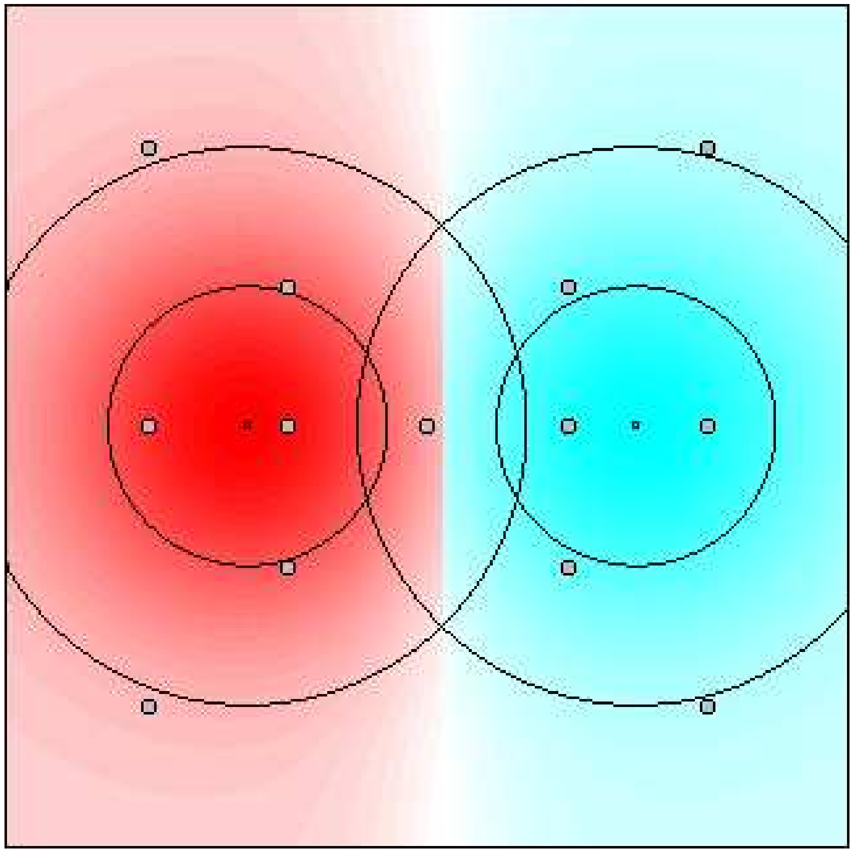
A similar derivation as for the fuzzy c-means algorithm
yields the same update rule for the cluster centers:

$$\forall i; 1 \leq i \leq c: \qquad \vec{\mu}_i = \frac{\sum_{j=1}^n u_{ij}^w \vec{x}_j}{\sum_{j=1}^n u_{ij}^w}$$

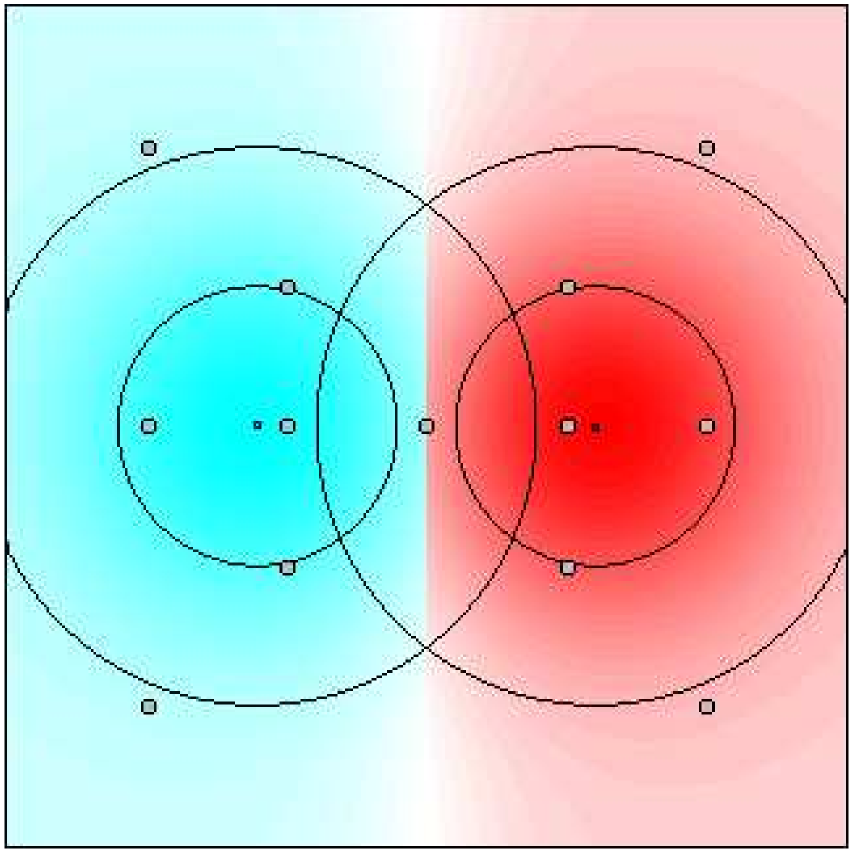
Update rule for the covariance matrices (m is the number of dimensions):

$$\mathbf{C}_i = \frac{1}{\sqrt[m]{|\mathbf{\Sigma}_i|}} \mathbf{\Sigma}_i \qquad \text{where} \qquad \mathbf{\Sigma}_i = \sum_{j=1}^n u_{ij}^w (\vec{x}_j - \vec{\mu}_i)(\vec{x}_j - \vec{\mu}_i)^\top.$$

Fuzzy Clustering: Overlapping Clusters

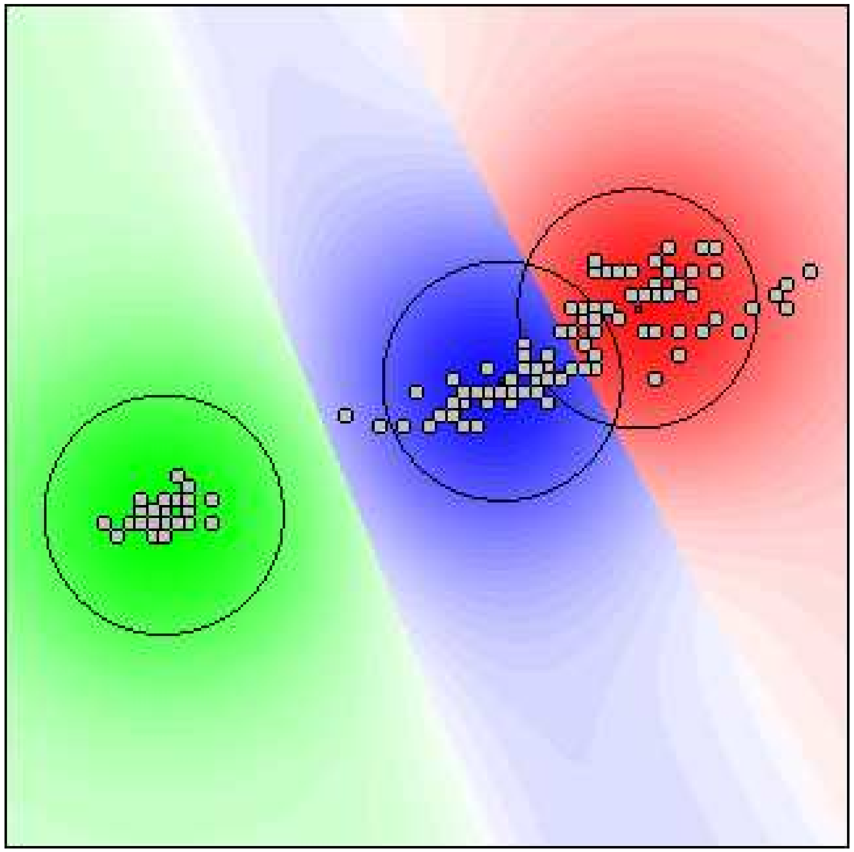


Classical c-Means

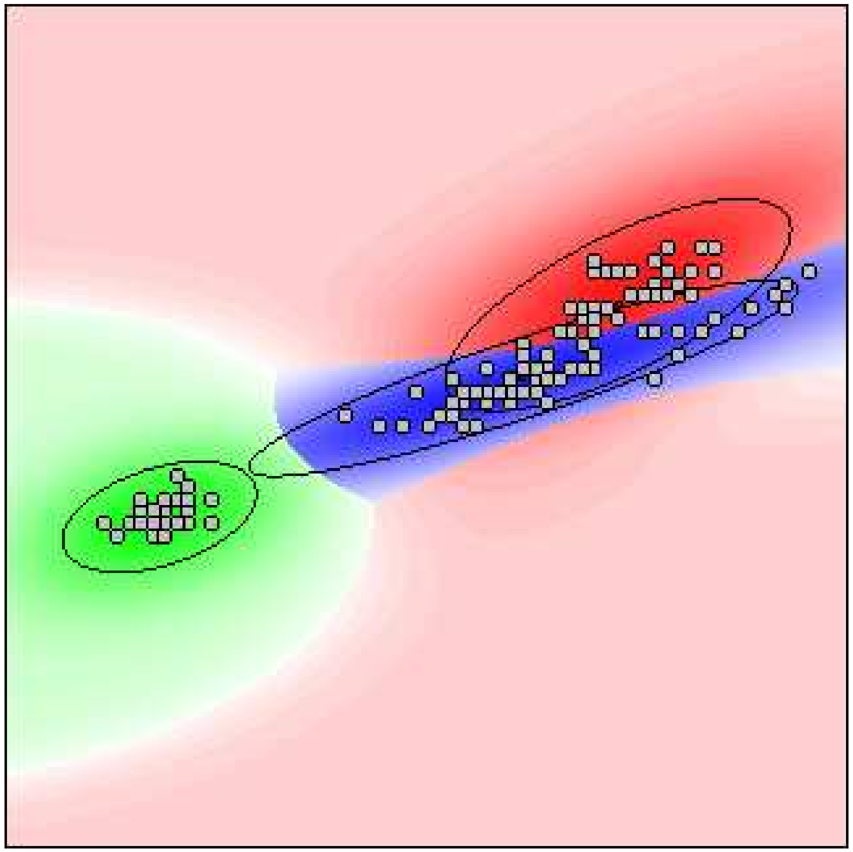


Fuzzy c-Means

Fuzzy Clustering of the Iris Data



Fuzzy c-Means



Gustafson–Kessel

Expectation Maximization: Mixture of Gaussians

- **Assumption:** Data was generated by sampling a set of normal distributions. (The probability density is a mixture of Gaussian distributions.)
- **Formally:** We assume that the probability density can be described as

$$f_{\vec{X}}(\vec{x};\mathbf{C}) = \sum_{y=1}^{c} f_{\vec{X},Y}(\vec{x},y;\mathbf{C}) = \sum_{y=1}^{c} p_Y(y;\mathbf{C}) \cdot f_{\vec{X}|Y}(\vec{x}|y;\mathbf{C}).$$

$\mathbf{C}$	is the set of cluster parameters
$\vec{X}$	is a random vector that has the data space as its domain
Y	is a random variable that has the cluster indices as possible values (i.e., $\mathrm{dom}(\vec{X}) = \mathbb{R}^m$ and $\mathrm{dom}(Y) = \{1,\ldots,c\}$)
$p_Y(y;\mathbf{C})$	is the probability that a data point belongs to (is generated by) the y-th component of the mixture
$f_{\vec{X}|Y}(\vec{x}|y;\mathbf{C})$	is the conditional probability density function of a data point given the cluster (specified by the cluster index y)

Expectation Maximization

- **Basic idea:** Do a maximum likelihood estimation of the cluster parameters.
- **Problem:** The likelihood function,

$$L(\mathbf{X};\mathbf{C}) = \prod_{j=1}^{n} f_{\vec{X}_j}(\vec{x}_j;\mathbf{C}) = \prod_{j=1}^{n} \sum_{y=1}^{c} p_Y(y;\mathbf{C}) \cdot f_{\vec{X}|Y}(\vec{x}_j|y;\mathbf{C}),$$

  is difficult to optimize, even if one takes the natural logarithm (cf. the maximum likelihood estimation of the parameters of a normal distribution), because

$$\ln L(\mathbf{X};\mathbf{C}) = \sum_{j=1}^{n} \ln \sum_{y=1}^{c} p_Y(y;\mathbf{C}) \cdot f_{\vec{X}|Y}(\vec{x}_j|y;\mathbf{C})$$

  contains the natural logarithms of complex sums.

- **Approach:** Assume that there are "hidden" variables Y_j stating the clusters that generated the data points $\vec{x}_j$, so that the sums reduce to one term.
- **Problem:** Since the Y_j are hidden, we do not know their values.

Expectation Maximization

- **Formally:** Maximize the likelihood of the "completed" data set $(\mathbf{X}, \vec{y})$, where $\vec{y} = (y_1, \ldots, y_n)$ combines the values of the variables Y_j. That is,

$$L(\mathbf{X}, \vec{y}; \mathbf{C}) = \prod_{j=1}^{n} f_{\vec{X}_j, Y_j}(\vec{x}_j, y_j; \mathbf{C}) = \prod_{j=1}^{n} p_{Y_j}(y_j; \mathbf{C}) \cdot f_{\vec{X}_j|Y_j}(\vec{x}_j | y_j; \mathbf{C}).$$

- **Problem:** Since the Y_j are hidden, the values y_j are unknown (and thus the factors $p_{Y_j}(y_j; \mathbf{C})$ cannot be computed).

- **Approach to find a solution nevertheless:**
  - See the Y_j as random variables (the values y_j are not fixed) and consider a probability distribution over the possible values.
  - As a consequence $L(\mathbf{X}, \vec{y}; \mathbf{C})$ becomes a random variable, even for a fixed data set $\mathbf{X}$ and fixed cluster parameters $\mathbf{C}$.
  - Try to **maximize the expected value** of $L(\mathbf{X}, \vec{y}; \mathbf{C})$ or $\ln L(\mathbf{X}, \vec{y}; \mathbf{C})$ (hence the name **expectation maximization**).

Expectation Maximization

- **Formally:** Find the cluster parameters as

$$\hat{\mathbf{C}} = \arg\max_{\mathbf{C}} E([\ln] L(\mathbf{X}, \vec{y}; \mathbf{C}) \mid \mathbf{X}; \mathbf{C}),$$

  that is, maximize the expected likelihood

$$E(L(\mathbf{X}, \vec{y}; \mathbf{C}) \mid \mathbf{X}; \mathbf{C}) = \sum_{\vec{y} \in \{1,\ldots,c\}^n} p_{\vec{Y}|\mathcal{X}}(\vec{y}\,|\mathbf{X}; \mathbf{C}) \cdot \prod_{j=1}^{n} f_{\vec{X}_j, Y_j}(\vec{x}_j, y_j; \mathbf{C})$$

  or, alternatively, maximize the expected log-likelihood

$$E(\ln L(\mathbf{X}, \vec{y}; \mathbf{C}) \mid \mathbf{X}; \mathbf{C}) = \sum_{\vec{y} \in \{1,\ldots,c\}^n} p_{\vec{Y}|\mathcal{X}}(\vec{y}\,|\mathbf{X}; \mathbf{C}) \cdot \sum_{j=1}^{n} \ln f_{\vec{X}_j, Y_j}(\vec{x}_j, y_j; \mathbf{C}).$$

- Unfortunately, these functionals are still difficult to optimize directly.

- **Solution:** Use the equation as an iterative scheme, fixing $\mathbf{C}$ in some terms (iteratively compute better approximations, similar to Heron's algorithm).

Excursion: Heron's Algorithm

- **Task:** Find the square root of a given number x, i.e., find $y = \sqrt{x}$.
- **Approach:** Rewrite the defining equation $y^2 = x$ as follows:

$$y^2 = x \quad \Leftrightarrow \quad 2y^2 = y^2 + x \quad \Leftrightarrow \quad y = \frac{1}{2y}(y^2 + x) \quad \Leftrightarrow \quad y = \frac{1}{2}\left(y + \frac{x}{y}\right).$$

- Use the resulting equation as an iteration formula, i.e., compute the sequence

$$y_{k+1} = \frac{1}{2}\left(y_k + \frac{x}{y_k}\right) \qquad \text{with} \qquad y_0 = 1.$$

- It can be shown that $\quad 0 \leq y_k - \sqrt{x} \leq y_{k-1} - y_n \quad$ for $\quad k \geq 2$.
  Therefore this iteration formula provides increasingly better approximations of the square root of x and thus is a safe and simple way to compute it.
  Example $x = 2$: $y_0 = 1$, $y_1 = 1.5$, $y_2 \approx 1.41667$, $y_3 \approx 1.414216$, $y_4 \approx 1.414213$.

- Heron's algorithm converges very quickly and is often used in pocket calculators and microprocessors to implement the square root.

Expectation Maximization

- **Iterative scheme for expectation maximization:**
  Choose some initial set $\mathbf{C}_0$ of cluster parameters and then compute

$$\begin{aligned}
\mathbf{C}_{k+1} &= \arg\max_{\mathbf{C}} E(\ln L(\mathbf{X}, \vec{y}; \mathbf{C}) \mid \mathbf{X}; \mathbf{C}_k) \\
&= \arg\max_{\mathbf{C}} \sum_{\vec{y} \in \{1,\ldots,c\}^n} p_{\vec{Y}|\mathcal{X}}(\vec{y}\,|\mathbf{X}; \mathbf{C}_k) \sum_{j=1}^{n} \ln f_{\vec{X}_j, Y_j}(\vec{x}_j, y_j; \mathbf{C}) \\
&= \arg\max_{\mathbf{C}} \sum_{\vec{y} \in \{1,\ldots,c\}^n} \left(\prod_{l=1}^{n} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l; \mathbf{C}_k) \right) \sum_{j=1}^{n} \ln f_{\vec{X}_j, Y_j}(\vec{x}_j, y_j; \mathbf{C}) \\
&= \arg\max_{\mathbf{C}} \sum_{i=1}^{c} \sum_{j=1}^{n} p_{Y_j|\vec{X}_j}(i|\vec{x}_j; \mathbf{C}_k) \cdot \ln f_{\vec{X}_j, Y_j}(\vec{x}_j, i; \mathbf{C}).
\end{aligned}$$

- It can be shown that each EM iteration increases the likelihood of the data and that the algorithm converges to a local maximum of the likelihood function (i.e., EM is a safe way to maximize the likelihood function).

Expectation Maximization

Justification of the last step on the previous slide:

$$
\begin{aligned}
&\sum_{\vec{y}\in\{1,\ldots,c\}^n} \left(\prod_{l=1}^{n} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k)\right) \sum_{j=1}^{n} \ln f_{\vec{X}_j,Y_j}(\vec{x}_j, y_j;\mathbf{C}) \\
&= \sum_{y_1=1}^{c} \cdots \sum_{y_n=1}^{c} \left(\prod_{l=1}^{n} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k)\right) \sum_{j=1}^{n}\sum_{i=1}^{c} \delta_{i,y_j} \ln f_{\vec{X}_j,Y_j}(\vec{x}_j, i;\mathbf{C}) \\
&= \sum_{i=1}^{c}\sum_{j=1}^{n} \ln f_{\vec{X}_j,Y_j}(\vec{x}_j, i;\mathbf{C}) \sum_{y_1=1}^{c} \cdots \sum_{y_n=1}^{c} \delta_{i,y_j} \prod_{l=1}^{n} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k) \\
&= \sum_{i=1}^{c}\sum_{j=1}^{n} p_{Y_j|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}_k) \cdot \ln f_{\vec{X}_j,Y_j}(\vec{x}_j, i;\mathbf{C}) \\
&\qquad \underbrace{\sum_{y_1=1}^{c} \cdots \sum_{y_{j-1}=1}^{c} \sum_{y_{j+1}=1}^{c} \cdots \sum_{y_n=1}^{c} \prod_{l=1,l\neq j}^{n} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k)}_{=\ \prod_{l=1,l\neq j}^{n} \sum_{y_l=1}^{c} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k)\ =\ \prod_{l=1,l\neq j}^{n} 1\ =\ 1}.
\end{aligned}
$$

Expectation Maximization

- The probabilities $p_{Y_j|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}_k)$ are computed as

$$p_{Y_j|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}_k) = \frac{f_{\vec{X}_j,Y_j}(\vec{x}_j,i;\mathbf{C}_k)}{f_{\vec{X}_j}(\vec{x}_j;\mathbf{C}_k)} = \frac{f_{\vec{X}_j|Y_j}(\vec{x}_j|i;\mathbf{C}_k)\cdot p_{Y_j}(i;\mathbf{C}_k)}{\sum_{l=1}^{c} f_{\vec{X}_j|Y_j}(\vec{x}_j|l;\mathbf{C}_k)\cdot p_{Y_j}(l;\mathbf{C}_k)},$$

  that is, as the relative probability densities of the different clusters (as specified by the cluster parameters) at the location of the data points $\vec{x}_j$.

- The $p_{Y_j|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}_k)$ are the posterior probabilities of the clusters given the data point $\vec{x}_j$ and a set of cluster parameters $\mathbf{C}_k$.

- They can be seen as **case weights** of a "completed" data set:
  - Split each data point $\vec{x}_j$ into c data points $(\vec{x}_j, i)$, $i = 1, \dots, c$.
  - Distribute the unit weight of the data point $\vec{x}_j$ according to the above probabilities, i.e., assign to $(\vec{x}_j, i)$ the weight $p_{Y_j|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}_k)$, $i = 1, \dots, c$.

Expectation Maximization: Cookbook Recipe

Core Iteration Formula

$$\mathbf{C}_{k+1} = \arg\max_{\mathbf{C}} \sum_{i=1}^{c} \sum_{j=1}^{n} p_{Y_j|\vec{X}_j}(i|\vec{x}_j; \mathbf{C}_k) \cdot \ln f_{\vec{X}_j, Y_j}(\vec{x}_j, i; \mathbf{C})$$

Expectation Step

- For all data points $\vec{x}_j$:
  Compute for each normal distribution the probability $p_{Y_j|\vec{X}_j}(i|\vec{x}_j; \mathbf{C}_k)$
  that the data point was generated from it
  (ratio of probability densities at the location of the data point).
  $\rightarrow$ "weight" of the data point for the estimation.

Maximization Step

- For all normal distributions:
  Estimate the parameters by standard maximum likelihood estimation
  using the probabilities ("weights") assigned to the data points
  w.r.t. the distribution in the expectation step.

Expectation Maximization: Mixture of Gaussians

Expectation Step: Use Bayes' rule to compute

$$p_{C|\vec{X}}(i|\vec{x};\mathbf{C}) = \frac{p_C(i;\mathbf{c}_i)\cdot f_{\vec{X}|C}(\vec{x}|i;\mathbf{c}_i)}{f_{\vec{X}}(\vec{x};\mathbf{C})} = \frac{p_C(i;\mathbf{c}_i)\cdot f_{\vec{X}|C}(\vec{x}|i;\mathbf{c}_i)}{\sum_{k=1}^{c} p_C(k;\mathbf{c}_k)\cdot f_{\vec{X}|C}(\vec{x}|k;\mathbf{c}_k)}.$$

$\rightarrow$ "weight" of the data point $\vec{x}$ for the estimation.

Maximization Step: Use maximum likelihood estimation to compute

$$\varrho_i^{(t+1)} = \frac{1}{n}\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}^{(t)}), \qquad \vec{\mu}_i^{(t+1)} = \frac{\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}^{(t)})\cdot\vec{x}_j}{\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}^{(t)})},$$

and $$\mathbf{\Sigma}_i^{(t+1)} = \frac{\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}^{(t)})\cdot\left(\vec{x}_j-\vec{\mu}_i^{(t+1)}\right)\left(\vec{x}_j-\vec{\mu}_i^{(t+1)}\right)^\top}{\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}^{(t)})}$$

Iterate until convergence (checked, e.g., by change of mean vector).

Expectation Maximization: Technical Problems

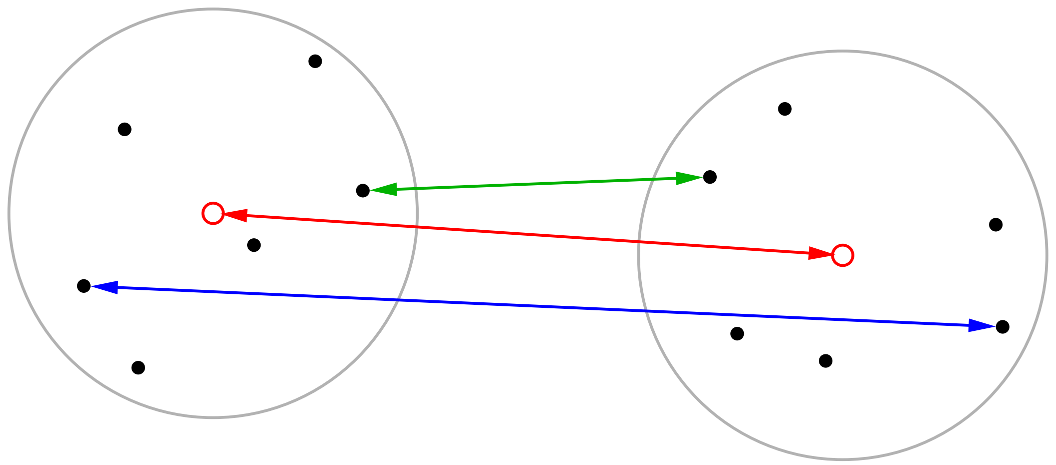
- If a fully general mixture of Gaussian distributions is used, the likelihood function is truly optimized if
  - all normal distributions except one are contracted to single data points and
  - the remaining normal distribution is the maximum likelihood estimate for the remaining data points.
- This undesired result is rare, because the algorithm gets stuck in a local optimum.
- Nevertheless it is recommended to take countermeasures, which consist mainly in reducing the degrees of freedom, like
  - Fix the determinants of the covariance matrices to equal values.
  - Use a diagonal instead of a general covariance matrix.
  - Use an isotropic variance instead of a covariance matrix.
  - Fix the prior probabilities of the clusters to equal values.

Hierarchical Agglomerative Clustering

- Start with every data point in its own cluster.
  (i.e., start with so-called **singletons**: single element clusters)
- In each step merge those two clusters that are closest to each other.
- Keep on merging clusters until all data points are contained in one cluster.
- The result is a hierarchy of clusters that can be visualized in a tree structure
  (a so-called **dendrogram** — from the Greek $\delta\acute{\varepsilon}\nu\tau\varrho\omega\nu$ (dendron): tree)
- **Measuring the Distances**
  - The distance between singletons is simply the distance between the (single) data points contained in them.
  - However: How do we compute the distance between clusters that contain more than one data point?

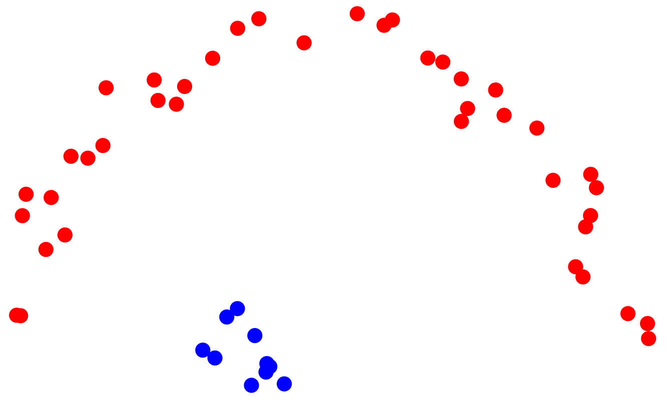
Measuring the Distance between Clusters

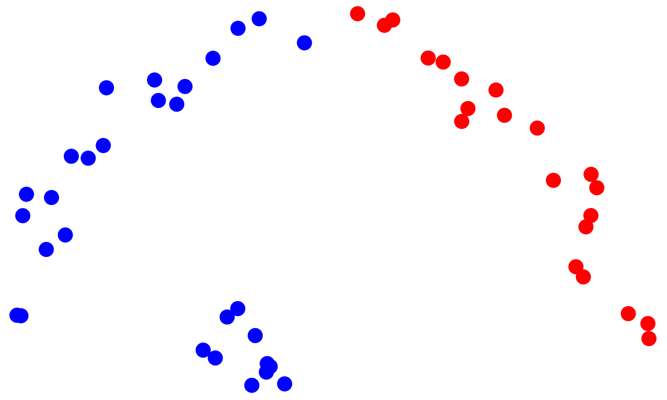
- **Centroid** (red)
  Distance between the centroids (mean value vectors) of the two clusters.
- **Average Linkage**
  Average distance between two points of the two clusters.
- **Single Linkage** (green)
  Distance between the two closest points of the two clusters.
- **Complete Linkage** (blue)
  Distance between the two farthest points of the two clusters.



Measuring the Distance between Clusters

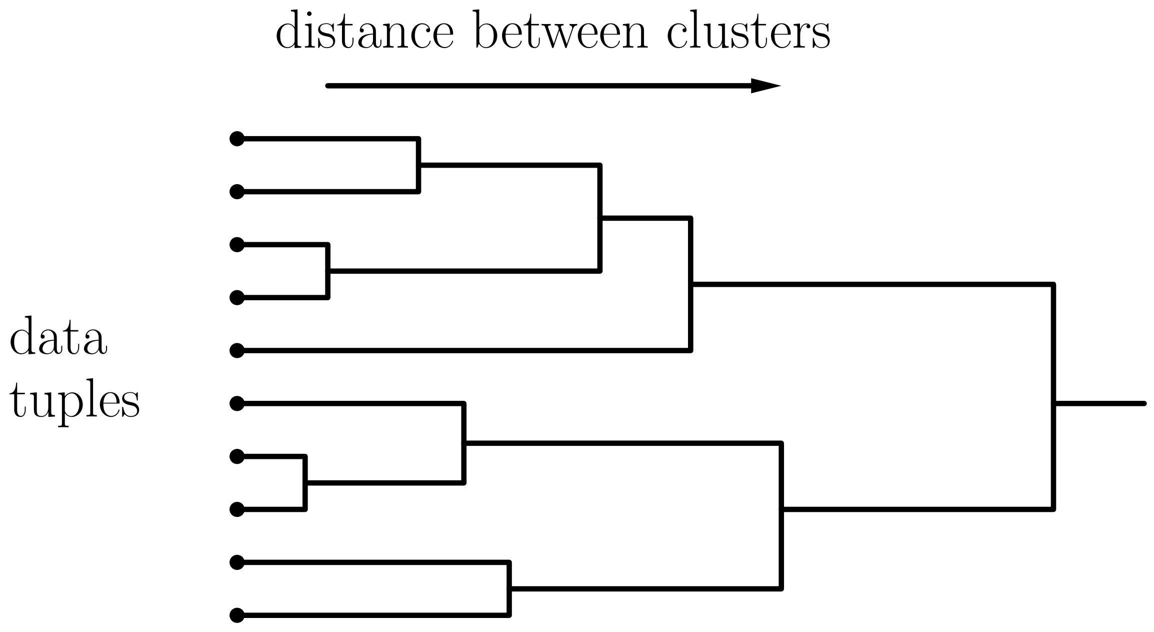
- Single linkage can "follow chains" in the data (may be desirable in certain applications).
- Complete linkage leads to very compact clusters.
- Average linkage also tends clearly towards compact clusters.


Single Linkage


Complete Linkage

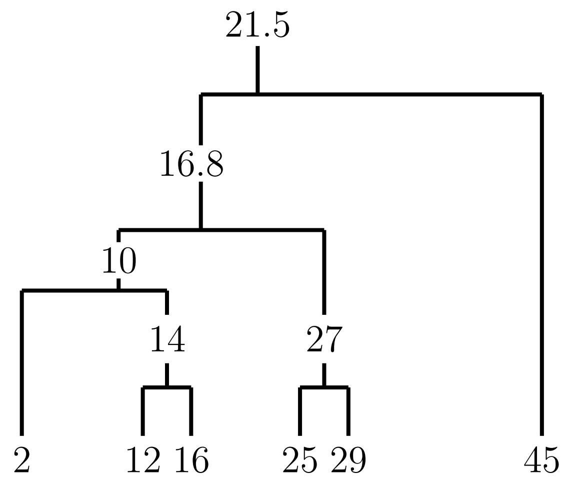
Dendrograms

- The cluster merging process arranges the data points in a binary tree.
- Draw the data tuples at the bottom or on the left
(equally spaced if they are multi-dimensional).
- Draw a connection between clusters that are merged, with the distance to the data points representing the distance between the clusters.

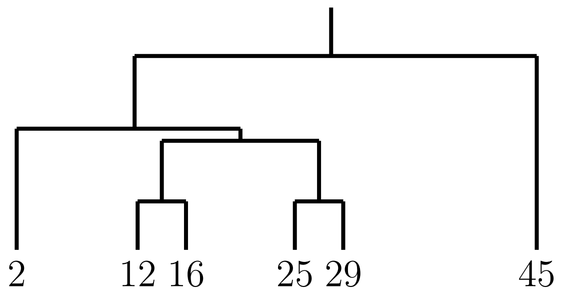


Hierarchical Agglomerative Clustering

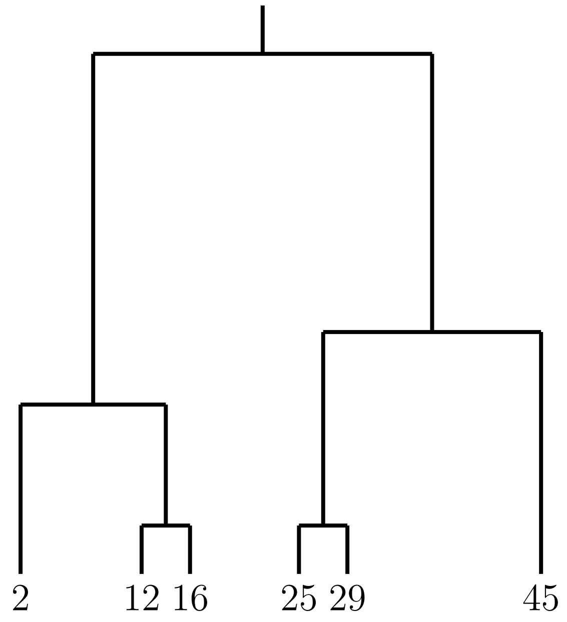
- Example: Clustering of the 1-dimensional data set $\{2, 12, 16, 25, 29, 45\}$.
- All three approaches to measure the distance between clusters lead to different dendrograms.



Centroid



Single Linkage



Complete Linkage

Implementation Aspects

- Hierarchical agglomerative clustering can be implemented by processing the matrix $\mathbf{D} = (d_{ij})_{1 \le i,j \le n}$ containing the pairwise distances of the data points. (The data points themselves are actually not needed.)
- In each step the rows and columns corresponding to the two clusters that are closest to each other are deleted.
- A new row and column corresponding to the cluster formed by merging these clusters is added to the matrix.
- The elements of this new row/column are computed according to

$$\forall k: \qquad d_{k*} = d_{*k} = \alpha_i d_{ik} + \alpha_j d_{jk} + \beta d_{ij} + \gamma |d_{ik} - d_{jk}|$$

i, j	indices of the two clusters that are merged
k	indices of the old clusters that are *not* merged
$*$	index of the new cluster (result of merger)
$\alpha_i, \alpha_j, \beta, \gamma$	parameters specifying the method (single linkage etc.)

Implementation Aspects

- The parameters defining the different methods are (n_i, n_j, n_k are the numbers of data points in the clusters):

method	α_i	α_j	β	γ
centroid method	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	$-\frac{n_i n_j}{n_i+n_j}$	0
median method	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{4}$	0
single linkage	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
complete linkage	$\frac{1}{2}$	$\frac{1}{2}$	0	$+\frac{1}{2}$
average linkage	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	0	0
Ward's method	$\frac{n_i+n_k}{n_i+n_j+n_k}$	$\frac{n_j+n_k}{n_i+n_j+n_k}$	0	0

Choosing the Clusters

- **Simplest Approach:**
  - Specify a minimum desired distance between clusters.
  - Stop merging clusters if the closest two clusters are farther apart than this distance.
- **Visual Approach:**
  - Merge clusters until all data points are combined into one cluster.
  - Draw the dendrogram and find a good cut level.
  - Advantage: Cut need not be strictly horizontal.
- **More Sophisticated Approaches:**
  - Analyze the sequence of distances in the merging process.
  - Try to find a step in which the distance between the two clusters merged is considerably larger than the distance of the previous step.
  - Several heuristic criteria exist for this step selection.

Summary Clustering

- **Prototype-based Clustering**
  - Alternating adaptation of data point assignment and cluster parameters.
  - Online or batch adaptation of the cluster center.
  - Crisp or fuzzy/probabilistic assignment of a datum to a cluster.
  - Local minima can pose a problem.
  - Fuzzy/probabilistic approaches are usually more robust.

- **Hierarchical Agglomerative Clustering**
  - Start with singletons (one element clusters).
  - Always merge those clusters that are closest.
  - Different ways to measure the distance of clusters.
  - Cluster hierarchy can be depicted as a dendrogram.

Association Rules and Frequent Item Sets

Frequent Item Set Mining: Motivation

- Frequent Item Set Mining is a method for **market basket analysis**.
- It aims at finding regularities in the shopping behavior of customers of supermarkets, mail-order companies, on-line shops etc.
- More specifically:
  Find sets of products that are frequently bought together.
- Possible applications of found frequent item sets:
  - Improve arrangement of products in shelves, on a catalog's pages.
  - Support cross-selling (suggestion of other products), product bundling.
  - Fraud detection, technical dependence analysis.
- Often found patterns are expressed as **association rules**, for example:

  If a customer buys **bread** and **wine**,
  then she/he will probably also buy **cheese**.

Frequent Item Set Mining: Basic Notions

- Let $A = \{a_1, \ldots, a_m\}$ be a set of **items**.

  Items may be products, special equipment items, service options etc.

- Any subset $I \subseteq A$ is called an **item set**.

  An item set may be any set of products that can be bought (together).

- Let $T = (t_1, \ldots, t_n)$ with $\forall i, 1 \leq i \leq n : t_i \subseteq A$
  be a vector of **transactions** over A.

  Each transaction is an item set, but some item sets may not appear in T.

  Transactions need not be pairwise different: it may be $t_i = t_k$ for $i \neq k$.

  T may also be defined as a *bag* or *multiset* of transactions.

  The set A may not be explicitly given, but only implicitely as $A = \bigcup_{i=1}^{n} t_i$.

  A vector of transactions can list, for example, the sets of products bought
  by the customers of a supermarket in a given period of time.

Frequent Item Set Mining: Basic Notions

Let $I \subseteq A$ be an item set and T a vector of transactions over A.

- A transaction $t \in T$ **covers** the item set I or
  the item set I is **contained in** a transaction $t \in T$ iff $I \subseteq t$.
- The set $K_T(I) = \{k \in \{1, \ldots, n\} \mid I \subseteq t_k\}$ is called the **cover** of I w.r.t. T.

  The cover of an item set is the index set of the transactions that cover it.

  It may also be defined as a vector of all transactions that cover it
  (which, however, is complicated to write in formally correct way).
- The value $s_T(I) = |K_T(I)|$ is called the **(absolute) support** of I w.r.t. T.

  The value $\sigma_T(I) = \frac{1}{n}|K_T(I)|$ is called the **relative support** of I w.r.t. T.

  The support of I is the number or fraction of transactions that contain it.

  Sometimes $\sigma_T(I)$ is also called the *(relative) frequency* of I w.r.t. T.

Frequent Item Set Mining: Formal Definition

Given:

- a set $A = \{a_1, \ldots, a_m\}$ of items,
- a vector $T = (t_1, \ldots, t_n)$ of transactions over A,
- a number $s_{\min} \in \mathbb{N}$, $0 < s_{\min} \leq n$ or (equivalently)
  a number $\sigma_{\min} \in \mathbb{R}$, $0 < \sigma_{\min} \leq 1$, the **minimum support**.

Desired:

- the set of **frequent item sets**, that is,
  the set $F_T(s_{\min}) = \{I \subseteq A \mid s_T(I) \geq s_{\min}\}$ or (equivalently)
  the set $\Phi_T(\sigma_{\min}) = \{I \subseteq A \mid \sigma_T(I) \geq \sigma_{\min}\}$.

Note that with the relations $s_{\min} = \lceil n\sigma_{\min} \rceil$ and $\sigma_{\min} = \frac{1}{n} s_{\min}$
the two versions can easily be transformed into each other.

Frequent Item Sets: Example

transaction vector

1:	$\{a, d, e\}$
2:	$\{b, c, d\}$
3:	$\{a, c, e\}$
4:	$\{a, c, d, e\}$
5:	$\{a, e\}$
6:	$\{a, c, d\}$
7:	$\{b, c\}$
8:	$\{a, c, d, e\}$
9:	$\{c, b, e\}$
10:	$\{a, d, e\}$

frequent item sets

0 items	1 item	2 items	3 items
$\emptyset$: 100%	$\{a\}$: 70%	$\{a, c\}$: 40%	$\{a, c, d\}$: 30%
	$\{b\}$: 30%	$\{a, d\}$: 50%	$\{a, c, e\}$: 30%
	$\{c\}$: 70%	$\{a, e\}$: 60%	$\{a, d, e\}$: 40%
	$\{d\}$: 60%	$\{b, c\}$: 30%	
	$\{e\}$: 70%	$\{c, d\}$: 40%	
		$\{c, e\}$: 40%	
		$\{d, e\}$: 40%	

- The minimum support is $s_{\min} = 3$ or $\sigma_{\min} = 0.3 = 30\%$ in this example.
- There are $2^5 = 32$ possible item sets over $A = \{a, b, c, d, e\}$.
- There are 16 frequent item sets (but only 10 transactions).

Properties of the Support of an Item Set

- A **brute force approach** that enumerates all possible item sets, determines their support, and discards infrequent item sets is usually **infeasible**:

  The number of possible item sets grows exponentially with the number of items.

  A typical supermarket has thousands of different products.

- **Idea:** Consider the properties of the support, in particular:

$$\forall I : \forall J \supseteq I : \quad K_T(J) \subseteq K_T(I).$$

  This property holds, since $\forall t : \forall I : \forall J \supseteq I : \quad J \subseteq t \to I \subseteq t$.

  Each additional item is another condition a transaction has to satisfy. Transactions that do not satisfy this condition are removed from the cover.

- It follows:

$$\forall I : \forall J \supseteq I : \quad s_T(I) \geq s_T(J).$$

  That is: **If an item set is extended, its support cannot increase.**

  One also says that support is **anti-monotone** or **downward closed**.

Properties of the Support of an Item Set

- From $\forall I : \forall J \supseteq I : s_T(I) \geq s_T(J)$ it follows

  $$\forall s_{\min} : \forall I : \forall J \supseteq I : \quad s_T(I) < s_{\min} \;\rightarrow\; s_T(J) < s_{\min}.$$

  That is: **No superset of an infrequent item set can be frequent.**

- This property is often referred to as the **Apriori Property**.

  Rationale: Sometimes we can know *a priori*, that is, before checking its support by accessing the given transaction vector, that an item set cannot be frequent.

- Of course, the contraposition of this implication also holds:

  $$\forall s_{\min} : \forall I : \forall J \subseteq I : \quad s_T(I) \geq s_{\min} \;\rightarrow\; s_T(J) \geq s_{\min}.$$

  That is: **All subsets of a frequent item set are frequent.**

- This suggests a compressed representation of the set of frequent item sets.

Maximal Item Sets

- Consider the set of **maximal (frequent) item sets**:

$$M_T(s_{\min}) = \{I \subseteq A \mid s_T(I) \geq s_{\min} \wedge \forall J \supset I : s_T(J) < s_{\min}\}.$$

That is: An item set is maximal if it is frequent,
but none of its proper supersets is frequent.

- Since with this definition we know that

$$\forall s_{\min} : \forall I : \quad I \in M_T(s_{\min}) \ \vee \ \exists J \supset I : s_T(J) \geq s_{\min}$$

it follows (can easily be proven by successively extending the item set I)

$$\forall s_{\min} : \forall I : \quad I \in F_T(s_{\min}) \ \rightarrow \ \exists J \in M_T(s_{\min}) : I \subseteq J.$$

That is: **Every frequent item set has a maximal superset.**

- Therefore:

$$\forall s_{\min} : \quad F_T(s_{\min}) = \bigcup_{I \in M_T(s_{\min})} 2^I$$

Maximal Frequent Item Sets: Example

transaction vector

1:	$\{a,d,e\}$
2:	$\{b,c,d\}$
3:	$\{a,c,e\}$
4:	$\{a,c,d,e\}$
5:	$\{a,e\}$
6:	$\{a,c,d\}$
7:	$\{b,c\}$
8:	$\{a,c,d,e\}$
9:	$\{c,b,e\}$
10:	$\{a,d,e\}$

frequent item sets

0 items	1 item	2 items	3 items
$\emptyset$: 100%	$\{a\}$: 70%	$\{a,c\}$: 40%	$\{a,c,d\}$: 30%
	$\{b\}$: 30%	$\{a,d\}$: 50%	$\{a,c,e\}$: 30%
	$\{c\}$: 70%	$\{a,e\}$: 60%	$\{a,d,e\}$: 40%
	$\{d\}$: 60%	$\{b,c\}$: 30%	
	$\{e\}$: 70%	$\{c,d\}$: 40%	
		$\{c,e\}$: 40%	
		$\{d,e\}$: 40%	

- The maximal item sets are:

$$\{b,c\}, \quad \{a,c,d\}, \quad \{a,c,e\}, \quad \{a,d,e\}.$$

- Every frequent item set is a subset of at least one of these sets.

Limits of Maximal Item Sets

- The set of maximal item sets captures the set of all frequent item sets, but then we know only the support of the maximal item sets.

- About the support of a non-maximal frequent item set we only know:

$$\forall s_{\min} : \forall I \in F_T(s_{\min}) - M_T(s_{\min}) : \quad s_T(I) \geq \max_{J \in M_T(s_{\min}), J \supset I} s_T(J).$$

  This relation follows immediately from $\forall I : \forall J \supseteq I : s_T(I) \geq s_T(J)$, that is, an item set cannot have a lower support than any of its supersets.

- Note that we have generally

$$\forall s_{\min} : \forall I \in F_T(s_{\min}) : \quad s_T(I) \geq \max_{J \in M_T(s_{\min}), J \supseteq I} s_T(J).$$

- **Question:** Can we find a subset of the set of all frequent item sets, which also preserves knowledge of all support values?

Closed Item Sets

- Consider the set of **closed (frequent) item sets**:

$$C_T(s_{\min}) = \{I \subseteq A \mid s_T(I) \geq s_{\min} \wedge \forall J \supset I : s_T(J) < s_T(I)\}.$$

  That is: An item set is closed if it is frequent,
  but none of its proper supersets has the same support.

- Since with this definition we know that

$$\forall s_{\min} : \forall I : \quad I \in C_T(s_{\min}) \;\vee\; \exists J \supset I : s_T(J) = s_T(I)$$

  it follows (can easily be proven by successively extending the item set I)

$$\forall s_{\min} : \forall I : \quad I \in F_T(s_{\min}) \;\rightarrow\; \exists J \in C_T(s_{\min}) : I \subseteq J.$$

  That is: **Every frequent item set has a closed superset.**

- Therefore:

$$\forall s_{\min} : \quad F_T(s_{\min}) = \bigcup_{I \in C_T(s_{\min})} 2^I$$

Closed Item Sets

- However, not only has every frequent item set a closed superset, but it has a **closed superset with the same support**:

$$\forall s_{\min} : \forall I : \quad I \in F_T(s_{\min}) \;\rightarrow\; \exists J \supseteq I : J \in C_T(s_{\min}) \;\wedge\; s_T(J) = s_T(I).$$

  (Proof: see the considerations on the next slide)

- The set of all closed item sets preserves knowledge of all support values:

$$\forall s_{\min} : \forall I \in F_T(s_{\min}) : \quad s_T(I) = \max_{J \in C_T(s_{\min}), J \supseteq I} s_T(J).$$

- Note that the weaker statement

$$\forall s_{\min} : \forall I \in F_T(s_{\min}) : \quad s_T(I) \geq \max_{J \in C_T(s_{\min}), J \supseteq I} s_T(J)$$

  follows immediately from $\forall I : \forall J \supseteq I : s_T(I) \geq s_T(J)$, that is, an item set cannot have a lower support than any of its supersets.

Closed Item Sets

- Alternative characterization of closed item sets:

$$I \text{ closed} \quad \Leftrightarrow \quad s_T(I) \geq s_{\min} \quad \wedge \quad I = \bigcap_{k \in K_T(I)} t_k.$$

  Reminder: $K_T(I) = \{k \in \{1, \ldots, n\} \mid I \subseteq t_k\}$ is the *cover* of I w.r.t. T.

- This is derived as follows: since $\forall k \in K_T(I) : I \subseteq t_k$, it is obvious that

$$\forall s_{\min} : \forall I \in F_T(s_{\min}) : \quad I \subseteq \bigcap_{k \in K_T(I)} t_k,$$

  If $I \subset \bigcap_{k \in K_T(I)} t_k$, it is not closed, since $\bigcap_{k \in K_T(I)} t_k$ has the same support. On the other hand, no superset of $\bigcap_{k \in K_T(I)} t_k$ has the cover $K_T(I)$.

- Note that the above characterization allows us to construct the (uniquely determined) closed superset of a frequent item set that has the same support.

Closed Frequent Item Sets: Example

transaction vector

1:	$\{a, d, e\}$
2:	$\{b, c, d\}$
3:	$\{a, c, e\}$
4:	$\{a, c, d, e\}$
5:	$\{a, e\}$
6:	$\{a, c, d\}$
7:	$\{b, c\}$
8:	$\{a, c, d, e\}$
9:	$\{c, b, e\}$
10:	$\{a, d, e\}$

frequent item sets

0 items	1 item	2 items	3 items
$\emptyset$: 100%	$\{a\}$: 70%	$\{a, c\}$: 40%	$\{a, c, d\}$: 30%
	$\{b\}$: 30%	$\{a, d\}$: 50%	$\{a, c, e\}$: 30%
	$\{c\}$: 70%	$\{a, e\}$: 60%	$\{a, d, e\}$: 40%
	$\{d\}$: 60%	$\{b, c\}$: 30%	
	$\{e\}$: 70%	$\{c, d\}$: 40%	
		$\{c, e\}$: 40%	
		$\{d, e\}$: 40%	

- All frequent item sets are closed with the exception of $\{b\}$ and $\{d, e\}$.
- $\{b\}$ is a subset of $\{b, c\}$, both have support 30%.
  $\{d, e\}$ is a subset of $\{a, d, e\}$, both have a support of 40%.

Types of Frequent Item Sets

- **Frequent Item Set**
  Any frequent item set (support is higher than the minimal support):
  I frequent $\quad \Leftrightarrow \quad s_T(I) \geq s_{\min}$

- **Closed Item Set**
  A frequent item set is called *closed* if no superset has the same support:
  I closed $\quad \Leftrightarrow \quad s_T(I) \geq s_{\min} \quad \wedge \quad \forall J \supset I : s_T(J) < s_T(I)$

- **Maximal Item Set**
  A frequent item set is called *maximal* if no superset is frequent:
  I maximal $\quad \Leftrightarrow \quad s_T(I) \geq s_{\min} \quad \wedge \quad \forall J \supset I : s_T(J) < s_{\min}$

- Obvious relations between these types of item sets:
  - All maximal and all closed item sets are frequent.
  - All maximal item sets are closed.

Types of Frequent Item Sets: Example

0 items	1 item	2 items	3 items
$\emptyset^{+}$: 100%	$\{a\}^{+}$: 70%	$\{a,c\}^{+}$: 40%	$\{a,c,d\}^{+*}$: 30%
	$\{b\}$: 30%	$\{a,d\}^{+}$: 50%	$\{a,c,e\}^{+*}$: 30%
	$\{c\}^{+}$: 70%	$\{a,e\}^{+}$: 60%	$\{a,d,e\}^{+*}$: 40%
	$\{d\}^{+}$: 60%	$\{b,c\}^{+*}$: 30%	
	$\{e\}^{+}$: 70%	$\{c,d\}^{+}$: 40%	
		$\{c,e\}^{+}$: 40%	
		$\{d,e\}$: 40%	

- **Frequent Item Set**
  Any frequent item set (support is higher than the minimal support).
- **Closed Item Set** (marked with $^{+}$)
  A frequent item set is called *closed* if no superset has the same support.
- **Maximal Item Set** (marked with *)
  A frequent item set is called *maximal* if no superset is frequent.

Searching for Frequent Item Sets

- We know that it suffices to find the closed item sets together with their support.
- The characterization of closed item sets by

$$I \text{ closed} \quad \Leftrightarrow \quad s_T(I) \geq s_{\min} \quad \wedge \quad I = \bigcap_{k \in K_T(I)} t_k$$

  suggests to find them by forming all possible intersections of the transactions and checking their support.

  However, approaches using this idea are not competitive with other methods.

- If the support of all frequent item sets is needed, it can be clumsy and tedious to compute the support of a non-closed frequent item set with

$$\forall s_{\min} : \forall I \in F_T(s_{\min}) - C_T(s_{\min}) : \quad s_T(I) = \max_{J \in C_T(s_{\min}), J \supset I} s_T(J).$$

- In order to find the closed sets one may have to visit many frequent sets anyway.

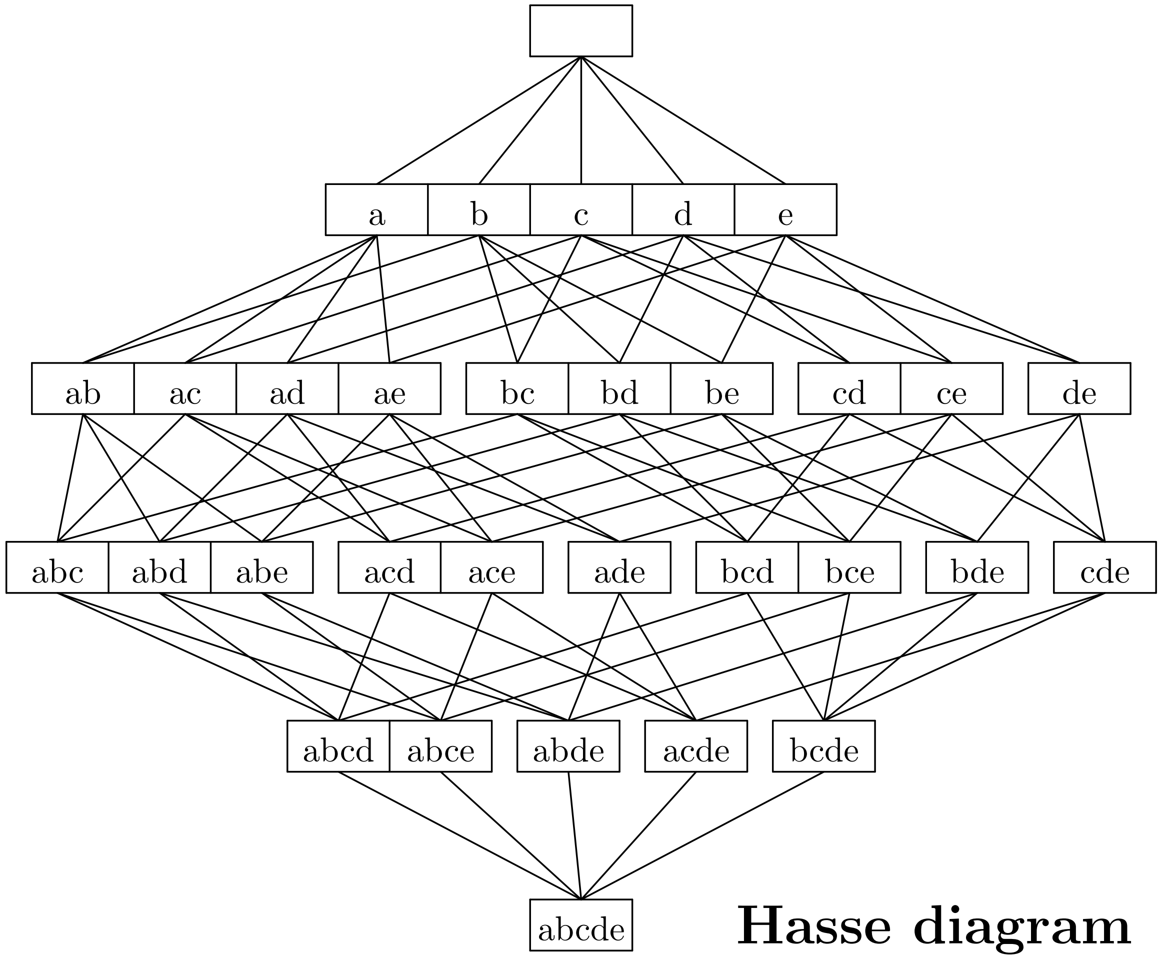
Finding the Frequent Item Sets

Idea: Use the properties of the support to organize the search for all frequent item sets, especially

$$\forall I : \forall J \supset I : \quad s_T(I) < s_{\min} \;\rightarrow\; s_T(J) < s_{\min}.$$

Since these properties relate the support of an item set to the support of its **subsets** and **supersets**, it is reasonable to organize the search based on the **subset lattice** of the set A, the set of all items.

A subset lattice for five items $\{a, b, c, d, e\}$:



Hasse diagram

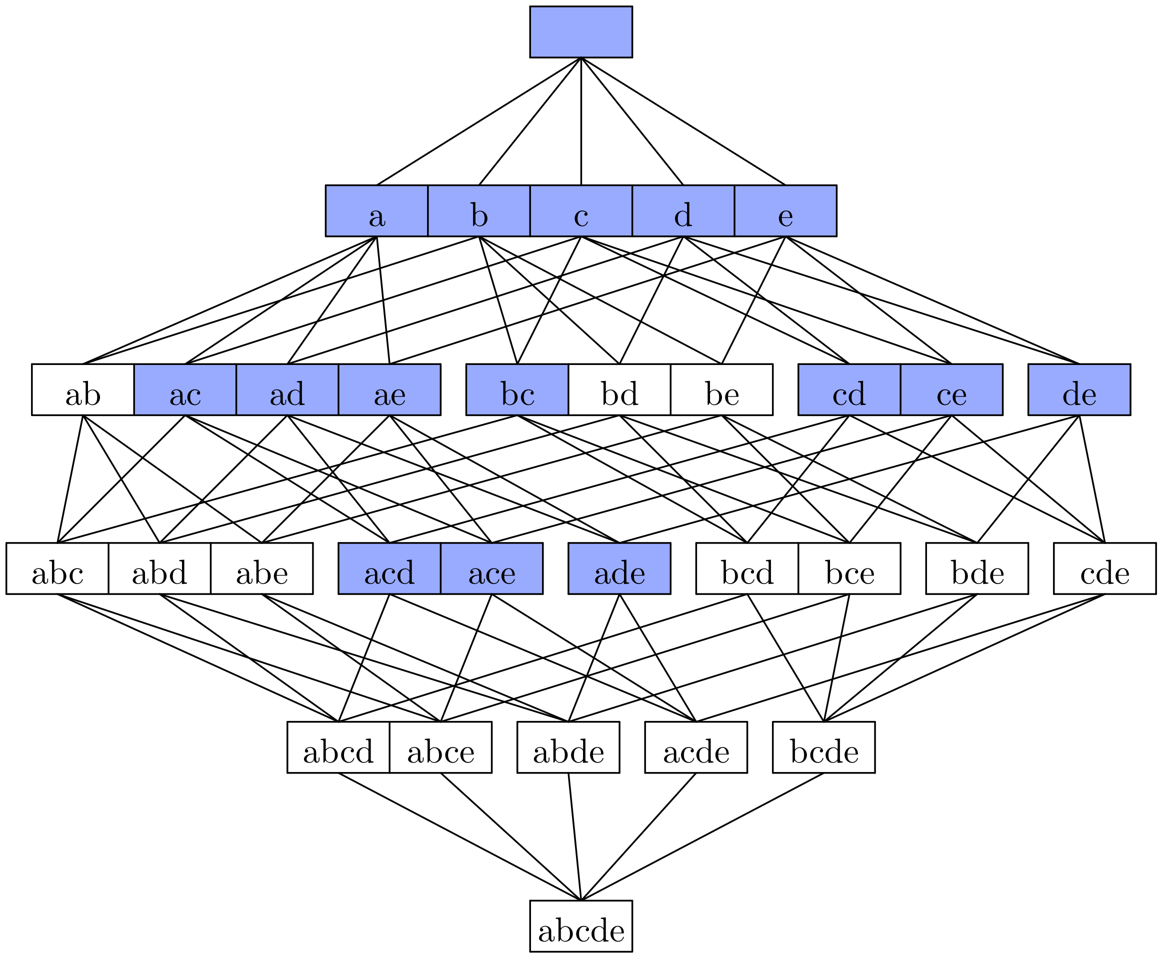
Subset Lattice and Frequent Item Sets

transaction vector

1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$

Blue boxes are frequent item sets, white boxes infrequent item sets.

subset lattice with frequent item sets ($s_{\min} = 3$):



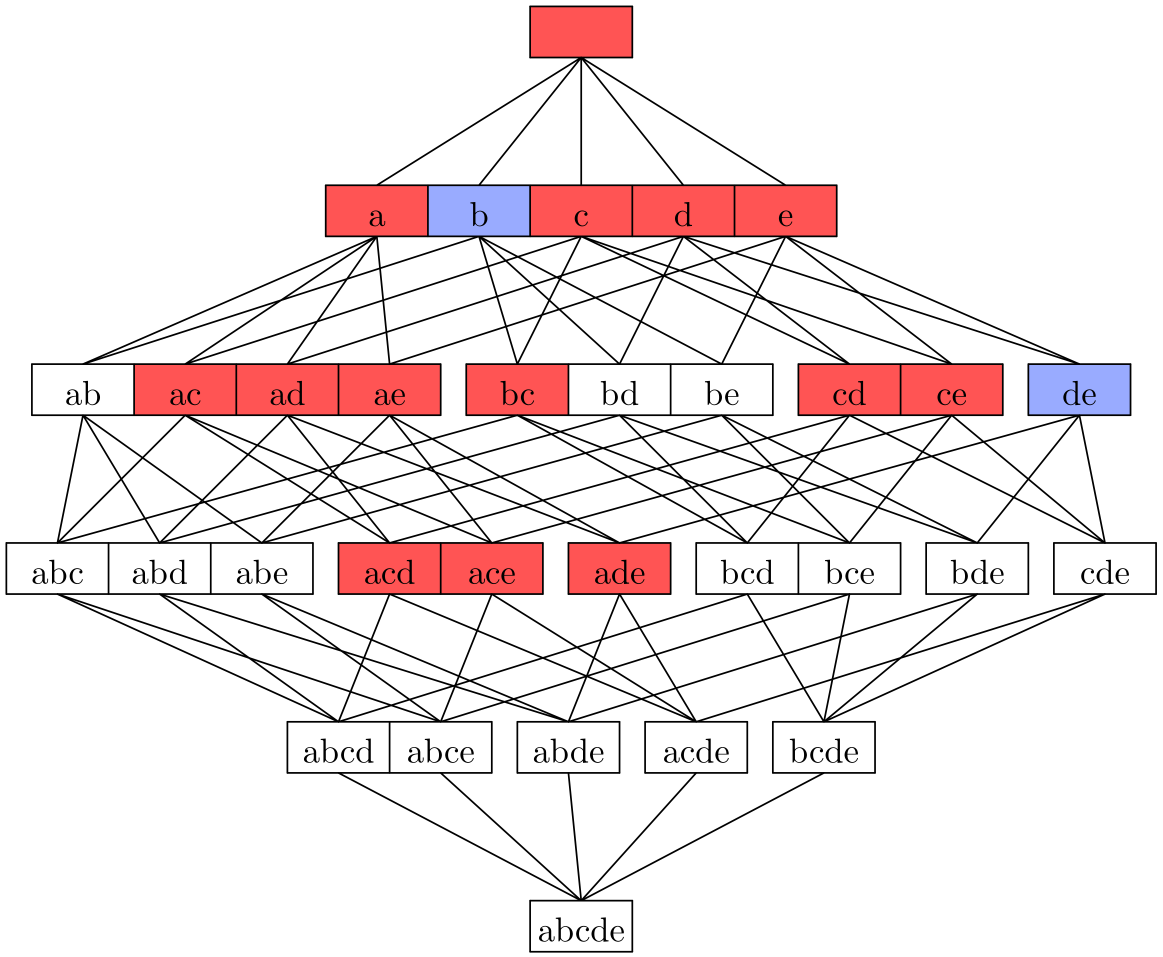
Subset Lattice and Closed Item Sets

transaction vector

1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$

Red boxes are closed item sets, white boxes infrequent item sets.

subset lattice with closed item sets ($s_{\min} = 3$):



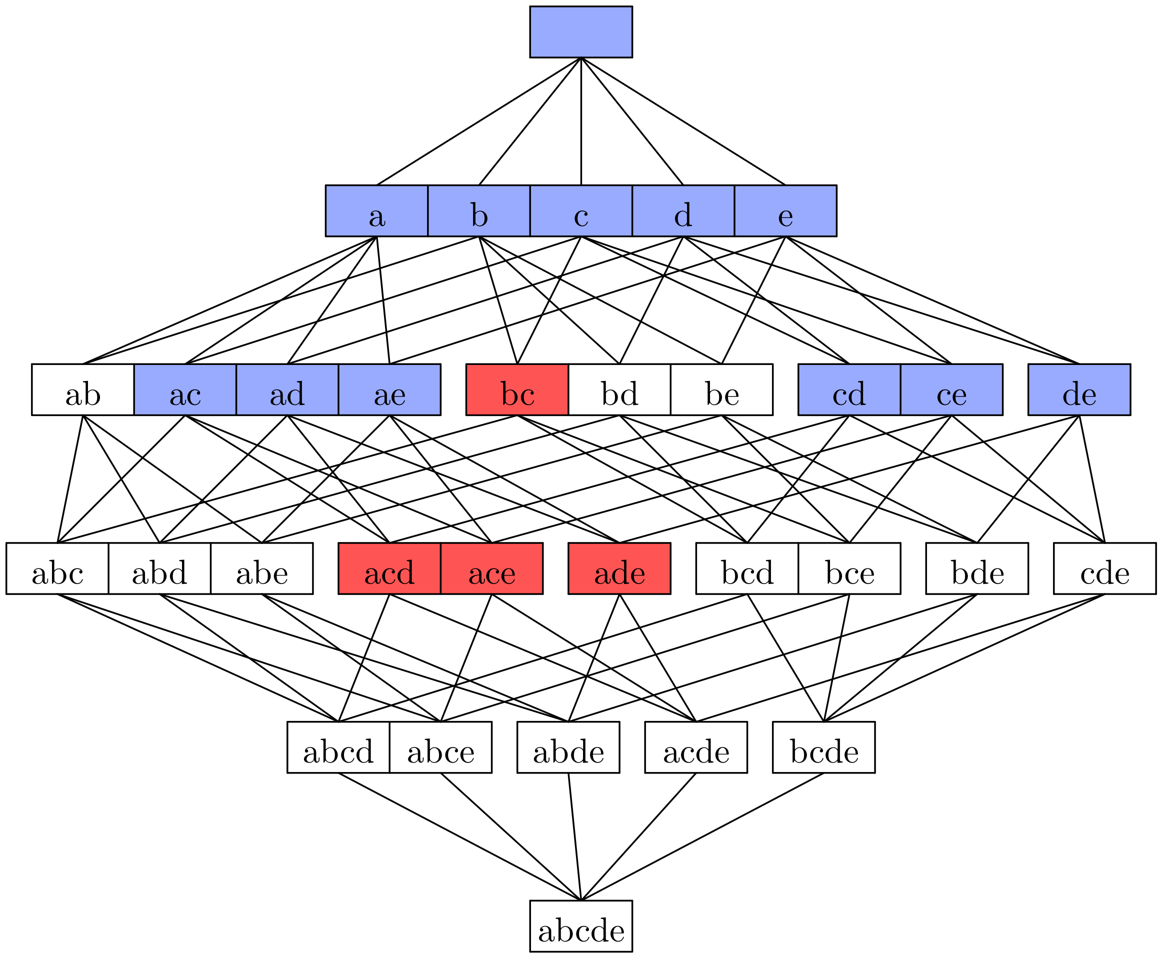
Subset Lattice and Maximal Item Sets

transaction vector

1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$

Red boxes are maximal item sets, white boxes infrequent item sets.

subset lattice with maximal item sets ($s_{\min} = 3$):



The Apriori Algorithm

[Agrawal and Srikant 1994]

Searching for Frequent Item Sets

One possible scheme for the search:

- Determine the support of the one element item sets and discard the infrequent items.
- Form candidate item sets with two items (both items must be frequent), determine their support, and discard the infrequent item sets.
- Form candidate item sets with three items (all pairs must be frequent), determine their support, and discard the infrequent item sets.
- Continue by forming candidate item sets with four, five etc. items until no candidate item set is frequent.

This is the general scheme of the **Apriori Algorithm**.

It is based on two main steps: **candidate generation** and **pruning**.

All frequent item set mining algorithms are based on these steps in some form.

The Apriori Algorithm 1

```
function apriori $(A, T, s_{\min})$                  (* Apriori algorithm *)
begin
    $k$ $:= 1$;                                      (* initialize the item set size *)
    $E_k := \bigcup_{a \in A}\{\{a\}\}$;             (* start with single element sets *)
    $F_k := \mathrm{prune}(E_k, T, s_{\min})$;       (* and determine the frequent ones *)
    while $F_k \neq \emptyset$ do begin              (* while there are frequent item sets *)
        $E_{k+1} := \mathrm{candidates}(F_k)$;       (* create item sets with one item more *)
        $F_{k+1} := \mathrm{prune}(E_{k+1}, T, s_{\min})$; (* and determine the frequent ones *)
        $k$ $:= k + 1$;                              (* increment the item counter *)
    end;
    return $\bigcup_{j=1}^{k} F_j$;                  (* return the frequent item sets *)
end (* apriori *)
```

The Apriori Algorithm 2

```
function candidates ($F_k$)                    (* generate candidates with $k+1$ items *)
begin
    $E := \emptyset$;                              (* initialize the set of candidates *)
    forall $f_1, f_2 \in F_k$                      (* traverse all pairs of frequent item sets *)
    with  $f_1 = \{a_1, \dots, a_{k-1}, a_k\}$      (* that differ only in one item and *)
    and   $f_2 = \{a_1, \dots, a_{k-1}, a'_k\}$     (* are in a lexicographic order *)
    and   $a_k < a'_k$ do begin                    (* (the order is arbitrary, but fixed) *)
        $f := f_1 \cup f_2 = \{a_1, \dots, a_{k-1}, a_k, a'_k\}$;   (* union has $k+1$ items *)
        if $\forall a \in f: \; f - \{a\} \in F_k$      (* only if all subsets are frequent, *)
        then $E := E \cup \{f\}$;                   (* add the new item set to the candidates *)
    end;                                        (* (otherwise it cannot be frequent) *)
    return $E$;                                   (* return the generated candidates *)
end (* candidates *)
```

The Apriori Algorithm 3

```
function prune ($E, T, s_{\min}$)          (* prune infrequent candidates *)
begin
    forall $e \in E$ do                     (* initialize the support counters *)
        $s_T(e) := 0$;                      (* of all candidates to be checked *)
    forall $t \in T$ do                     (* traverse the transactions *)
        forall $e \in E$ do                 (* traverse the candidates *)
            if $e \subseteq t$              (* if transaction contains the candidate, *)
            then $s_T(e) := s_T(e) + 1$;    (* increment the support counter *)
    $F := \emptyset$;                       (* initialize the set of frequent candidates *)
    forall $e \in E$ do                     (* traverse the candidates *)
        if $s_T(e) \geq s_{\min}$           (* if a candidate is frequent, *)
        then $F := F \cup \{e\}$;           (* add it to the set of frequent candidates *)
    return $F$;                             (* return the pruned set of candidates *)
end (* prune *)
```

Searching for Frequent Item Sets

- The Apriori algorithm searches the subset lattice top-down level by level.
- Collecting the frequent item sets of size k in a *set* F_k has drawbacks:
  A frequent item set of size $k+1$ can be formed in

  $$j = \frac{k(k+1)}{2}$$

  possible ways. (For infrequent item sets the number may be smaller.)

  As a consequence, the candidate generation step may carry out a lot of redundant work, since it suffices to generate each candidate item set once.
- **Question:** Can we reduce or even eliminate this redundant work?

  More generally:
  How can we make sure that any candidate item set is generated at most once?
- **Idea:** Assign to each item set a unique parent item set,
  from which this item set is to be generated.

Searching for Frequent Item Sets

- A core problem is that an item set of size k (that is, with k items) can be generated in $k!$ different ways (on $k!$ paths in the Hasse diagram), because in principle the items may be added in any order.

- If we consider an item by item process of building an item set (which can be imagined as a levelwise traversal of the lattice), there are k possible ways of forming an item set of size k from item sets of size $k - 1$ by adding the remaining item.

- It is obvious that it suffices to consider each item set at most once in order to find the frequent ones (infrequent item sets need not be generated at all).

- **Question:** Can we reduce or even eliminate this variety?

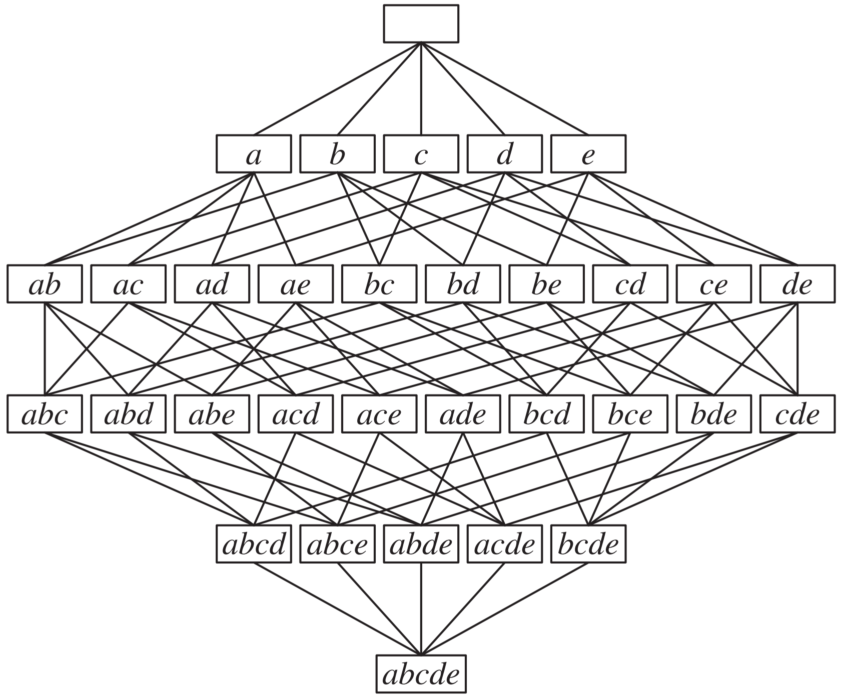
  More generally:
  How can we make sure that any candidate item set is generated at most once?

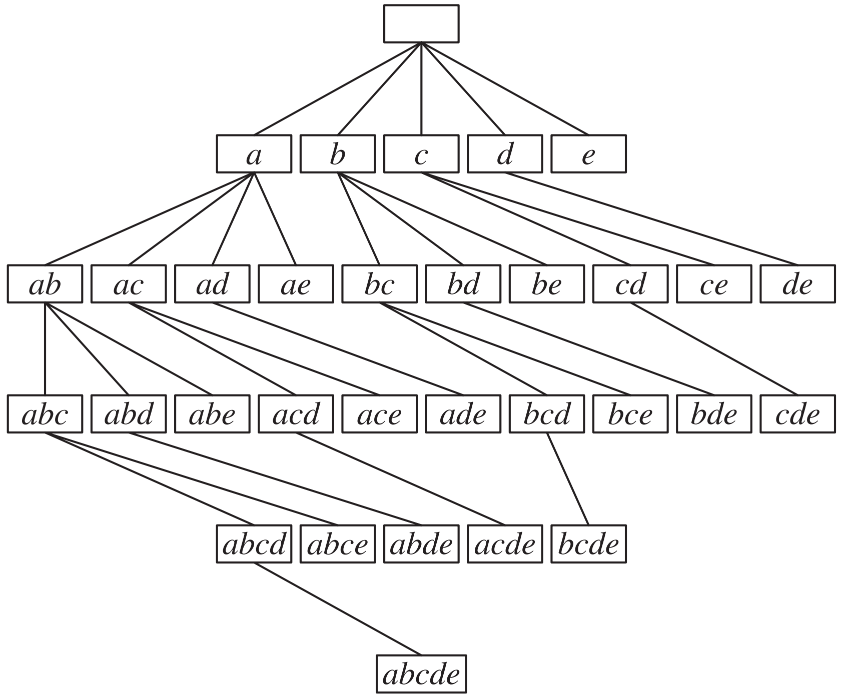
- **Idea:** Assign to each item set a unique parent item set, from which this item set is to be generated.

Searching for Frequent Item Sets

- We have to search the item subset lattice / its Hasse diagram.
- Assigning unique parents turns the Hasse diagram into a tree.
- Traversing the resulting tree explores each item set exactly once.

Subset lattice (Hasse diagram) and a possible tree for five items:





Searching with Unique Parents

Principle of a Search Algorithm based on Unique Parents:

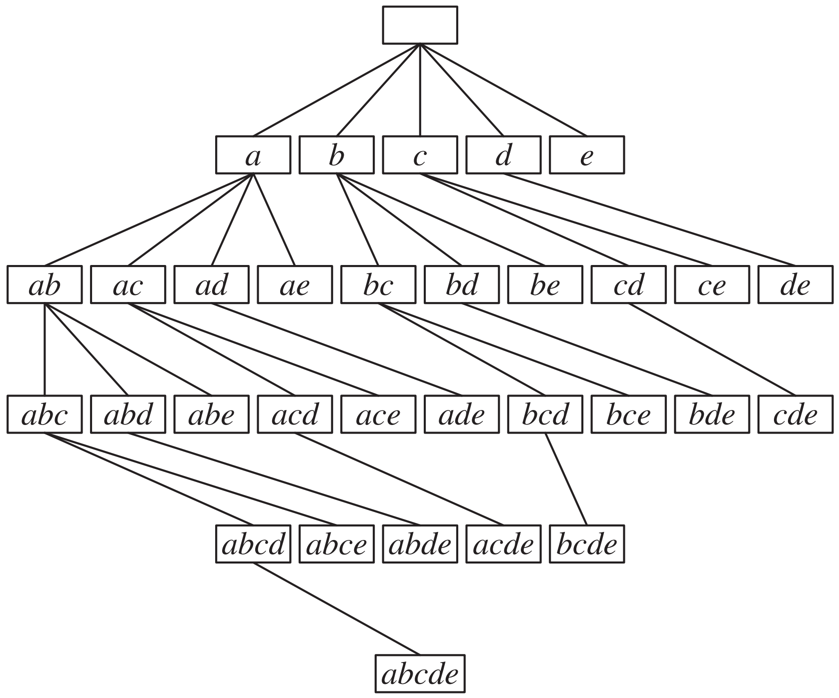
- **Base Loop:**
  - Traverse all one-element item sets (their unique parent is the empty set).
  - Recursively process all one-element item sets that are frequent.
- **Recursive Processing:**

  For a given frequent item set I:
  - Generate all extensions J of I by one item (that is, $J \supset I$, $|J| = |I| + 1$) for which the item set I is the chosen unique parent.
  - For all J: if J is frequent, process J recursively, otherwise discard J.
- **Questions:**
  - How can we formally assign unique parents?
  - How can we make sure that we generate only those extensions for which the item set that is extended is the chosen unique parent?

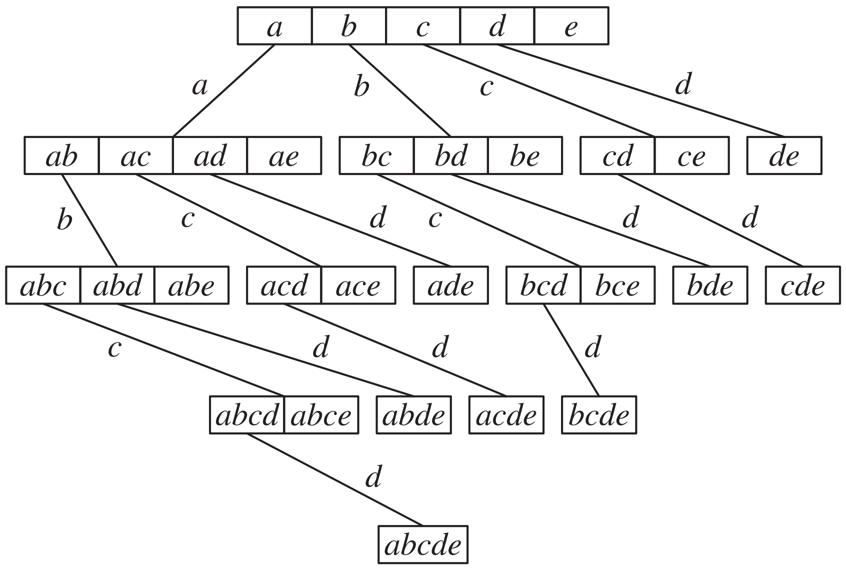
Unique Parents and Prefix Trees

- Item sets sharing the same longest proper prefix are siblings, because they have the same unique parent.
- This allows us to represent the unique parent tree as a **prefix tree** or **trie**.

Canonical parent tree and corresponding prefix tree for five items:



a b c d e
ab ac ad ae bc bd be cd ce de
abc abd abe acd ace ade bcd bce bde cde
abcd abce abde acde bcde
abcde



a b c d e
a b c d
ab ac ad ae bc bd be cd ce de
b c d c d d
abc abd abe acd ace ade bcd bce bde cde
c d d d
abcd abce abde acde bcde
d
abcde

Apriori: Levelwise Search

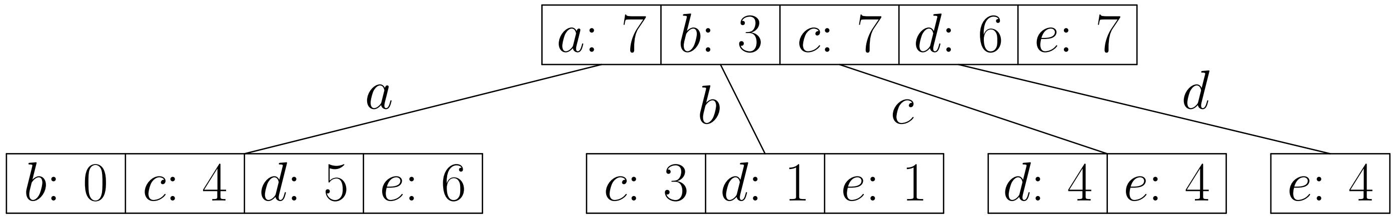
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$

a: 7	b: 3	c: 7	d: 6	e: 7

- Example transaction database with 5 items and 10 transactions.
- Minimum support: 30%, i.e., at least 3 transactions must contain the item set.
- All one item sets are frequent $\rightarrow$ full second level is needed.

Apriori: Levelwise Search

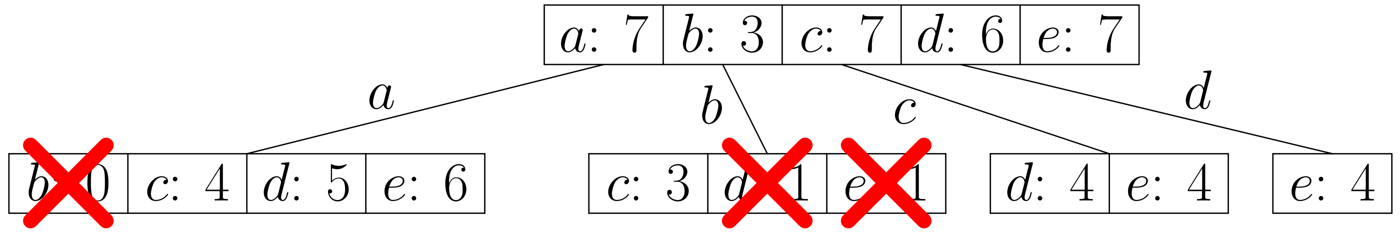
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Determining the support of item sets: For each item set traverse the database and count the transactions that contain it (highly inefficient).
- Better: Traverse the tree for each transaction and find the item sets it contains (efficient: can be implemented as a simple doubly recursive procedure).

Apriori: Levelwise Search

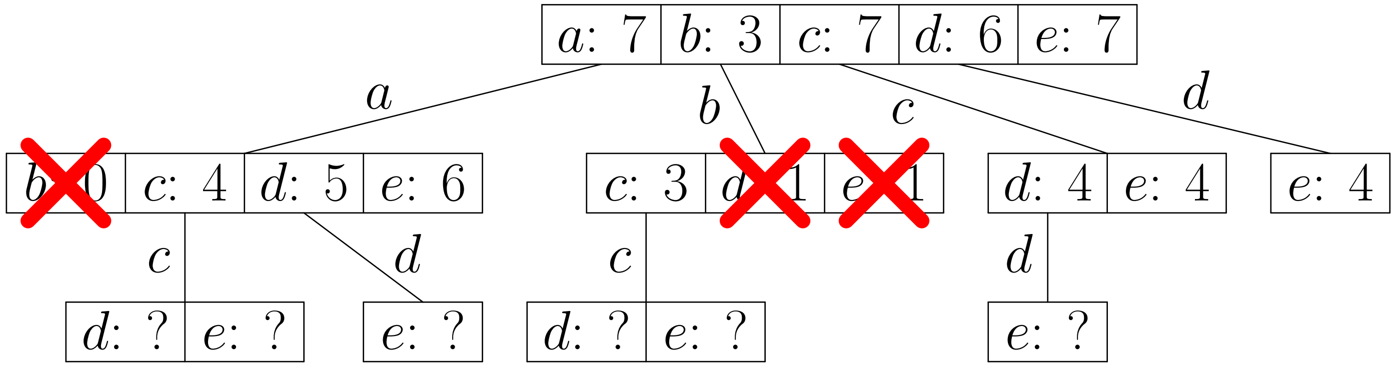
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Minimum support: 30%, i.e., at least 3 transactions must contain the item set.
- Infrequent item sets: $\{a, b\}$, $\{b, d\}$, $\{b, e\}$.
- The subtrees starting at these item sets can be pruned.

Apriori: Levelwise Search

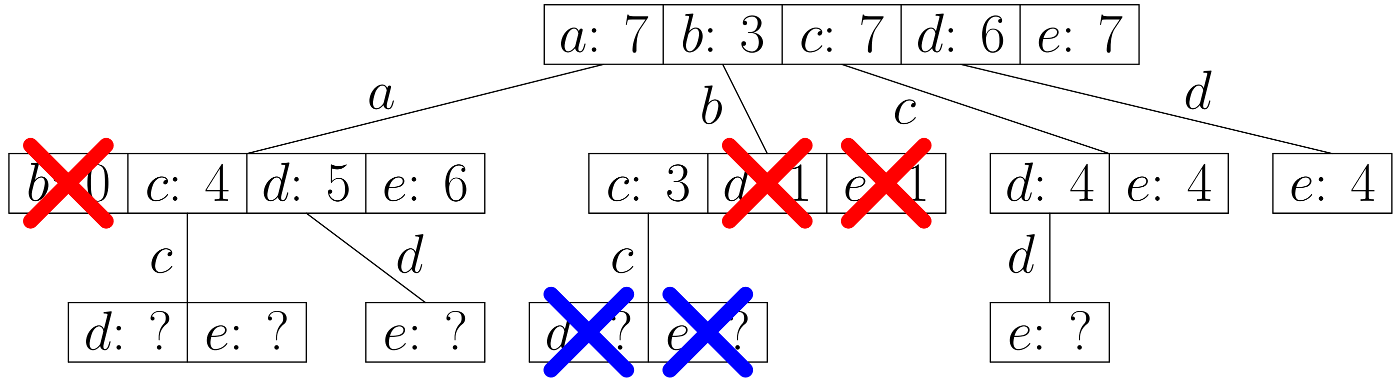
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Generate candidate item sets with 3 items (parents must be frequent).
- Before counting, check whether the candidates contain an infrequent item set.
  - An item set with k items has k subsets of size $k - 1$.
  - The parent is only one of these subsets.

Apriori: Levelwise Search

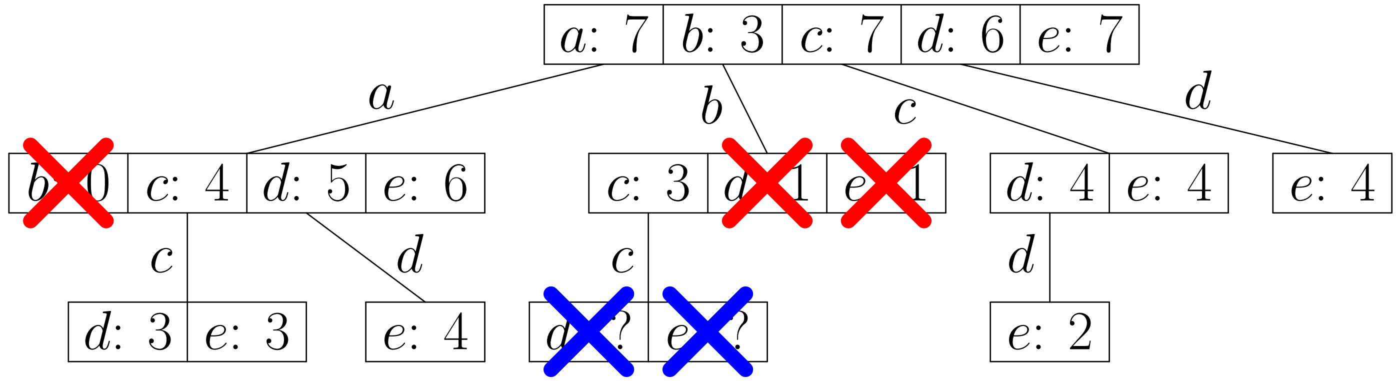
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- The item sets $\{b, c, d\}$ and $\{b, c, e\}$ can be pruned, because
  - $\{b, c, d\}$ contains the infrequent item set $\{b, d\}$ and
  - $\{b, c, e\}$ contains the infrequent item set $\{b, e\}$.

Apriori: Levelwise Search

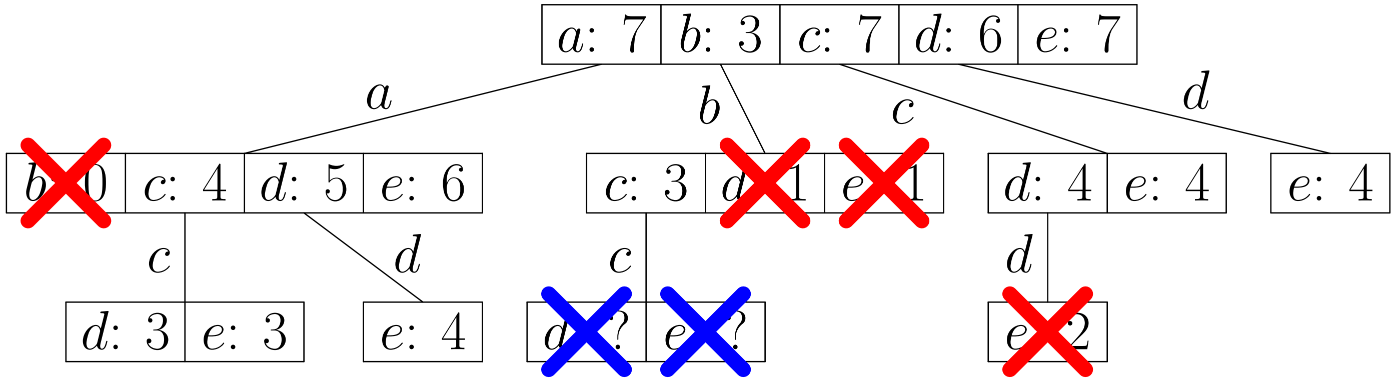
1:	$\{a, d, e\}$
2:	$\{b, c, d\}$
3:	$\{a, c, e\}$
4:	$\{a, c, d, e\}$
5:	$\{a, e\}$
6:	$\{a, c, d\}$
7:	$\{b, c\}$
8:	$\{a, c, d, e\}$
9:	$\{c, b, e\}$
10:	$\{a, d, e\}$



- Only the remaining four item sets of size 3 are evaluated.

Apriori: Levelwise Search

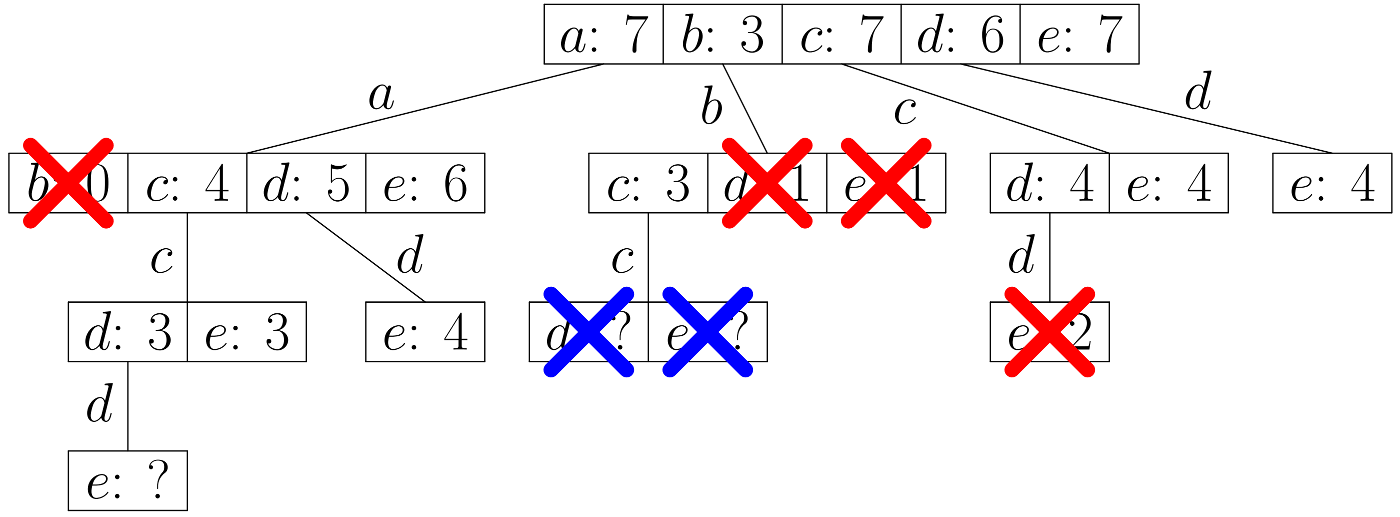
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Minimum support: 30%, i.e., at least 3 transactions must contain the item set.
- Infrequent item set: $\{c, d, e\}$.

Apriori: Levelwise Search

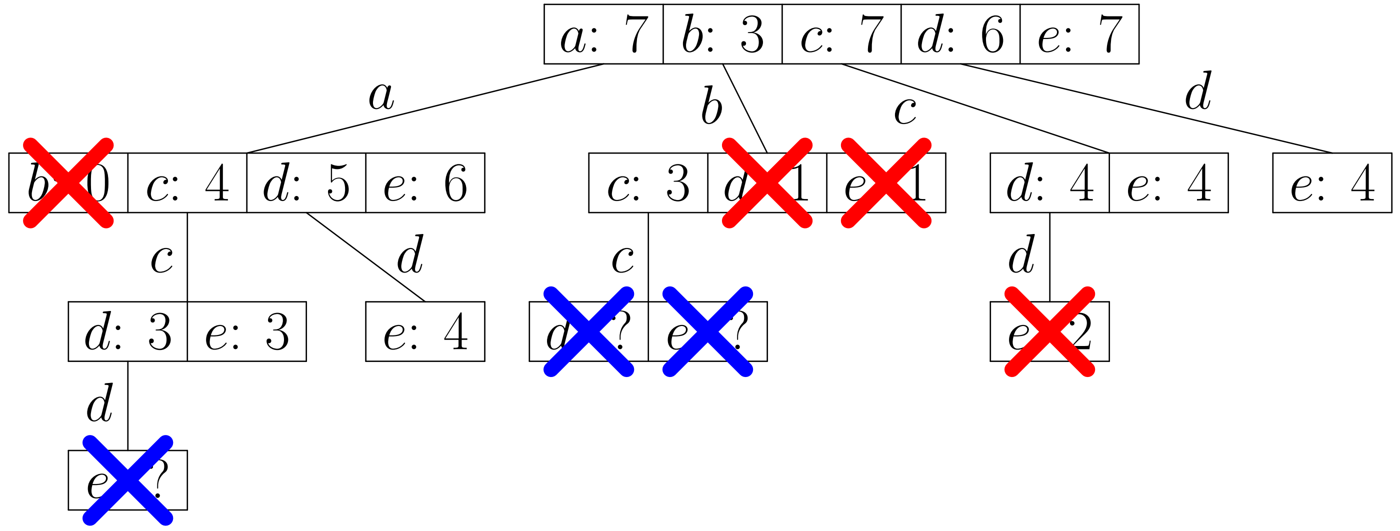
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Generate candidate item sets with 4 items (parents must be frequent).
- Before counting, check whether the candidates contain an infrequent item set.

Apriori: Levelwise Search

1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- The item set $\{a, c, d, e\}$ can be pruned,
  because it contains the infrequent item set $\{c, d, e\}$.
- Consequence: No candidate item sets with four items.
- Fourth access to the transaction database is not necessary.

Apriori: Node Organization 1

Idea: Optimize the organization of the counters and the child pointers.

Direct Indexing:

- Each node is a simple vector (array) of counters.
- An item is used as a direct index to find the counter.
- Advantage: Counter access is extremely fast.
- Disadvantage: Memory usage can be high due to "gaps" in the index space.

Sorted Vectors:

- Each node is a vector (array) of item/counter pairs.
- A binary search is necessary to find the counter for an item.
- Advantage: Memory usage may be smaller, no unnecessary counters.
- Disadvantage: Counter access is slower due to the binary search.

Apriori: Node Organization 2

Hash Tables:

- Each node is a vector (array) of item/counter pairs (closed hashing).
- The index of a counter is computed from the item code.
- Advantage: Faster counter access than with binary search.
- Disadvantage: Higher memory usage than sorted vectors (pairs, fill rate). The order of the items cannot be exploited.

Child Pointers:

- The deepest level of the item set tree does not need child pointers.
- Fewer child pointers than counters are needed.

  $\rightarrow$ It pays to represent the child pointers in a separate array.
- The sorted array of item/counter pairs can be reused for a binary search.

Apriori: Item Coding

- Items are coded as consecutive integers starting with 0 (needed for the direct indexing approach).
- The size and the number of the "gaps" in the index space depends on how the items are coded.
- Idea: It is plausible that frequent item sets consist of frequent items.
  - Sort the items w.r.t. their frequency (group frequent items).
  - Sort descendingly: Prefix tree has fewer nodes.
  - Sort ascendingly: There are fewer and smaller index "gaps".
  - Empirical evidence: sorting ascendingly is better.
- Extension: Sort items w.r.t. the sum of the sizes of the transacions that cover them.
  - Empirical evidence: Better than simple item frequencies.

Apriori: Recursive Counting

- The items in a transaction are sorted (ascending item codes).
- Processing a transaction is then a *doubly recursive procedure*.
  To process a transaction for a node of the item set tree:
  - Go to the child corresponding to the first item in the transaction and count the remainder of the transaction recursively for that child.

    (In the currently deepest level of the tree we increment the counter corresponding to the item instead of going to the child node.)
  - Discard the first item of the transaction and
    process it recursively for the node itself.
- Optimizations:
  - Directly skip all items preceding the first item in the node.
  - Abort the recursion if the first item is beyond the last one in the node.
  - Abort the recursion if a transaction is too short to reach the deepest level.

Apriori: Transaction Representation

Direct Representation:

- Each transaction is represented as an array of items.
- The transactions are stored in a simple list.

Organization as a Prefix Tree:

- The items in each transaction are sorted.
- Transactions with the same prefix are grouped together.
- Advantage: a common prefix is processed only once.
- Gains from this organization depend on how the items are coded:
  - Common transaction prefixes are more likely
    if the items are sorted with descending frequency.

Summary Apriori

Basic Processing Scheme

- Breadth-first/levelwise traversal of the subset lattice.
- Candidates are formed by merging item sets that differ in only one item.
- Support counting is done with a doubly recursive procedure.

Advantages

- "Perfect" pruning of infrequent candidate item sets (with infrequent subsets).

Disadvantages

- Can require a lot of memory (since all frequent item sets are represented).
- Support counting takes very long for large transactions.

Software

- `http://www.borgelt.net/apriori.html`

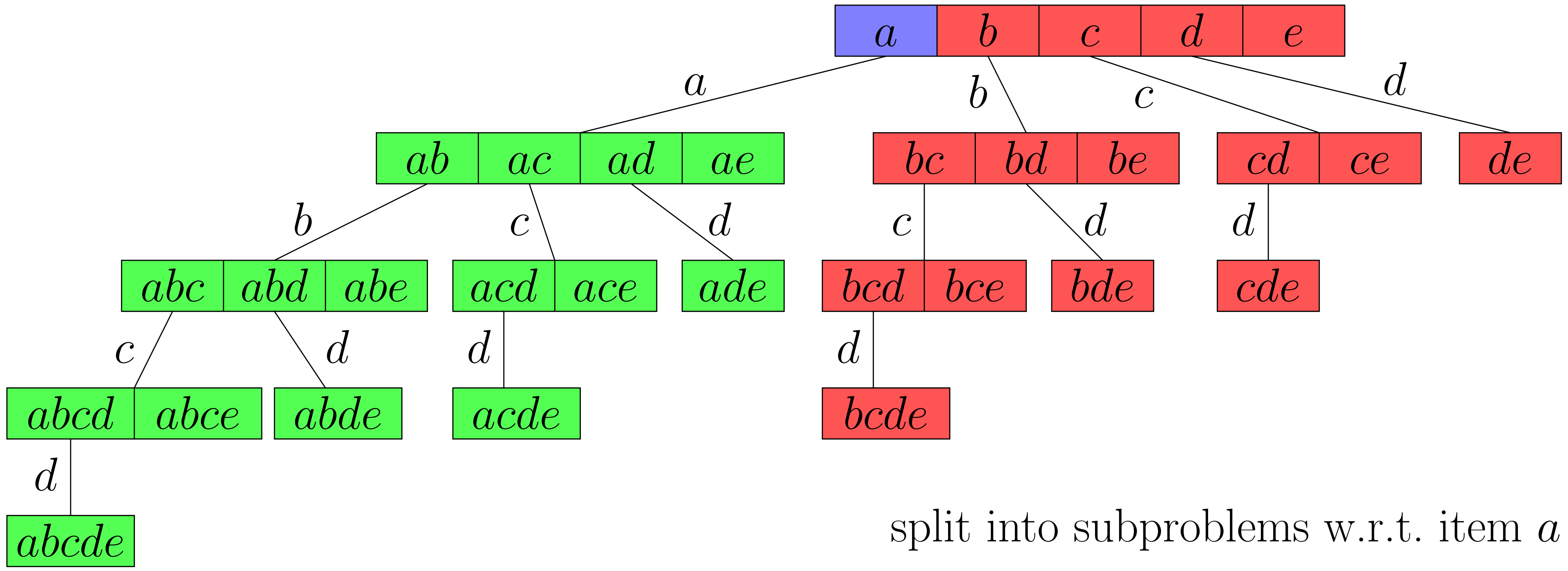
Depth-First Search and Conditional Databases

- In contrast to the levelwise search of the Apriori algorithm, the **Eclat Algorithm** executes a depth-first search in the prefix tree.
- This depth-first search can also be seen as a **divide-and-conquer scheme**:
  - Let the item order be $a < b < c \ldots$.
  - Restrict the transaction vector to those transactions that contain a. This is the **conditional database** for the prefix a.

    Recursively search this conditional database for frequent item sets and add the prefix a to all frequent item sets found in the recursion.
  - Remove the item a from the transactions in the full transaction vector. This is the **conditional database** for item sets without a.

    Recursively search this conditional database for frequent item sets.
- With this scheme only frequent one-element item sets have to be determined. Larger item sets result from adding possible prefixes.

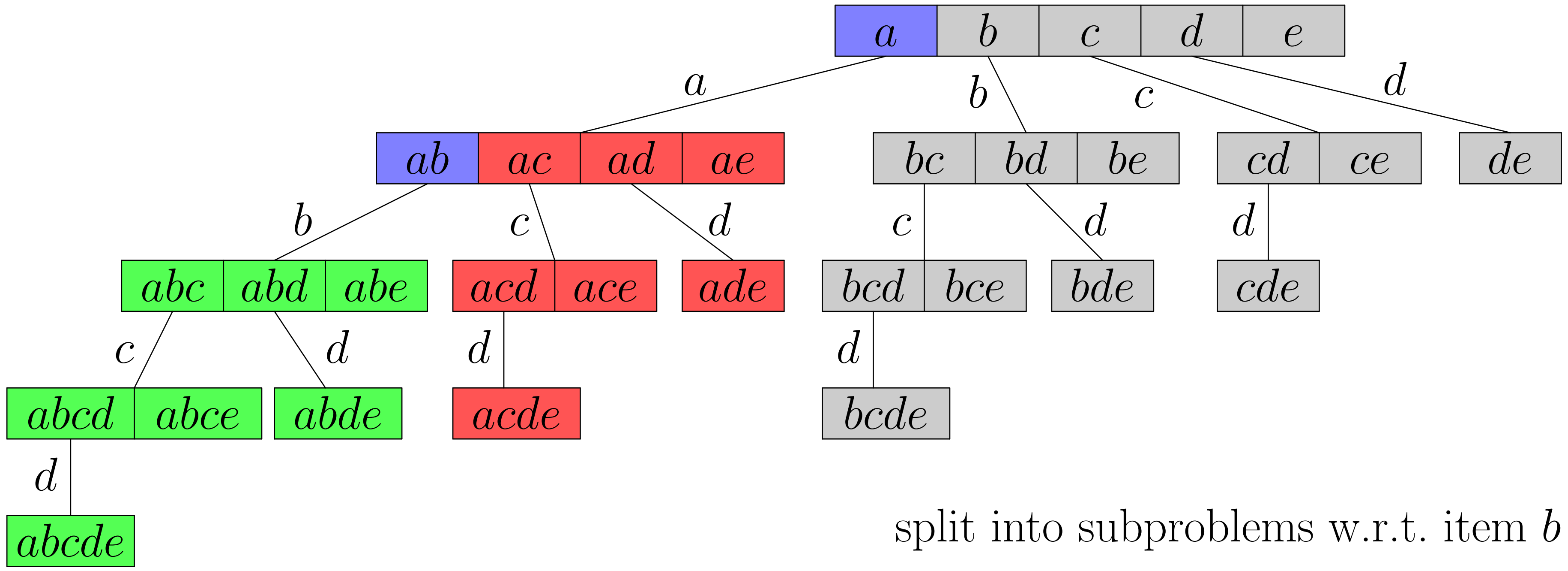
Depth-First Search and Conditional Databases



split into subproblems w.r.t. item a

- blue : item set consisting of only item a.
  green: item sets containing item a (and at least one other item).
  red : item sets not containing item a (but at least one other item).
- green: database with transactions containing a.
  red : database with all transactions, but with item a removed.

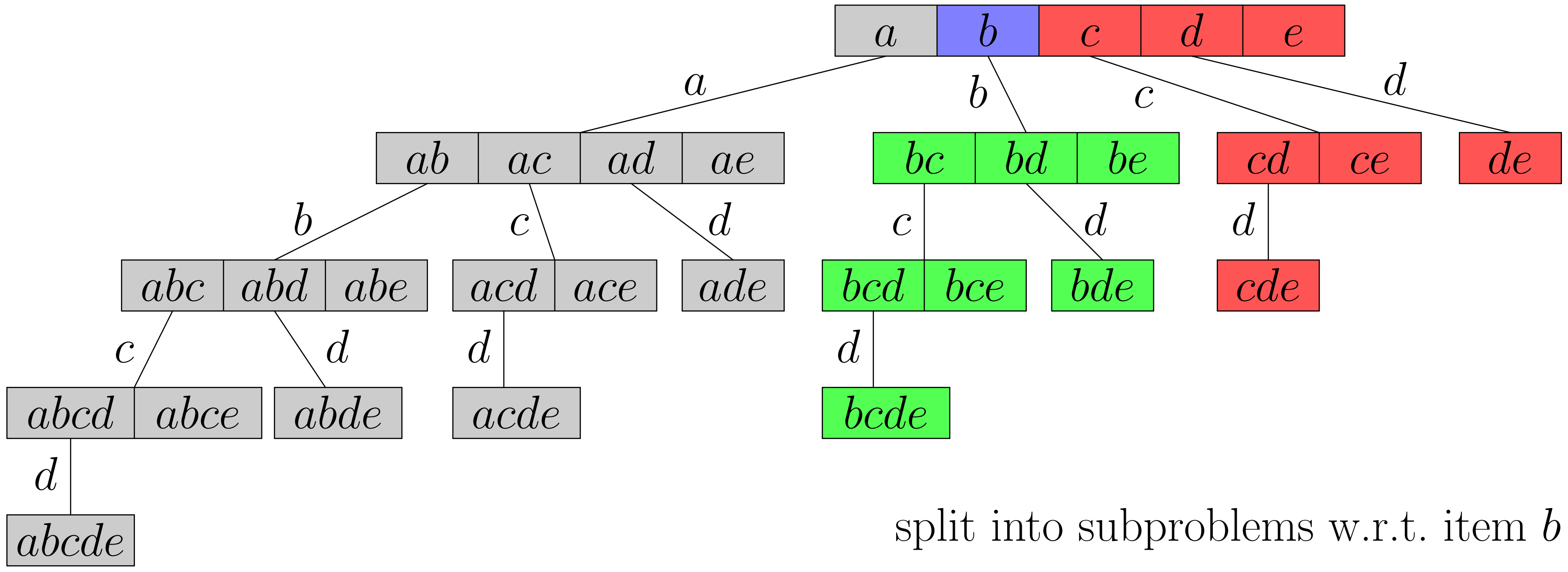
Depth-First Search and Conditional Databases



split into subproblems w.r.t. item b

- blue : item sets $\{a\}$ and $\{a, b\}$.
  green: item sets containing items a and b (and at least one other item).
  red : item sets containing item a, but not item b.
- green: database with transactions containing a and b.
  red : database with transactions containing a, but with b removed.

Depth-First Search and Conditional Databases



split into subproblems w.r.t. item b

- blue : item set consisting of only item b.
  green: item sets containing item b, but not item a.
  red : item sets containing neither item a nor item b.
- green: database with transactions containing b, but not a.
  red : database with all transactions, but with a and b removed.

The Eclat Algorithm

[Zaki, Parthasarathy, Ogihara, and Li 1997]

Eclat: Basic Ideas

- The item sets are checked in lexicographic order
  (depth-first traversal of the prefix tree).
- Eclat generates more candidate item sets than Apriori,
  because it does not store the support of all visited item sets.
- Eclat uses a *vertical transaction representation*
  (see next slide for details).
- No subset tests and no subset generation is needed for the support computation.

Eclat: Transaction Representation

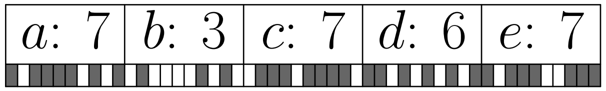
- The Apriori algorithm uses a **horizontal transaction representation**: each transaction is an array of the contained items.
  - Note that the alternative prefix tree organization is still an essentially *horizontal* representation.

- The Eclat algorithm uses a **vertical transaction representation**:
  - For each item a **transaction list** is created.
  - The transaction list of item a indicates the transactions that contain it, that is, it represents its *cover* $K_T(\{a\})$.
  - Advantage: the transaction list for a pair of items can be computed by intersecting the transaction lists of the individual items.
  - Generally, a vertical transaction representation can exploit

$$\forall I, J \subseteq A: \quad K_T(I \cup J) = K_T(I) \cap K_T(J).$$

Eclat: Depth-First Search

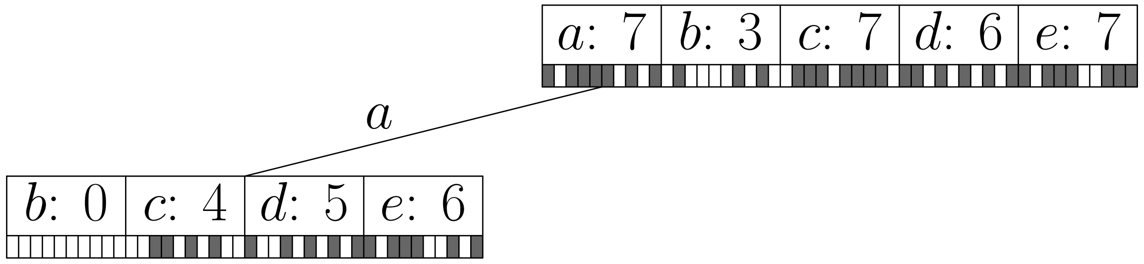
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Form a transaction list for each item. Here: bit vector representation.
  - grey: item is contained in transaction
  - white: item is not contained in transaction
- Transaction database is needed only once (for the single item transaction lists).

Eclat: Depth-First Search

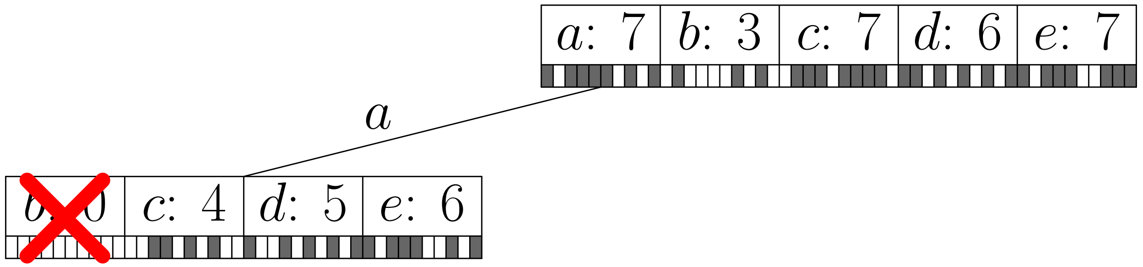
1:	$\{a, d, e\}$
2:	$\{b, c, d\}$
3:	$\{a, c, e\}$
4:	$\{a, c, d, e\}$
5:	$\{a, e\}$
6:	$\{a, c, d\}$
7:	$\{b, c\}$
8:	$\{a, c, d, e\}$
9:	$\{c, b, e\}$
10:	$\{a, d, e\}$



- Intersect the transaction list for item a with the transaction lists of all other items (*conditional database*).
- Count the number of set bits (number of containing transactions).
- The item set $\{a, b\}$ is infrequent and can be pruned.

Eclat: Depth-First Search

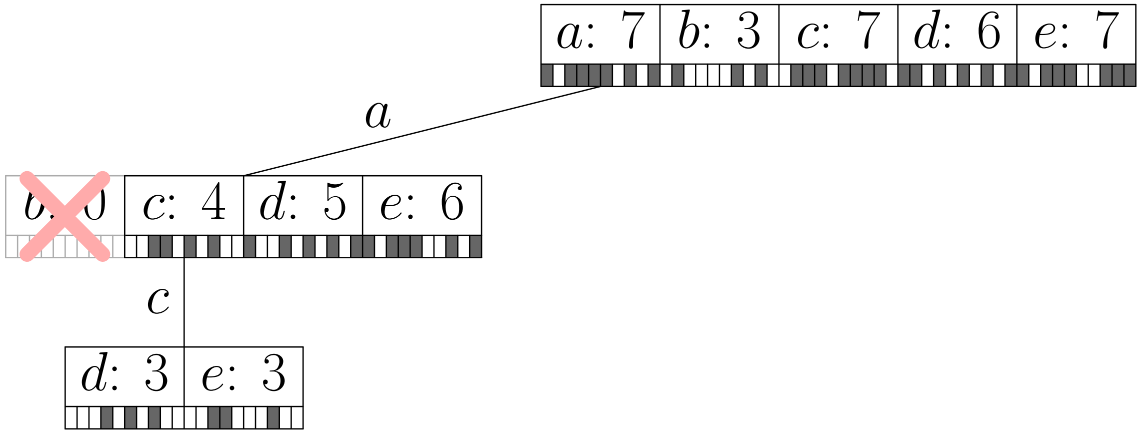
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Intersect the transaction list for item a with the transaction lists of all other items (*conditional database*).
- Count the number of set bits (number of containing transactions).
- The item set $\{a, b\}$ is infrequent and can be pruned.

Eclat: Depth-First Search

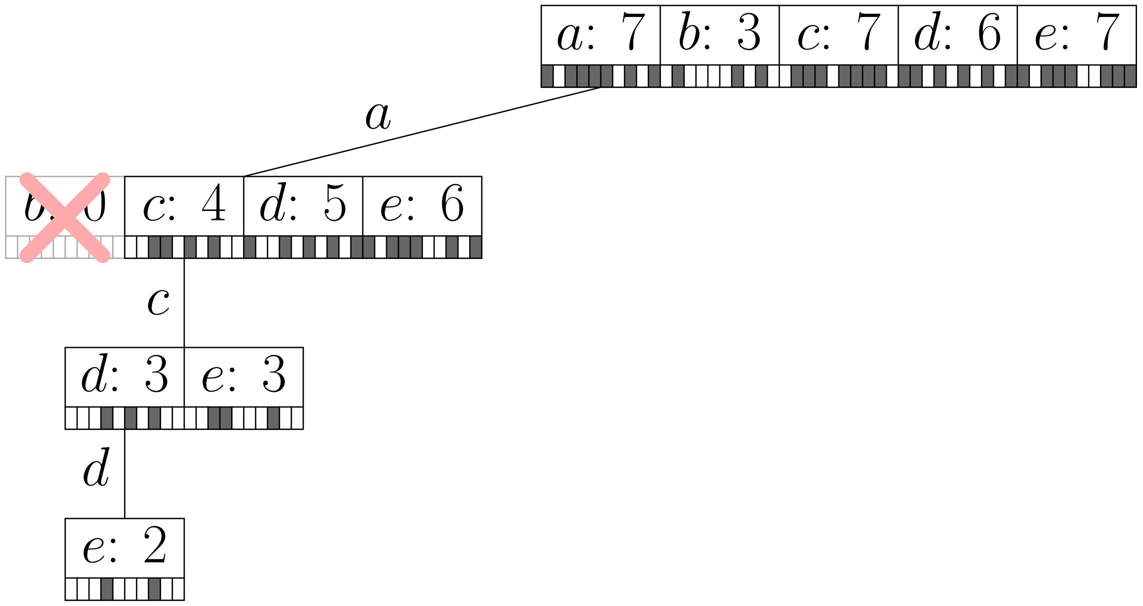
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Intersect the transaction list for $\{a, c\}$ with the transaction lists of $\{a, x\}$, $x \in \{d, e\}$.
- Result: Transaction lists for the item sets $\{a, c, d\}$ and $\{a, c, e\}$.

Eclat: Depth-First Search

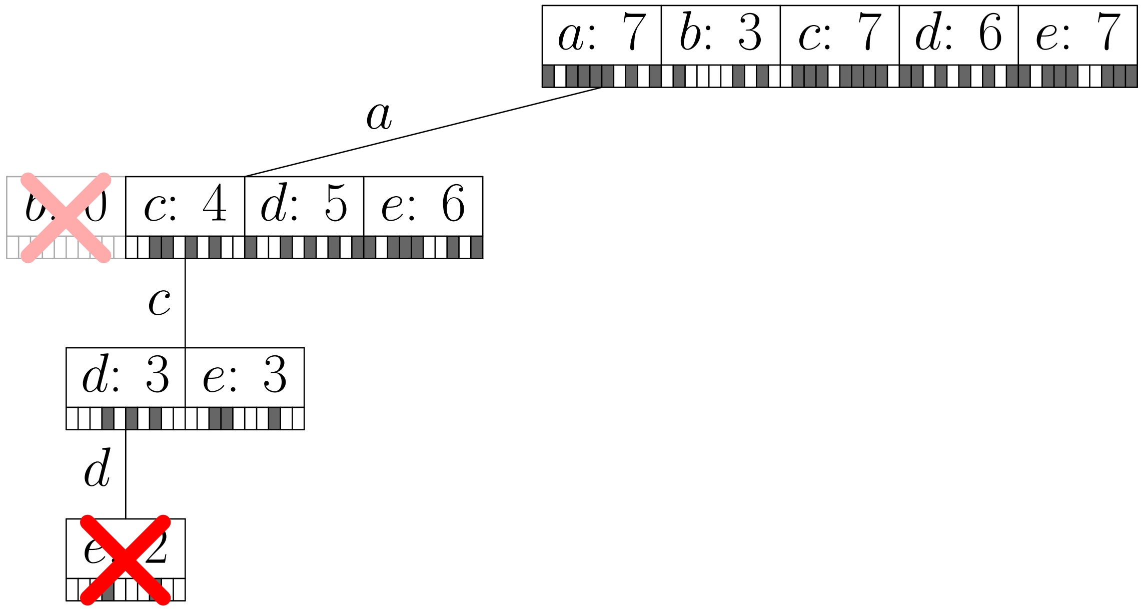
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Intersect the transaction list for $\{a, c, d\}$ and $\{a, c, e\}$.
- Result: Transaction list for the item set $\{a, c, d, e\}$.
- With Apriori this item set could be pruned before counting, because it was known that $\{c, d, e\}$ is infrequent.

Eclat: Depth-First Search

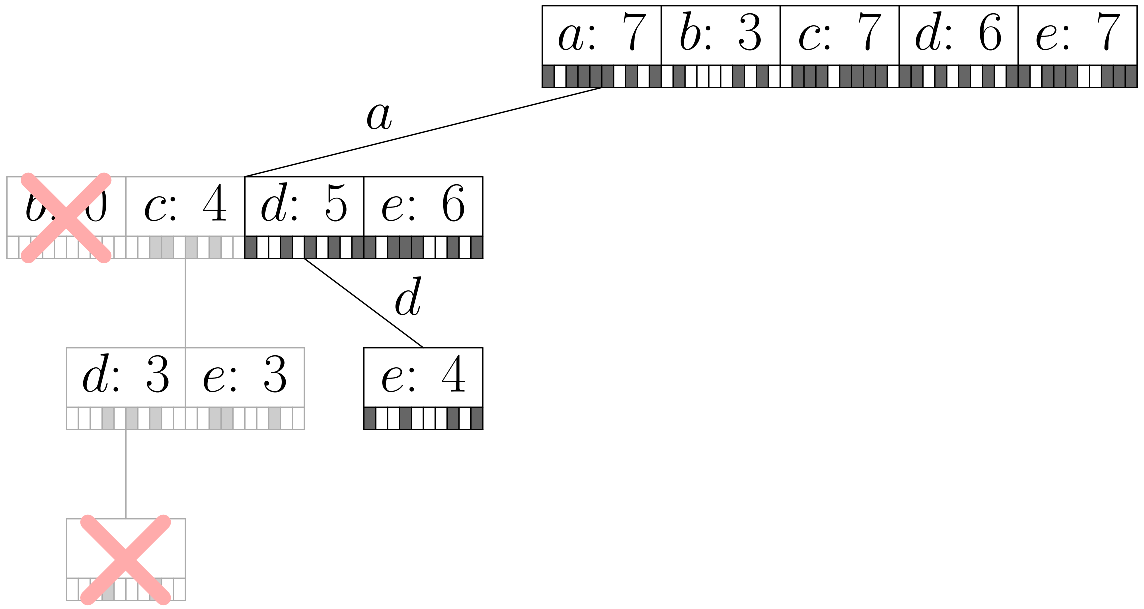
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Intersect the transaction list for $\{a, c, d\}$ and $\{a, c, e\}$.
- Result: Transaction list for the item set $\{a, c, d, e\}$.
- With Apriori this item set could be pruned before counting, because it was known that $\{c, d, e\}$ is infrequent.

Eclat: Depth-First Search

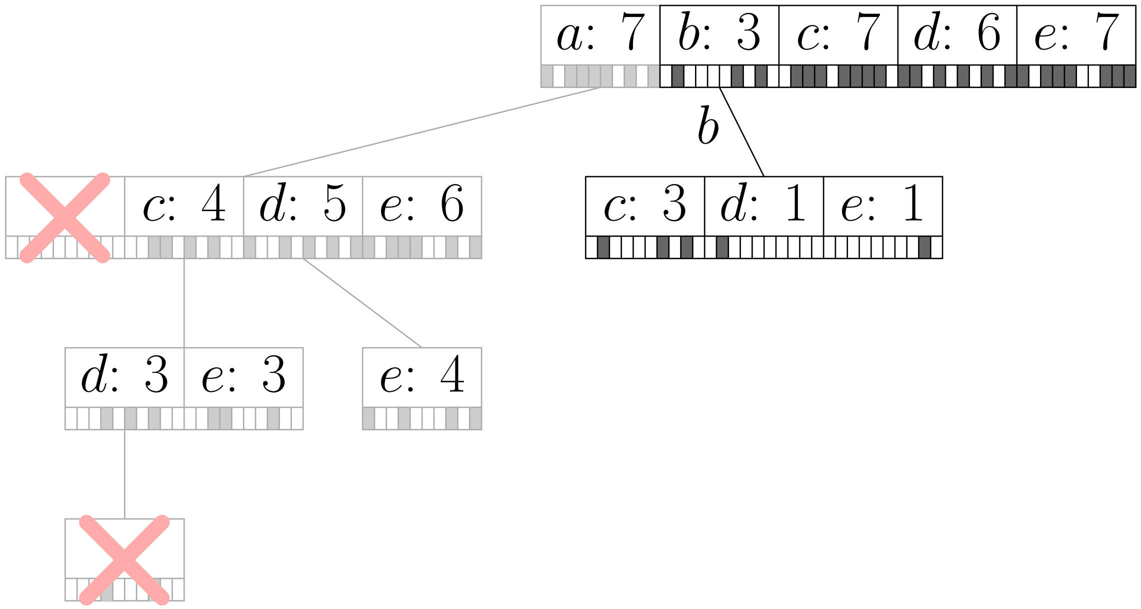
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Backtrack to the second level of the search tree and intersect the transaction list for $\{a, d\}$ and $\{a, e\}$.
- Result: Transaction list for $\{a, d, e\}$.

Eclat: Depth-First Search

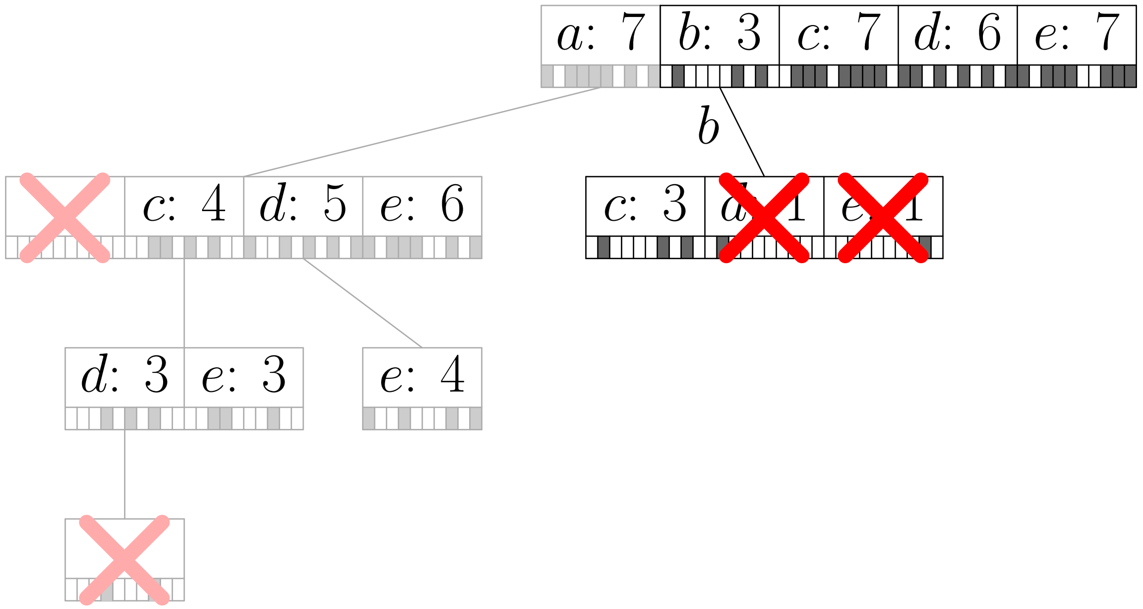
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Backtrack to the first level of the search tree and intersect the transaction list for b with the transaction lists for c, d, and e.
- Result: Transaction lists for the item sets $\{b, c\}$, $\{b, d\}$, and $\{b, e\}$.
- Only one item set with sufficient support $\rightarrow$ prune all subtrees.

Eclat: Depth-First Search

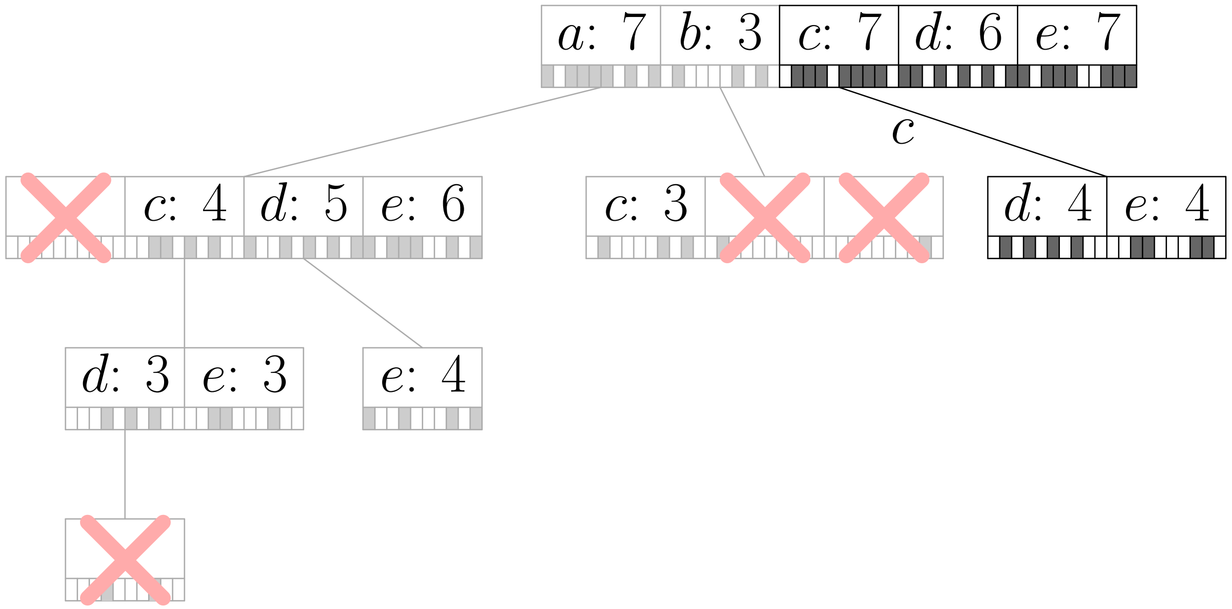
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Backtrack to the first level of the search tree and intersect the transaction list for b with the transaction lists for c, d, and e.
- Result: Transaction lists for the item sets $\{b, c\}$, $\{b, d\}$, and $\{b, e\}$.
- Only one item set with sufficient support $\rightarrow$ prune all subtrees.

Eclat: Depth-First Search

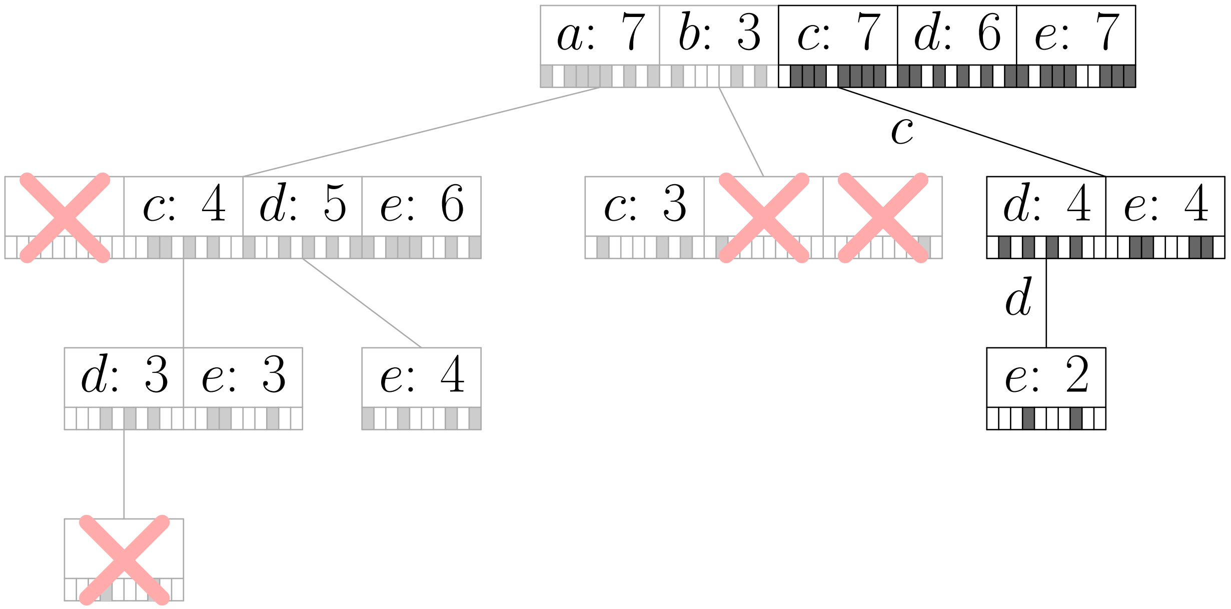
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Backtrack to the first level of the search tree and intersect the transaction list for c with the transaction lists for d and e.
- Result: Transaction lists for the item sets $\{c, d\}$ and $\{c, e\}$.

Eclat: Depth-First Search

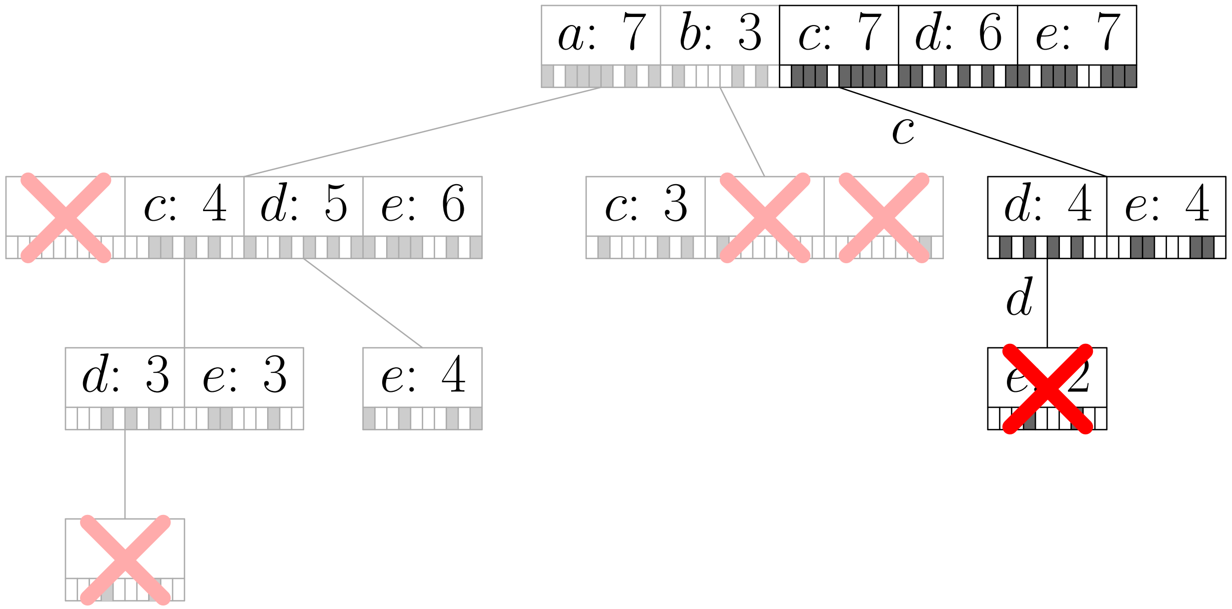
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Intersect the transaction list for $\{c, d\}$ and $\{c, e\}$.
- Result: Transaction list for $\{c, d, e\}$.
- Infrequent item set: $\{c, d, e\}$.

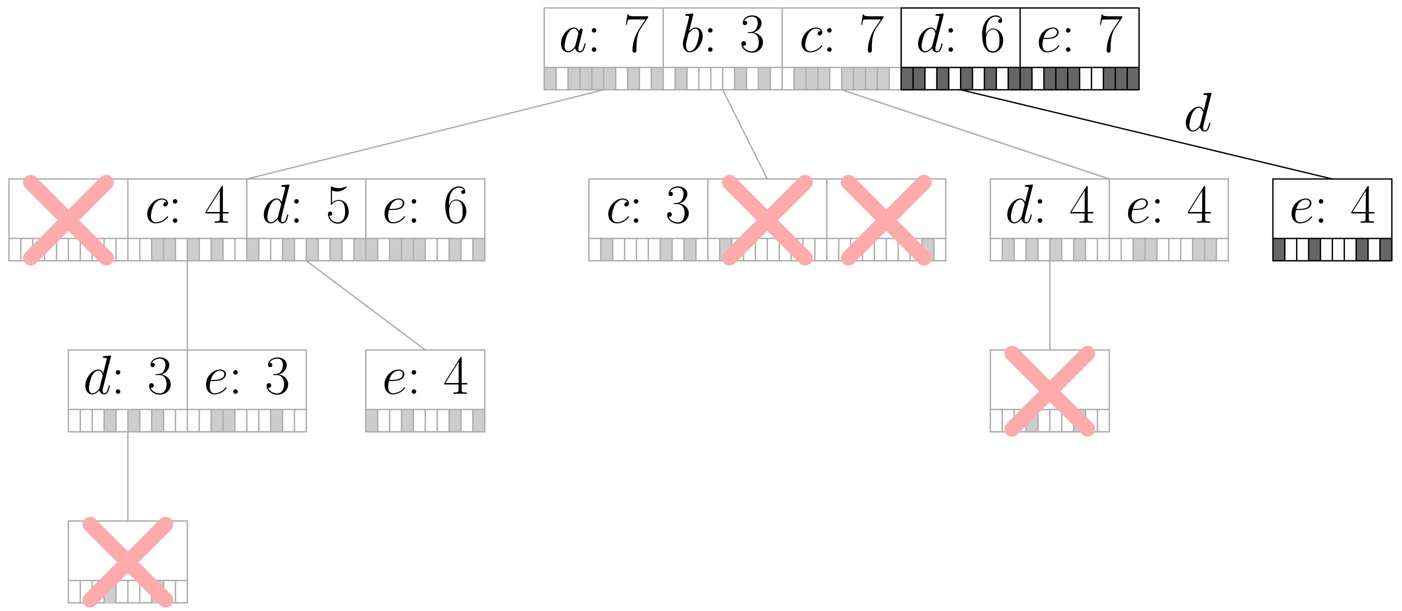
Eclat: Depth-First Search

1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{c, b, e\}$
10: $\{a, d, e\}$



- Intersect the transaction list for $\{c, d\}$ and $\{c, e\}$.
- Result: Transaction list for $\{c, d, e\}$.
- Infrequent item set: $\{c, d, e\}$.

1:	$\{a, d, e\}$
2:	$\{b, c, d\}$
3:	$\{a, c, e\}$
4:	$\{a, c, d, e\}$
5:	$\{a, e\}$
6:	$\{a, c, d\}$
7:	$\{b, c\}$
8:	$\{a, c, d, e\}$
9:	$\{c, b, e\}$
10:	$\{a, d, e\}$



- Backtrack to the first level of the search tree and intersect the transaction list for d with the transaction list for e.
- Result: Transaction list for the item set $\{d, e\}$.
- With this step the search is finished.

Eclat: Bit Matrices and Item Coding

Bit Matrices

- Represent transactions as a bit matrix:
  - Each column corresponds to an item.
  - Each row corresponds to a transaction.
- Normal and sparse representation of bit matrices:
  - Normal: one memory bit per matrix bit, zeros represented.
  - Sparse: lists of column indices of set bits (transaction lists).
- Which representation is preferable depends on the ratio of set bits to cleared bits.

Item Coding

- Sorting the item descendingly w.r.t. their frequency (individual or transaction size sum) leads to a better structure of the search tree.

Summary Eclat

Basic Processing Scheme

- Depth-first traversal of the prefix tree.
- Data is represented as lists of transaction ids (one per item).
- Support counting is done by intersecting lists of transaction ids.

Advantages

- Depth-first search reduces memory requirements.
- Usually (considerably) faster than Apriori.

Disadvantages

- Difficult to execute for modern processors (branch prediction).

Software

- `http://www.borgelt.net/eclat.html`

Additional Frequent Item Set Filtering

- **General problem of frequent item set mining:**

  The number of frequent item sets, even the number of closed or maximal item sets, can exceed the number of transactions in the database by far.

- Therefore: Additional filtering is necessary to find
  the ”“relevant” or “interesting” frequent item sets.

- General idea: **Compare support to expectation.**
  - Item sets consisting of items that appear frequently
    are likely to have a high support.
  - However, this is not surprising:
    we expect this even if the occurrence of the items is independent.
  - Additional filtering should remove item sets with a support
    close to the support expected from an independent occurrence.

Additional Frequent Item Set Filtering

Full Independence

- Evaluate item sets with

$$\varrho_{\text{fi}}(I) \;=\; \frac{s_T(I) \cdot n^{|I|-1}}{\prod_{a \in I} s_T(\{a\})} \;=\; \frac{\hat{p}_T(I)}{\prod_{a \in I} \hat{p}_T(\{a\})}.$$

  an require a minimum value for this measure.
  ($\hat{p}_T$ is the probability estimate based on T.)

- Assumes full independence of the items in order
  to form an expectation about the support of an item set.

- Advantage: Can be computed from only the support of the item set
  and the support values of the individual items.

- Disadvantage: If some item set I scores high on this measure,
  then all $J \supset I$ are also likely to score high,
  even if the items in $J - I$ are independent of I.

Additional Frequent Item Set Filtering

Incremental Independence

- Evaluate item sets with

$$\varrho_{\text{ii}}(I) \;=\; \min_{a \in I} \frac{n\ s_T(I)}{s_T(I - \{a\}) \cdot s_T(\{a\})} \;=\; \min_{a \in I} \frac{\hat{p}_T(I)}{\hat{p}_T(I - \{a\}) \cdot \hat{p}_T(\{a\})}.$$

  an require a minimum value for this measure.
  ($\hat{p}_T$ is the probability estimate based on T.)

- Advantage: If I contains independent items,
  the minimum ensures a low value.

- Disadvantages: We need to know the support values of all subsets $I - \{a\}$.

  If there exist high scoring independent subsets I_1 and I_2
  with $|I_1| > 1$, $|I_2| > 1$, $I_1 \cap I_2 = \emptyset$ and $I_1 \cup I_2 = I$,
  the item set I still receives a high evaluation.

Additional Frequent Item Set Filtering

Subset Independence

- Evaluate item sets with

$$\varrho_{\text{si}}(I) \;=\; \min_{J \subset I, J \neq \emptyset} \frac{n\, s_T(I)}{s_T(I-J) \cdot s_T(J)} \;=\; \min_{J \subset I, J \neq \emptyset} \frac{\hat{p}_T(I)}{\hat{p}_T(I-J) \cdot \hat{p}_T(J)}.$$

  an require a minimum value for this measure.
  ($\hat{p}_T$ is the probability estimate based on T.)

- Advantage: Detects all cases where a decomposition is possible and evaluates them with a low value.

- Disadvantages: We need to know the support values of all proper subsets J.

- Improvement: Use incremental independence and in the minimum consider only items $\{a\}$ for which $I - \{a\}$ has been evaluated high.

  This captures subset independence "incrementally".

Summary Frequent Item Set Mining

- Algorithms for frequent item set mining differ in:
  - the **traversal order** of the prefix tree:
    (breadth-first/levelwise versus depth-first traversal)
  - the **transaction representation**:
    horizontal (item arrays) versus *vertical* (transaction lists)
    versus *specialized data structures* like FP-trees
  - the **types of frequent item sets** found:
    frequent versus *closed* versus *maximal item sets*
    (additional pruning methods for closed and maximal item sets)
- **Additional filtering** is necessary to reduce the size of the output.

Association Rules: Basic Notions

- Often found patterns are expressed as **association rules**, for example:

  If a customer buys **bread** and **wine**,
  then she/he will probably also buy **cheese**.

- Formally, we consider rules of the form $X \to Y$,
  with $X, Y \subseteq A$ and $X \cap Y = \emptyset$.

- **Support of a Rule** $X \to Y$:

  Either: $\varsigma_T(X \to Y) = \sigma_T(X \cup Y)$ (more common: rule is correct)

  Or: $\varsigma_T(X \to Y) = \sigma_T(X)$ (more plausible: rule is applicable)

- **Confidence of a Rule** $X \to Y$:

$$c_T(X \to Y) = \frac{\sigma_T(X \cup Y)}{\sigma_T(X)} = \frac{s_T(X \cup Y)}{s_T(X)} = \frac{s_T(I)}{s_T(X)}$$

  The confidence can be seen as an estimate of $P(Y \mid X)$.

Association Rules: Formal Definition

Given:

- a set $A = \{a_1, \ldots, a_m\}$ of items,
- a vector $T = (t_1, \ldots, t_n)$ of transactions over A,
- a real number ς_{min}, $0 < \varsigma_{\text{min}} \leq 1$, the **minimum support**,
- a real number c_{min}, $0 < c_{\text{min}} \leq 1$, the **minimum confidence**.

Desired:

- the set of all **association rules**, that is, the set

$$\mathcal{R} = \{R : X \to Y \mid \varsigma_T(R) \geq \varsigma_{\text{min}} \land c_T(R) \geq c_{\text{min}}\}.$$

General Procedure:

- Find the frequent item sets.
- Construct rules and filter them w.r.t. ς_{min} and c_{min}.

Generating Association Rules

- Which minimum support has to be used for finding the frequent item sets depends on the definition of the support of a rule:
  - If $\varsigma_T(X \to Y) = \sigma_T(X \cup Y)$,
    then $\sigma_{\min} = \varsigma_{\min}$ or equivalently $s_{\min} = \lceil n\varsigma_{\min} \rceil$.
  - If $\varsigma_T(X \to Y) = \sigma_T(X)$,
    then $\sigma_{\min} = \varsigma_{\min} c_{\min}$ or equivalently $s_{\min} = \lceil n\varsigma_{\min} c_{\min} \rceil$.
- After the frequent item sets have been found,
  the rule construction then traverses all frequent item sets I and
  splits them into disjoint subsets X and Y ($X \cap Y = \emptyset$ and $X \cup Y = I$),
  thus forming rules $X \to Y$.
  - Filtering rules w.r.t. confidence is always necessary.
  - Filtering rules w.r.t. support is only necessary if $\varsigma_T(X \to Y) = \sigma_T(X)$.

Properties of the Confidence

- From $\forall I : \forall J \subseteq I : s_T(I) \le s_T(J)$ it obviously follows

$$\forall X, Y : \forall a \in X : \quad \frac{s_T(X \cup Y)}{s_T(X)} \ge \frac{s_T(X \cup Y)}{s_T(X - \{a\})}$$

  and therefore

$$\forall X, Y : \forall a \in X : \quad c_T(X \to Y) \ge c_T(X - \{a\} \to Y \cup \{a\}).$$

  That is: **Moving an item from the antecedent to the consequent cannot increase the confidence of a rule.**

- As an immediate consequence we have

$$\forall X, Y : \forall a \in X : \quad c_T(X \to Y) < c_{\min} \ \to\ c_T(X - \{a\} \to Y \cup \{a\}) < c_{\min}.$$

  That is: **If a rule fails to meet the minimum confidence, no rules over the same item set and with a larger consequent need to be considered.**

Generating Association Rules

```
function rules (F);                                   (∗ — generate association rules ∗)
  $R := \emptyset$;                                   (∗ initialize the set of rules ∗)
  forall $f \in F$ do begin                           (∗ traverse the frequent item sets ∗)
    $m := 1$;                                         (∗ start with rule heads (consequents) ∗)
    $H_m := \bigcup_{i \in f}\{\{i\}\}$;              (∗ that contain only one item ∗)
    repeat                                            (∗ traverse rule heads of increasing size ∗)
      forall $h \in H_m$ do                           (∗ traverse the possible rule heads ∗)
        if $\frac{s_T(f)}{s_T(f-h)} \geq c_{\min}$    (∗ if the confidence is high enough, ∗)
        then $R := R \cup \{[(f-h) \to h]\}$;         (∗ add rule to the result ∗)
        else $H_m := H_m - \{h\}$;                    (∗ otherwise discard the head ∗)
      $H_{m+1} :=$ candidates($H_m$);                 (∗ create heads with one item more ∗)
      $m := m + 1$;                                   (∗ increment the head item counter ∗)
    until $H_m = \emptyset$ or $m \geq |f|$;          (∗ until there are no more rule heads ∗)
  end;                                                (∗ or antecedent would become empty ∗)
  return $R$;                                         (∗ return the rules found ∗)
end; (∗ rules ∗)
```

Generating Association Rules

```
function candidates ($F_k$)                         (* generate candidates with $k+1$ items *)
begin
    $E := \emptyset$;                                (* initialize the set of candidates *)
    forall $f_1, f_2 \in F_k$                        (* traverse all pairs of frequent item sets *)
    with  $f_1 = \{a_1, \ldots, a_{k-1}, a_k\}$      (* that differ only in one item and *)
    and   $f_2 = \{a_1, \ldots, a_{k-1}, a'_k\}$     (* are in a lexicographic order *)
    and   $a_k < a'_k$ do begin                      (* (the order is arbitrary, but fixed) *)
        $f := f_1 \cup f_2 = \{a_1, \ldots, a_{k-1}, a_k, a'_k\}$;   (* union has $k+1$ items *)
        if $\forall a \in f: \; f - \{a\} \in F_k$   (* only if all subsets are frequent, *)
        then $E := E \cup \{f\}$;                    (* add the new item set to the candidates *)
    end;                                             (* (otherwise it cannot be frequent) *)
    return $E$;                                      (* return the generated candidates *)
end (* candidates *)
```

Frequent Item Sets: Example

transaction vector

1:	$\{a, d, e\}$
2:	$\{b, c, d\}$
3:	$\{a, c, e\}$
4:	$\{a, c, d, e\}$
5:	$\{a, e\}$
6:	$\{a, c, d\}$
7:	$\{b, c\}$
8:	$\{a, c, d, e\}$
9:	$\{c, b, e\}$
10:	$\{a, d, e\}$

frequent item sets

0 items	1 item	2 items	3 items
$\emptyset$: 100%	$\{a\}$: 70%	$\{a, c\}$: 40%	$\{a, c, d\}$: 30%
	$\{b\}$: 30%	$\{a, d\}$: 50%	$\{a, c, e\}$: 30%
	$\{c\}$: 70%	$\{a, e\}$: 60%	$\{a, d, e\}$: 40%
	$\{d\}$: 60%	$\{b, c\}$: 30%	
	$\{e\}$: 70%	$\{c, d\}$: 40%	
		$\{c, e\}$: 40%	
		$\{d, e\}$: 40%	

- The minimum support is $s_{\min} = 3$ or $\sigma_{\min} = 0.3 = 30\%$ in this example.
- There are $2^5 = 32$ possible item sets over $A = \{a, b, c, d, e\}$.
- There are 16 frequent item sets (but only 10 transactions).

Generating Association Rules

Example: $I = \{a, c, e\}$, $X = \{c, e\}$, $Y = \{a\}$.

$$c_T(c, e \to a) = \frac{s_T(\{a, c, e\})}{s_T(\{c, e\})} = \frac{30\%}{40\%} = 75\%$$

Minimum confidence: 80%

association rule	support of all items	support of antecedent	confidence
$b \to c$:	30%	30%	100%
$d \to a$:	50%	60%	83.3%
$e \to a$:	60%	70%	85.7%
$a \to e$:	60%	70%	85.7%
$d, e \to a$:	40%	40%	100%
$a, d \to e$:	40%	50%	80%

Support of an Association Rule

The two rule support definitions are not equivalent:

transaction vector

1: $\{a, c, e\}$
2: $\{b, d\}$
3: $\{b, c, d\}$
4: $\{a, e\}$
5: $\{a, b, c, d\}$
6: $\{c, e\}$
7: $\{a, b, d\}$
8: $\{a, c, d\}$

two association rules

association rule	support of all items	support of antecedent	confidence
$a \rightarrow c$	3 (37.5%)	5 (62.5%)	67.7%
$b \rightarrow d$	4 (50.0%)	4 (50.0%)	100.0%

Let the minimum confidence be $c_{\text{min}} = 65\%$.

- For $\varsigma_T(R) = \sigma(X \cup Y)$ and $3 < \varsigma_{\text{min}} \leq 4$ only the rule $b \rightarrow d$ is generated, but not the rule $a \rightarrow c$.

- For $\varsigma_T(R) = \sigma(X)$ there is no value ς_{min} that generates only the rule $b \rightarrow d$, but not at the same time also the rule $a \rightarrow c$.

Additional Rule Filtering

Simple Measures

- General idea: Compare $\hat{p}_T(Y \mid X) = c_T(X \to Y)$
  and $\hat{p}_T(Y) = c_T(\emptyset \to Y) = \sigma_T(Y)$.

- (Absolute) confidence difference to prior:

$$d_T(R) = |c_T(X \to Y) - \sigma_T(Y)|$$

- (Absolute) difference of confidence quotient to 1:

$$q_T(R) = \left| 1 - \min\left\{ \frac{c_T(X \to Y)}{\sigma_T(Y)}, \frac{\sigma_T(Y)}{c_T(X \to Y)} \right\} \right|$$

- Confidence to prior ratio (lift):

$$l_T(R) = \frac{c_T(X \to Y)}{\sigma_T(Y)}$$

Additional Rule Filtering

More Sophisticated Measures

- Consider the 2×2 contingency table or the estimated probability table:

	$X \not\subseteq t$	$X \subseteq t$	
$Y \not\subseteq t$	n_{00}	n_{01}	$n_{0.}$
$Y \subseteq t$	n_{10}	n_{11}	$n_{1.}$
	$n_{.0}$	$n_{.1}$	$n_{..}$

	$X \not\subseteq t$	$X \subseteq t$	
$Y \not\subseteq t$	p_{00}	p_{01}	$p_{0.}$
$Y \subseteq t$	p_{10}	p_{11}	$p_{1.}$
	$p_{.0}$	$p_{.1}$	1

- $n_{..}$ is the total number of transactions.
  $n_{1.}$ is the number of transactions to which the rule is applicable.
  n_{11} is the number of transactions for which the rule is correct.
  It is $\quad p_{ij} = \frac{n_{ij}}{n_{..}}, \quad p_{i.} = \frac{n_{i.}}{n_{..}}, \quad p_{.j} = \frac{n_{.j}}{n_{..}} \quad$ for $i, j = 1, 2$.

- General idea: Use measures for the strength of dependence of X and Y.

An Information-theoretic Evaluation Measure

Information Gain (Kullback and Leibler 1951, Quinlan 1986)

Based on Shannon Entropy $H = -\sum_{i=1}^{n} p_i \log_2 p_i$ (Shannon 1948)

$$I_{\text{gain}}(X,Y) = \overbrace{-\sum_{i=1}^{k_Y} p_{i.} \log_2 p_{i.}}^{H(Y)} - \overbrace{\sum_{j=1}^{k_X} p_{.j} \left(-\sum_{i=1}^{k_Y} p_{i|j} \log_2 p_{i|j} \right)}^{H(Y|X)}$$

$H(Y)$	Entropy of the distribution of Y
$H(Y\|X)$	*Expected entropy* of the distribution of Y if the value of the X becomes known
$H(Y) - H(Y\|X)$	Expected entropy reduction or *information gain*

A Statistical Evaluation Measure

χ^2 Measure

- Compares the actual joint distribution
  with a **hypothetical independent distribution**.
- Uses absolute comparison.
- Can be interpreted as a difference measure.

$$\chi^2(X,Y) = \sum_{i=1}^{k_X} \sum_{j=1}^{k_Y} n_{..} \frac{(p_{i.}p_{.j} - p_{ij})^2}{p_{i.}p_{.j}}$$

- Side remark: Information gain can also be interpreted as a difference measure.

$$I_{\text{gain}}(X,Y) = \sum_{j=1}^{k_X} \sum_{i=1}^{k_Y} p_{ij} \log_2 \frac{p_{ij}}{p_{i.}p_{.j}}$$

A Statistical Evaluation Measure

χ^2 Measure

- Compares the actual joint distribution with a **hypothetical independent distribution**.
- Uses absolute comparison.
- Can be interpreted as a difference measure.

$$\chi^2(X,Y) = \sum_{i=1}^{k_X} \sum_{j=1}^{k_Y} n_{..} \frac{(p_{i.}p_{.j} - p_{ij})^2}{p_{i.}p_{.j}}$$

For $k_X = k_Y = 2$ (as for rule evaluation) the χ^2 measure simplifies to

$$\chi^2(X,Y) \;=\; n_{..}\frac{(p_{1.}\,p_{.1} - p_{11})^2}{p_{1.}(1-p_{1.})p_{.1}(1-p_{.1})} \;=\; n_{..}\frac{(n_{1.}n_{.1} - n_{..}n_{11})^2}{n_{1.}(n_{..}-n_{1.})n_{.1}(n_{..}-n_{.1})}.$$

Summary Association Rules

- **Association Rule Induction is a Two Step Process**
  - Find the frequent item sets (minimum support).
  - Form the relevant association rules (minimum confidence).
- **Generating the Association Rules**
  - Form all possible association rules from the frequent item sets.
  - Filter "interesting" association rules
    based on minimum support and minimum confidence.
- **Filtering the Association Rules**
  - Compare rule confidence and consequent support.
  - Information gain
  - χ^2 measure

Industrial Applications

- Car manufacturer collects servicing tasks on all their vehicles.
  - What are interesting subgroups of cars?
  - How do these subgroups behave over time?
  - Which cars' suspension failure rate is strongly increasing in winter?
- Bank assesses credit contracts w. r. t. terminability.
  - What changes were there in the past?
  - Any common factors?
  - How do I communicate this to a non-statistician?
- Tracking user activity in a virtual environment
  - Are there any oddities in user behavior?
  - How do I parameterize "odd" things?

Or: What they have and what they want

Given:

- High-dimensional data
- Many-valued data
- Time-stamped data

Asked for:

- Easy-to-understand patterns (rules)
- Exploratory tools (visualization and inspection)
- Natural way of interaction
- Exploit temporal information (if desired)

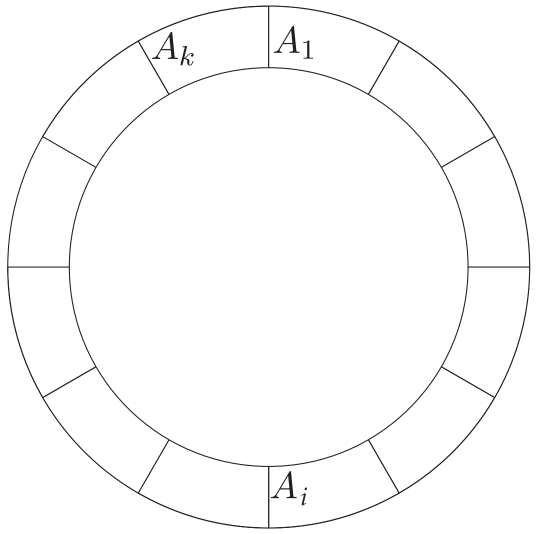
Rule Icons

- Every rule

$$\langle A_1 = a_1 \wedge \cdots \wedge A_k = a_k \rangle \rightarrow C = c$$

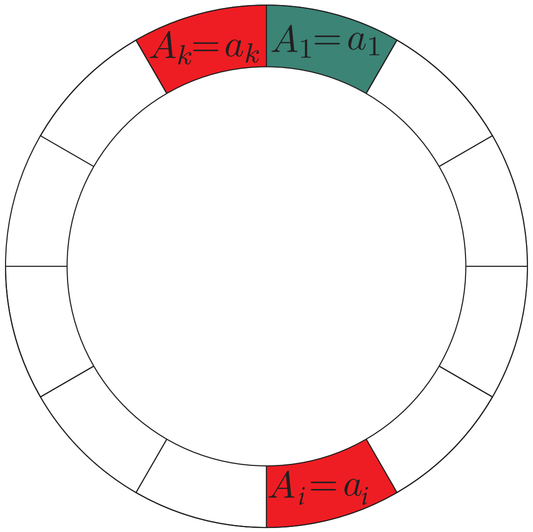
of a given rule set is represented as an icon:

  - For every possible item there is a reserved segment on the outer border.



Rule Icons

- Every rule
$$\langle A_1 = a_1 \wedge \cdots \wedge A_k = a_k \rangle \rightarrow C = c$$
of a given rule set is represented as an icon:
  - For every possible item there is a reserved segment on the outer border.
  - If the item is present in the antecedent, the segment is colored.



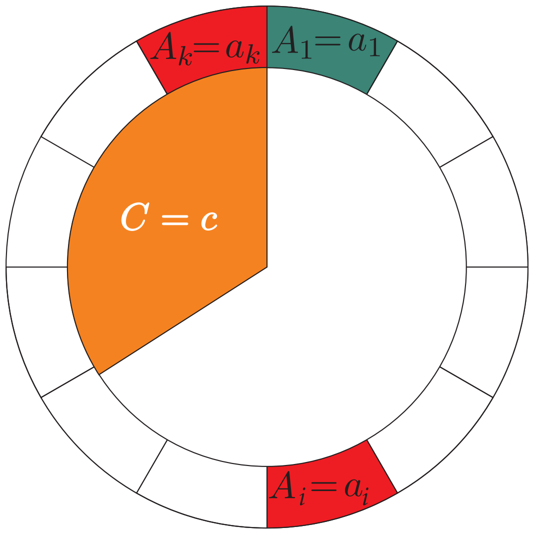
Rule Icons

- Every rule

$$\langle A_1 = a_1 \wedge \cdots \wedge A_k = a_k \rangle \to C = c$$

of a given rule set is represented as an icon:

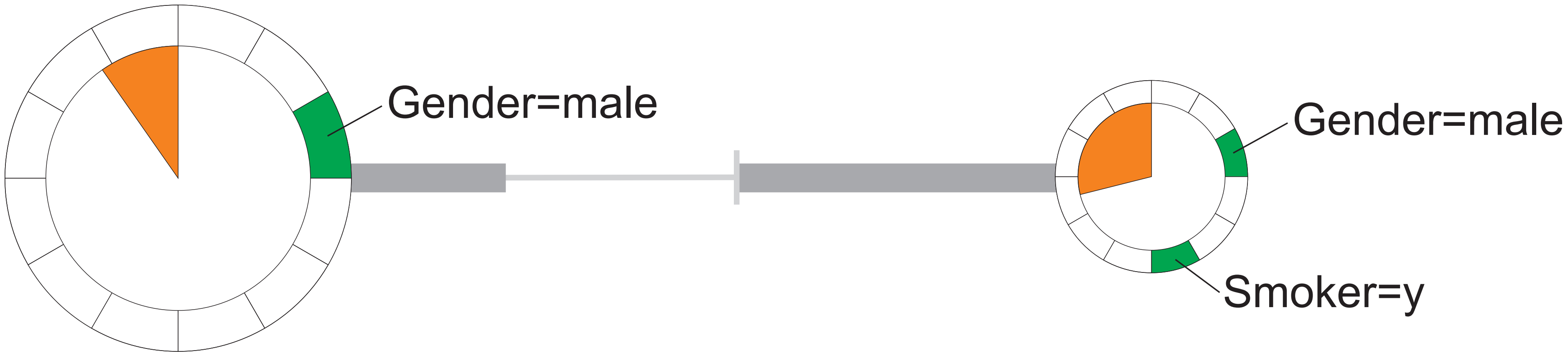
  - For every possible item there is a reserved segment on the outer border.
  - If the item is present in the antecedent, the segment is colored.
  - Interior encodes a rule-measure: here confidence.



Rule Icons: Overlapping

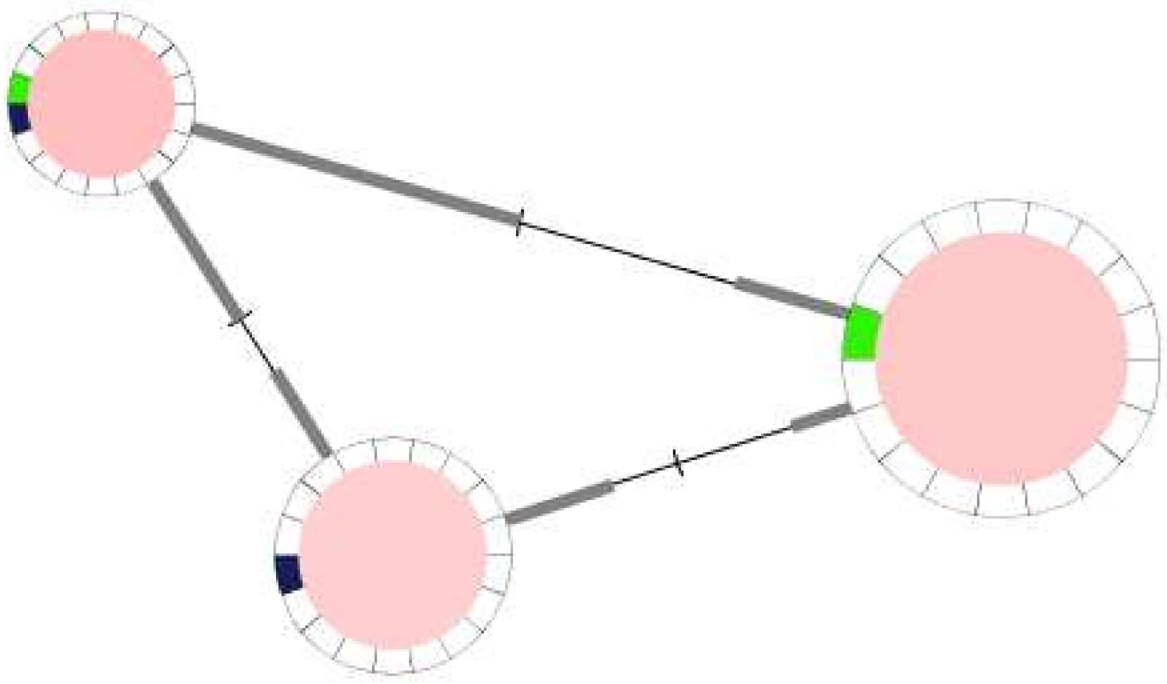
- The cover of two rules may be non-empty. Use a percentage bar to display the mutual overlap.
- Special case: inclusion

$$\begin{aligned} \text{Gender} = \text{male} &\rightarrow \text{Cancer} = \text{yes} \\ \text{Gender} = \text{male} \wedge \text{Smoker} = \text{yes} &\rightarrow \text{Cancer} = \text{yes} \end{aligned}$$



Rule Icons: Overlapping

- The cover of two rules may be non-empty. Use a percentage bar to display the mutual overlap.
- General case:



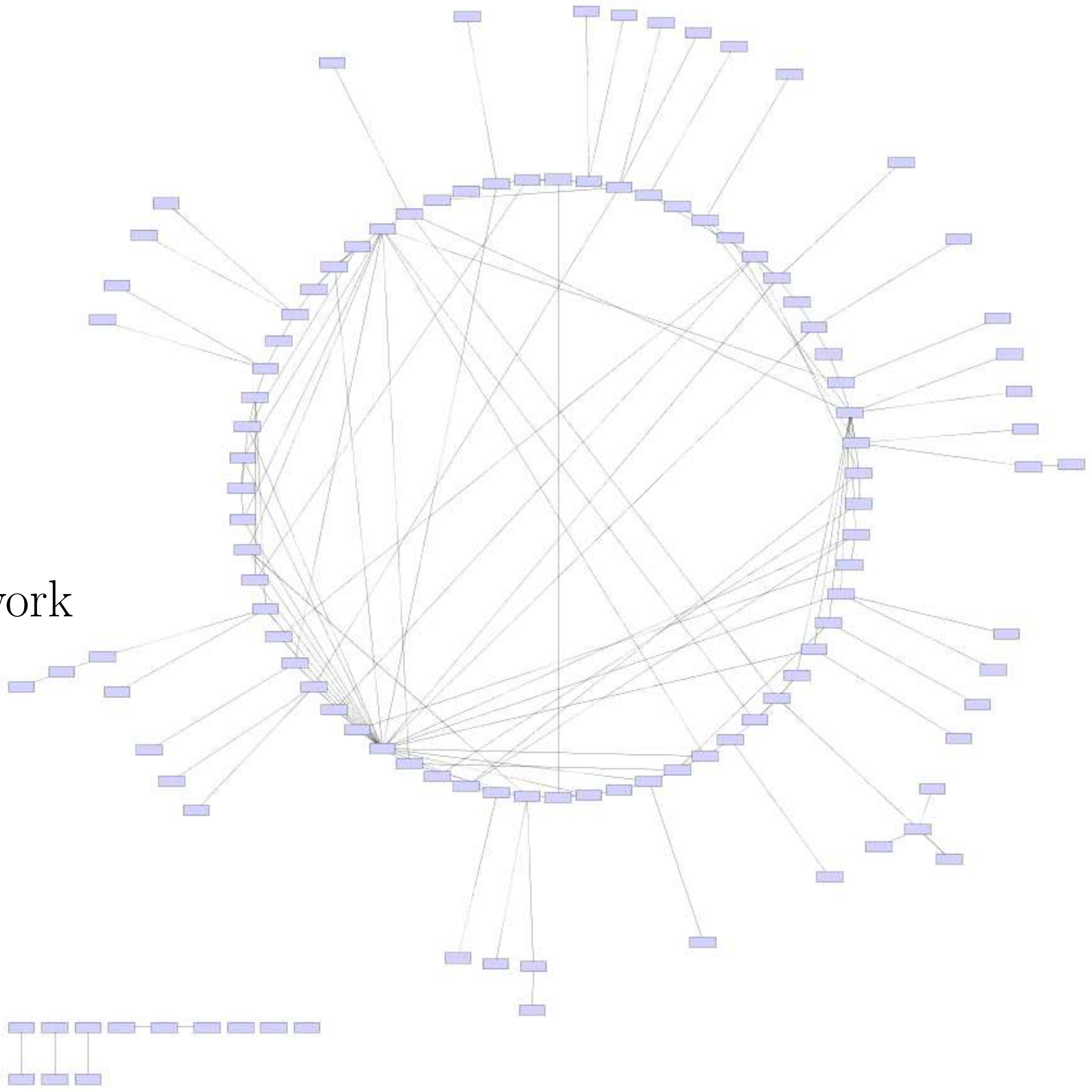
Rule Icons: Location

- Finally, arrange the icons in a two-dimensional chart.
- Choose association rule measures for the two axes and the size of the icon
- Our suggestion: For a rule $X \to Y$ choose the following measures:
  - x-coordinate: recall, i. e. $c_T(Y \to X)$
  - y-coordinate: lift, i. e. $c_T(X \to Y)/\sigma_T(Y)$
  - size: support, i. e. $\sigma_T(X \cup Y)$

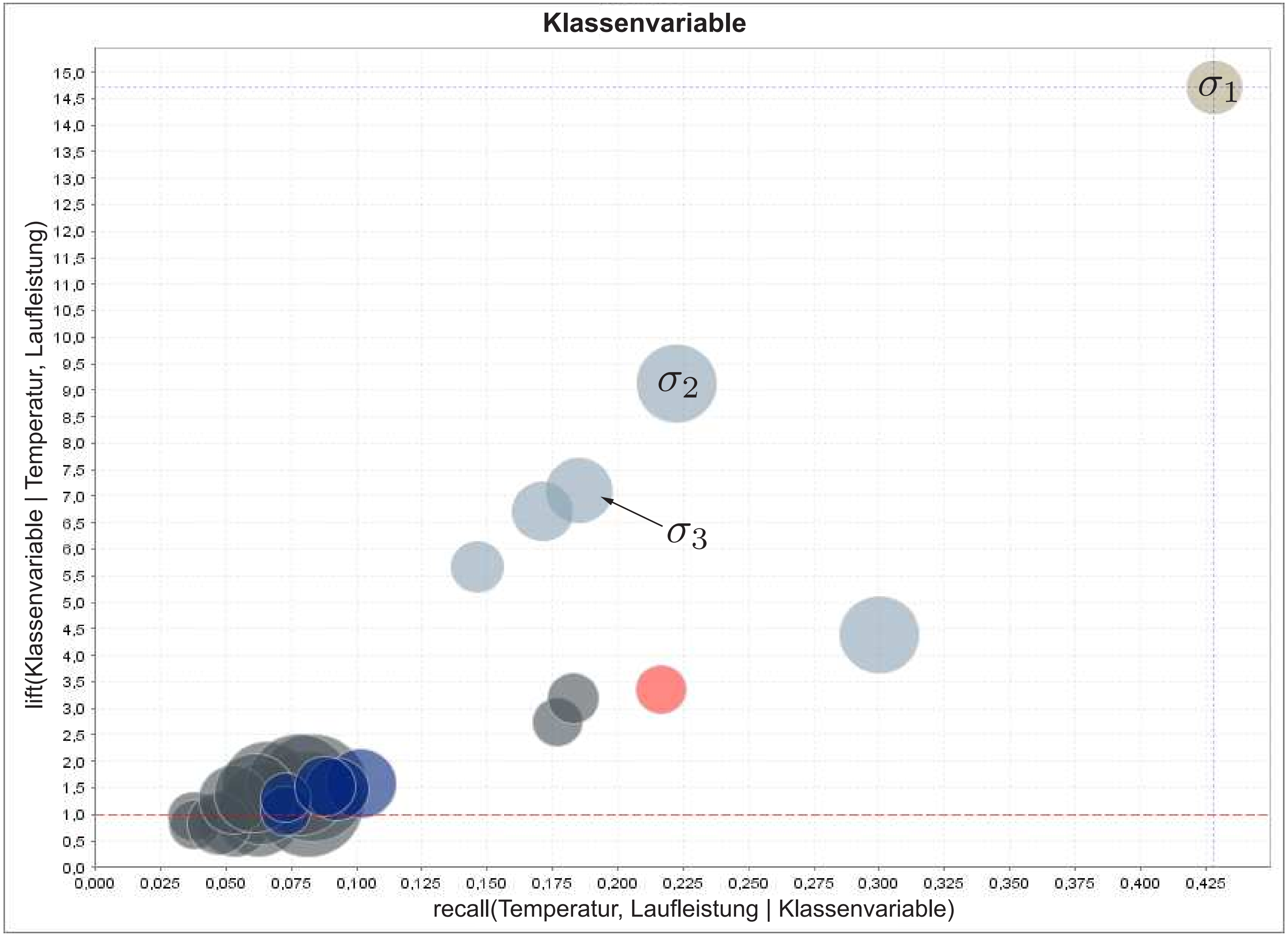
Real-world Example: Daimler AG

Car database

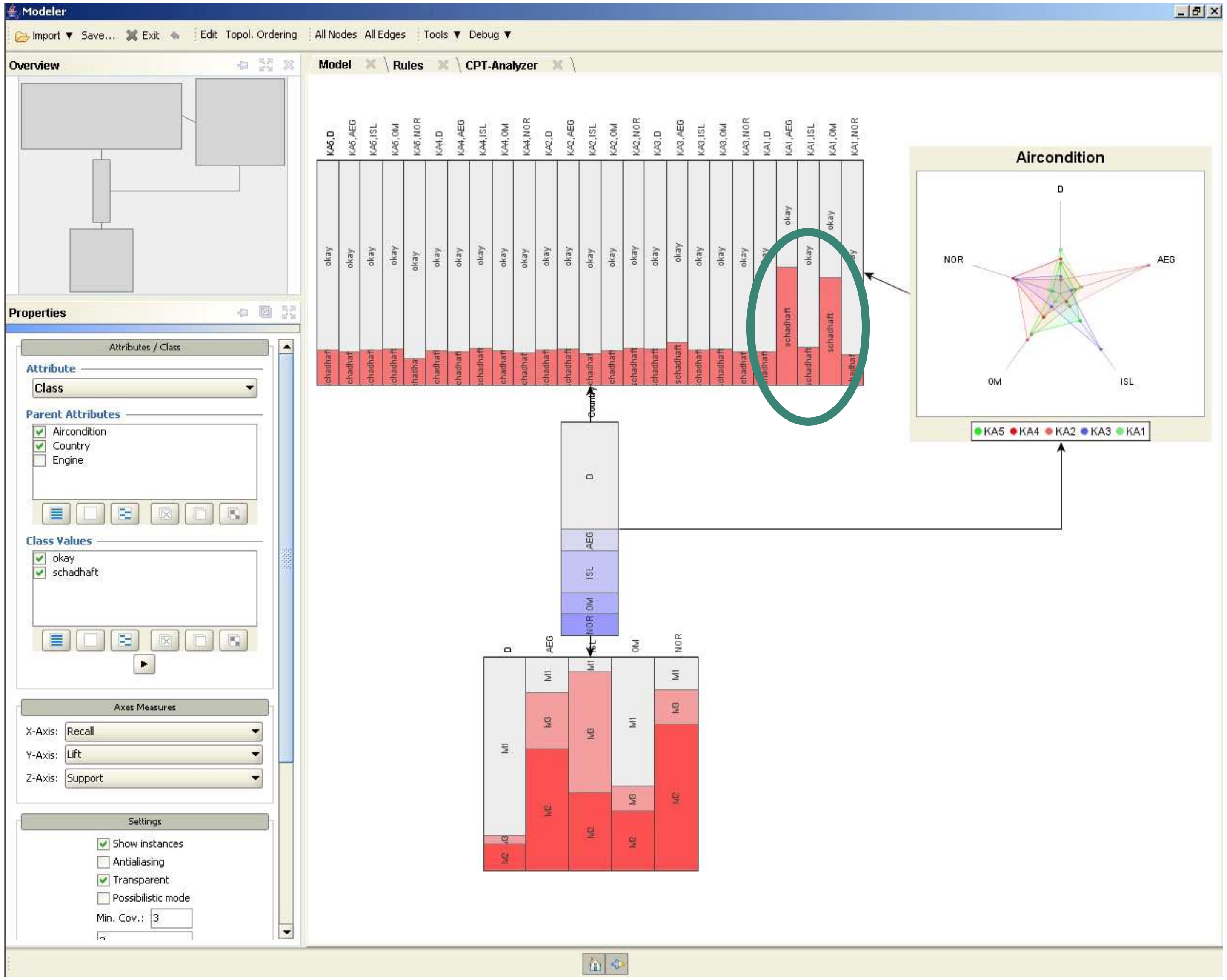
- 300 000 vehicles
- subset of 180 attributes
- 2 to 300 values per attribute
- Probabilistic Dependency Network



Real-world Example: Daimler AG



Explorative Analysis Tool



Attribute: Class

Parent Attributes:
- [✓] Aircondition
- [✓] Country
- [ ] Engine

Class Values:
- [✓] okay
- [✓] schadhaft

X-Axis: Recall
Y-Axis: Lift
Z-Axis: Support

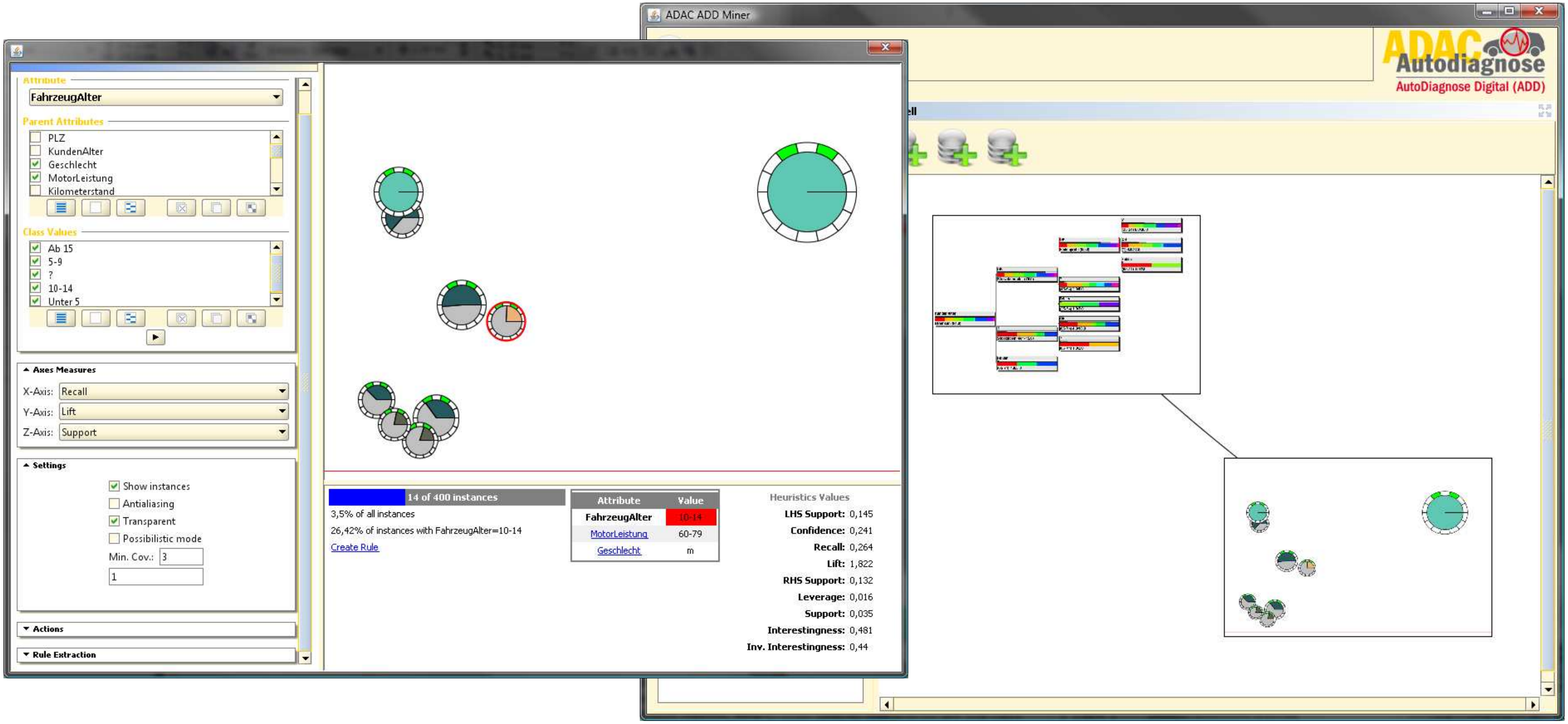
- [✓] Show instances
- [ ] Antialiasing
- [✓] Transparent
- [ ] Possibilistic mode

Min. Cov.: 3

Real-world Example: ADAC

Customer database

- Car and customer information
- Assessment of vehicle quality



Attribute: FahrzeugAlter

Parent Attributes:
- [ ] PLZ
- [ ] KundenAlter
- [✓] Geschlecht
- [✓] MotorLeistung
- [ ] Kilometerstand

Class Values:
- [✓] Ab 15
- [✓] 5-9
- [✓] ?
- [✓] 10-14
- [✓] Unter 5

Axes Measures
- X-Axis: Recall
- Y-Axis: Lift
- Z-Axis: Support

Settings
- [✓] Show instances
- [ ] Antialiasing
- [✓] Transparent
- [ ] Possibilistic mode
- Min. Cov.: 3
- 1

14 of 400 instances

3,5% of all instances

26,42% of instances with FahrzeugAlter=10-14

Create Rule

Attribute	Value
FahrzeugAlter	10-14
MotorLeistung	60-79
Geschlecht	m

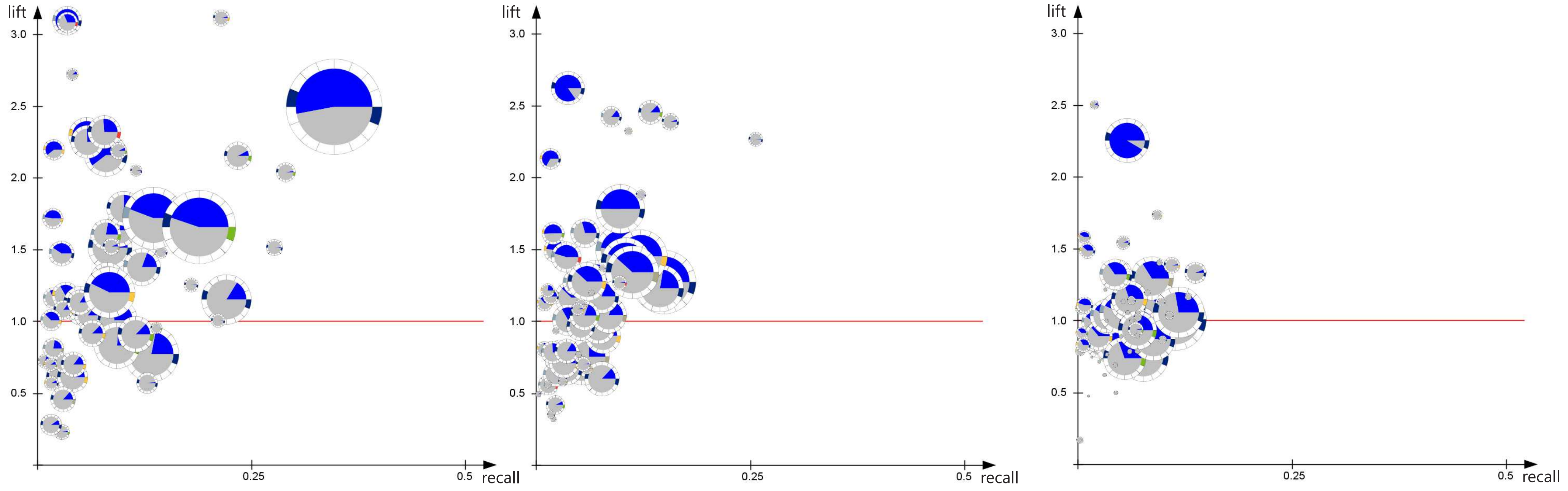
Heuristics Values

LHS Support:	0,145
Confidence:	0,241
Recall:	0,264
Lift:	1,822
RHS Support:	0,132
Leverage:	0,016
Support:	0,035
Interestingness:	0,481
Inv. Interestingness:	0,44

Temporal Change of Rules

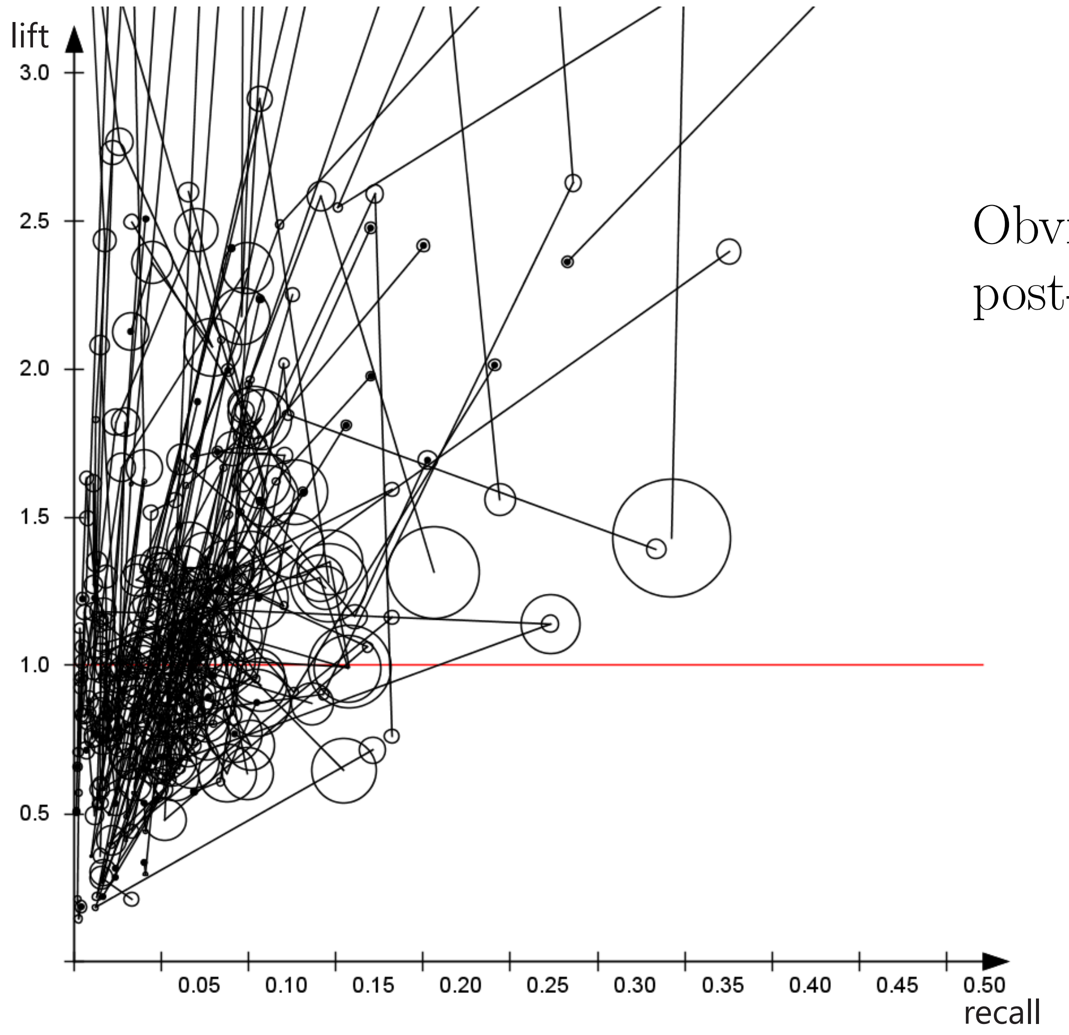
- **Why considering the temporal development of rules?** (i. e. the change of certain rule evaluation measures)
    - Failure patterns usually do not arise out of a sudden but rather evolve slowly over time.
    - A fixed problem takes some while to have a measurable effect.
- **How to present this evolution to the user?**
    - Create a time series for every measure used for locating and scaling the rule icon.
    - Interpolate between the frames and present an animation.
- **Problem:** Need to reduce the number of rules.

How does that look like?



Real-world dataset

How does that look like?



Obviously, there is a demand for post-processing the rule set!

Temporal Change of Rules

- Divide dataset into reasonable time frames
- Run respective pattern induction algorithm
- Quantify each pattern w. r. t. any desired measure(s)
- Generate time series for each measure and each pattern
- **Match the time series against a user-specified concept**
- Rank them according to the membership of the concept

User-driven Post-processing

Users often have an idea in which direction to investigate but cannot explicitly phrase a query to a Data Mining system. However, we can use intentions like

> "Show me only those rules that had a strong increasing support and a increasing confidence in the last quarter."

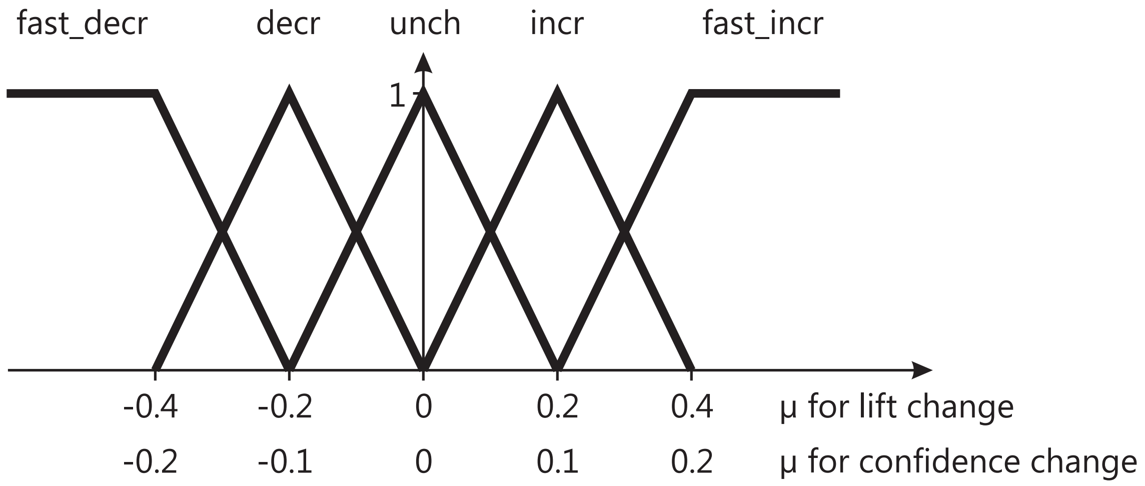
or

> "Which patterns exhibit an increasing lift while the support was stable or at most slightly decreasing?"

to thin out the rule set.

User-driven Post-processing

1. Specify a fuzzy partition on the change rate domain of every pattern evaluation measure.



User-driven Post-processing

1. Specify a fuzzy partition on the change rate domain of every pattern evaluation measure.
2. Encode the user-concept as a fuzzy antecedent.

E. g. "lift is unchanged and confidence is increasing":

$$\langle \Delta_{\text{lift}} \, \texttt{is} \text{ unch} \;\wedge\; \Delta_{\text{conf}} \, \texttt{is} \text{ incr} \rangle$$

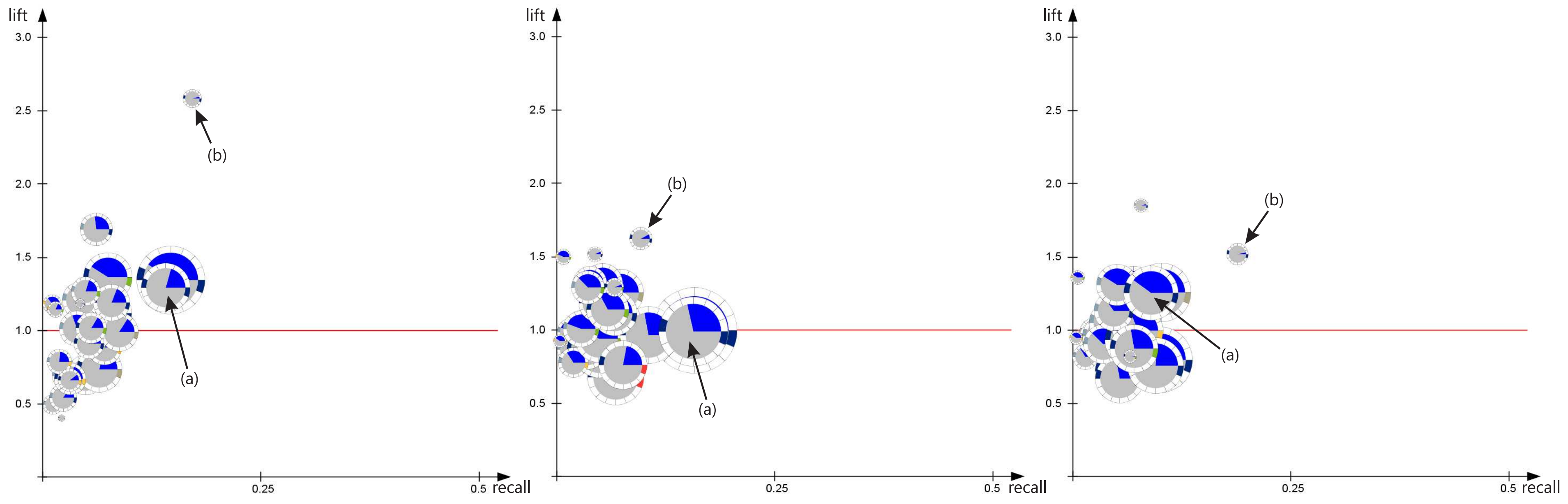
will be evaluated as

$$\top\left(\mu_{\Delta_{\text{lift}}}^{(\text{unch})}(\vec{a} \to c), \mu_{\Delta_{\text{conf}}}^{(\text{incr})}(\vec{a} \to c)\right)$$

where $\top$ is a t-norm that represents a fuzzy conjunction.

User-driven Post-processing

1. Specify a fuzzy partition on the change rate domain of every pattern evaluation measure.
2. Encode the user-concept as a fuzzy antecedent.
3. Order the patterns w. r. t. concept membership degrees.



Summary Industrial Applications

Requirements

- Easy-to-understand patterns
- Exploratory visual tools
- Natural and intuitive interaction
- Exploitation of temporal information

Desired Properties of Rules

- Almost free of parameters (support and confidence have a clear notion and can even be increased after the induction)
- No black box approach
- Intuitive type of patterns (decision rules, business rules)
- Natural way of treating missing values.
- Light data preprocessing overhead

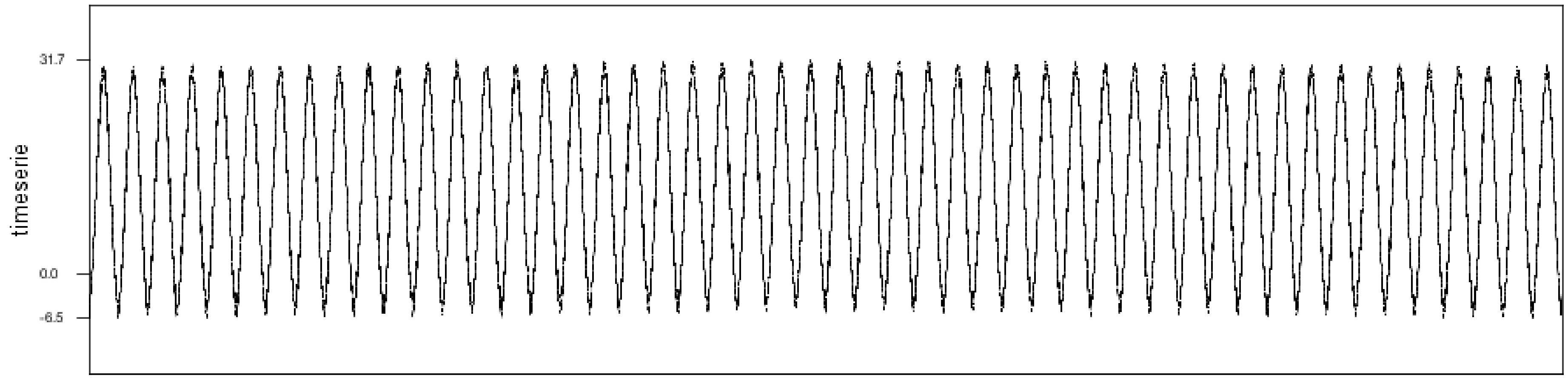
Time Series Analysis

Time Series

- **Motivation**
- **Decomposition Models**
    - Additive models, multiplicative models
- **Global Approaches**
    - Regression
    - With and without seasonal component
- **Local Approaches**
    - Moving Averages Smoothing
    - With and without seasonal component
- **Summary**

Motivation: Temperatures

Example: Temperatures data set (fictive)



- The plot shows the average temperature per day for 50 years.
- Is there any trend visible?
- How to extract seasonal effects?

Decomposition Models

- The time series is given as a sequence of values

$$y_1, \ldots, y_t, \ldots, y_n$$

- We assume that every y_t is a composition of (some of) the following components:
    - g_t trend component
    - s_t seasonal component
    - c_t cyclical variation
    - ϵ_t irregular component (random factors, noise)

- Assume a functional dependency:

$$y_t = f(g_t, s_t, c_t, \epsilon_t)$$

Components of Time Series

Trend Component

- Reflects long-term developments.
- Often assumed to be a monotone function of time.
- Represents the actual component we are interested in.

Cyclic Component

- Reflects mid-term developments.
- Models economical cycles such as booms and recessions.
- Variable cycle length.
- We do not consider this component here.

Remark: Often, both components are combined.

Components of Time Series

Seasonal Component

- Reflects short-term developments.
- Constant cycle length (i. e., 12 months)
- Represents changes that (re)occur rather regularly.

Irregular Component

- Represents everything else that cannot be related to the other components.
- Combines irregular changes, random noise and local fluctuations.
- We assume that the values are small and have an average of zero.

Decompositions

Additive Decomposition

$$y_t = g_t + s_t + \epsilon_t$$

- Pure trend model: $y_t = g_t + \epsilon_t$ (stock market, no season)
- Possible extension: $y_t = g_t + s_t + x_t\beta + \epsilon_t$ (calendar effects)

Multiplicative Decomposition

$$y_t = g_t \cdot s_t \cdot \epsilon_t$$

- Seasonal changes may increase with trend.
- Transform into additive model:

$$\tilde{y}_t = \log y_t + \log s_t + \log \epsilon_t$$

Time Series Analysis

Goal: Estimate the components from a given time series, i. e.

$$\hat{g}_t + \hat{s}_t + \epsilon_t \;\approx\; y_t$$

Application: With the estimates, we can compute the

- trend-adjusted series: $y_t - \hat{g}_t$
- season-adjusted series: $y_t - \hat{s}_t$
- We only consider additive models here.

$\Rightarrow$ Additional assumptions necessary in order to find ways to infer the desired components.

Overview

- **Global approach:** There is a fix functional dependence throughout the entire time range. ($\Rightarrow$ regression models)
- **Local approach:** We do not postulate a global model and rather use local estimations to describe the respective components.
- **Seasonal effects:** We have to decide beforehand whether to assume a seasonal component or not.

	Global	Local
without Season	Regression	Smoothing Averages
with Season	Dummy Variables	Smoothing Averages

Global Approach (without Season)

Model: $y_t = g_t + \epsilon_t$

Assumptions:

- No seasonal component: $s_t = 0$
- Depending on g_t, use regression analysis to estimate the parameter(s) to define the trend component.
  - linear trend: $g_t = \beta_0 + \beta_1 t$
  - quadratic trend: $g_t = \beta_0 + \beta_1 t + \beta_2 t^2$
  - polynomial trend: $g_t = \beta_0 + \beta_1 t + \cdots + \beta_q t^q$
  - exponential trend: $g_t = \beta_0 \exp(\beta_1 t)$
  - logistic trend: $g_t = \frac{\beta_0}{\beta_1 + \exp(-\beta_2 t)}$

Global Approach (with Season)

Model: $y_t = s_t + \epsilon_t$ (no trend)

Assumptions:

- No trend component: $g_t = 0$
- Seasonal component does not change from period to period.
- Introduce *dummy variables* for every time span (here: months) that serve as indicator functions to determine to which month a specific t belongs:

$$s_m(t) = \begin{cases} 1, & \text{if } t \text{ belongs to month } m \\ 0, & \text{otherwise} \end{cases}$$

- The seasonal component is then set up as $s_t = \sum_{m=1}^{12} \beta_m s_m(t)$.
- Determine the *monthly effects* β_m with normal least squares method.

Global Approach (with Season)

Model: $y_t = g_t + s_t + \epsilon_t$

Assumptions:

- Estimate $\hat{g}_t$ while temporarily ignoring s_t.
- Estimate s_t from the trend-adjusted $\tilde{y}_t = y_t - \hat{g}_t$.

Model: $y_t = \alpha_1 t + \cdots + \alpha_q t^q + \cdots + \beta_1 s_1(t) + \cdots + \beta_{12} s_{12}(t) + \epsilon_t$

Assumptions:

- Seasonal component does not change from period to period.
- Model the seasonal effects with trigonometric functions:

$$s_t = \beta_0 + \sum_{m=1}^{6} \beta_m \cos\Big(2\pi \frac{m}{12} t\Big) + \sum_{m=1}^{5} \gamma_m \sin\Big(2\pi \frac{m}{12} t\Big)$$

- Determine $\alpha_1, \ldots, \alpha_q, \beta_0, \ldots, \beta_6$ and $\gamma_1, \ldots, \gamma_5$ with normal least squares method.

Local Approach (without Season)

General Idea: Smooth the time series.

- Estimate the trend component g_t at time t as the average of the values around time t.

For a given time series $y_1, \dots, y_n$, the **Smoothing Average** $y_t^\star$ of order r is defined as follows:

$$y_t^\star = \begin{cases} \dfrac{1}{2k+1} \cdot \displaystyle\sum_{j=-k}^{k} y_{t+j}, & \text{if } r = 2k+1 \\[2ex] \dfrac{1}{2k} \cdot \left(\dfrac{1}{2} y_{t-k} + \displaystyle\sum_{j=-k+1}^{k-1} y_{t+j} + \dfrac{1}{2} y_{t+k}\right), & \text{if } r = 2k \end{cases}$$

Local Approach (without Season)

Model: $y_t = g_t + \epsilon_t$

Assumptions:

- In every time frame of width $2k + 1$ the time series can be assumed to be linear.
- ϵ_t averages to zero.
- Then we use the smoothing average to estimate the trend component:

$$\hat{g}_t = y_t^\star$$

Local Approach (with Season)

Model: $y_t = g_t + s_t + \epsilon_t$

Assumptions:

- Seasonal component has period length p (repeats after p points):

$$s_t = s_{t+p}, \quad t = 1, \ldots, n - p$$

- Sum of seasonal values is zero: $\sum_{j=1}^{p} s_j = 0$

- Trend component is linear in time frames of width p (if p is odd) or $p + 1$ (if p is even).

- Irregular component averages to zero.

Local Approach (with Season)

Let $k = \frac{p-1}{2}$ (for odd p) or $k = \frac{p}{2}$ (for even p).

Then:

- Estimate the trend component with smoothing average:

$$\hat{g}_t = y_t^\star, \qquad k+1 \leq t \leq n-k$$

- Estimate the seasonal components $s_1, \ldots, s_p$ as follows:

$$\hat{s}_i = \tilde{s}_i - \frac{1}{p}\sum_{j=1}^{p} \tilde{s}_j \quad \text{with} \quad \tilde{s}_t \frac{1}{m_i - l_i + 1}\sum_{j=l}^{m_i}(y_{i+jp} - y_{i+jp}^\star), \qquad 1 \leq i \leq p$$

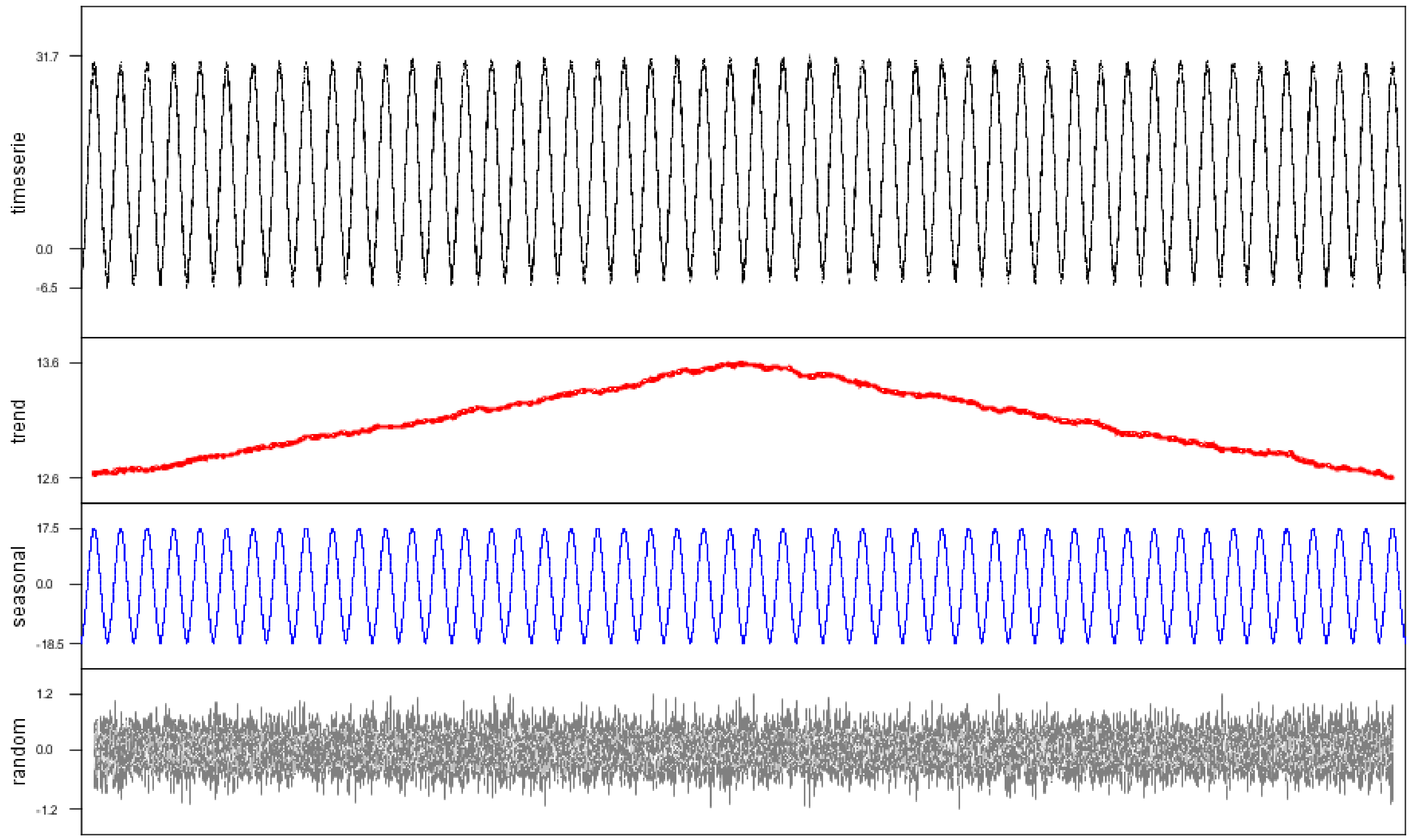
where

$$m_i = \max\{m \in \mathbb{N}_0 \mid i + mp \leq n - k\}$$

and

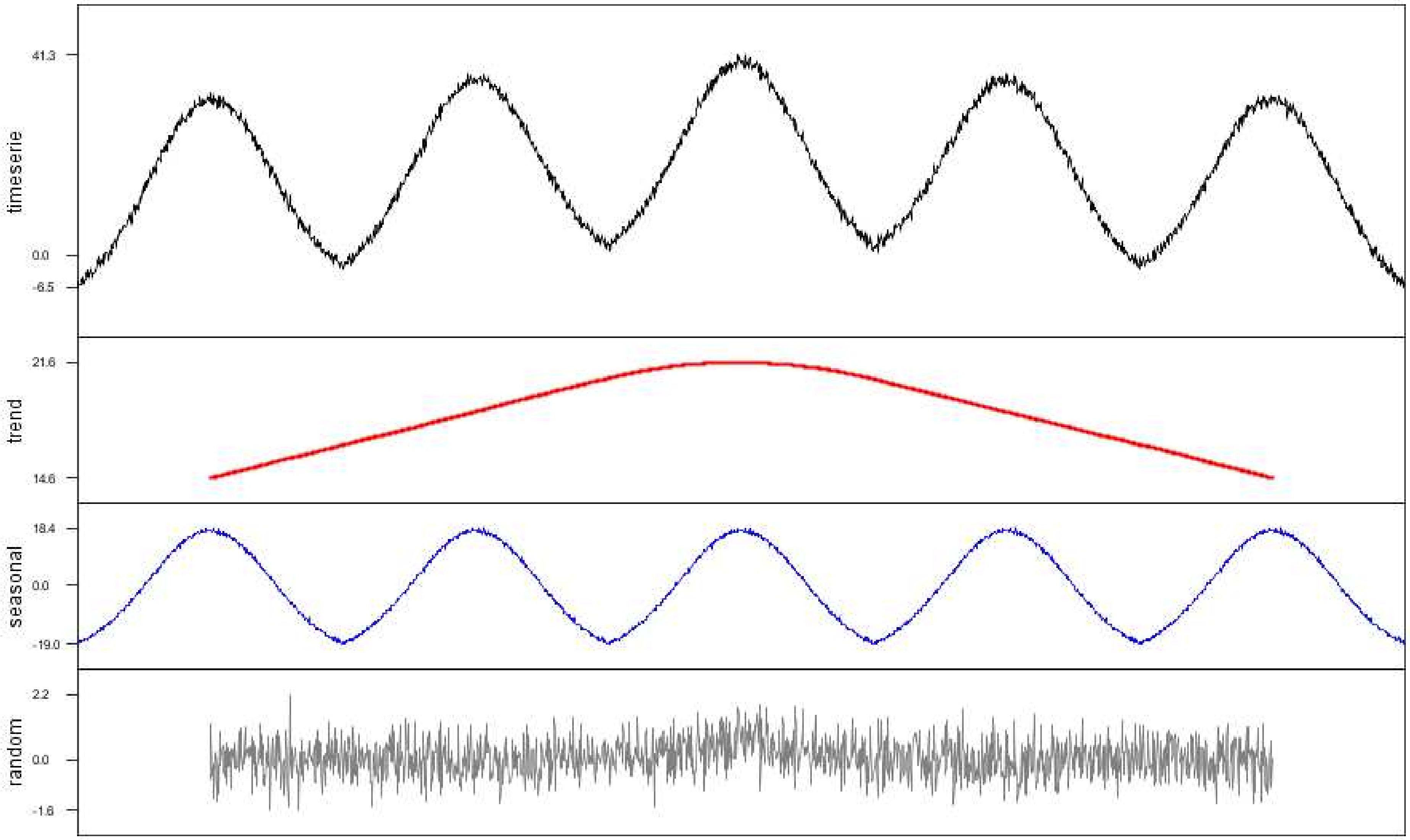
$$l_i = \min\{l \in \mathbb{N}_0 \mid i + lp \geq k + 1\}$$

Example (from motivation)



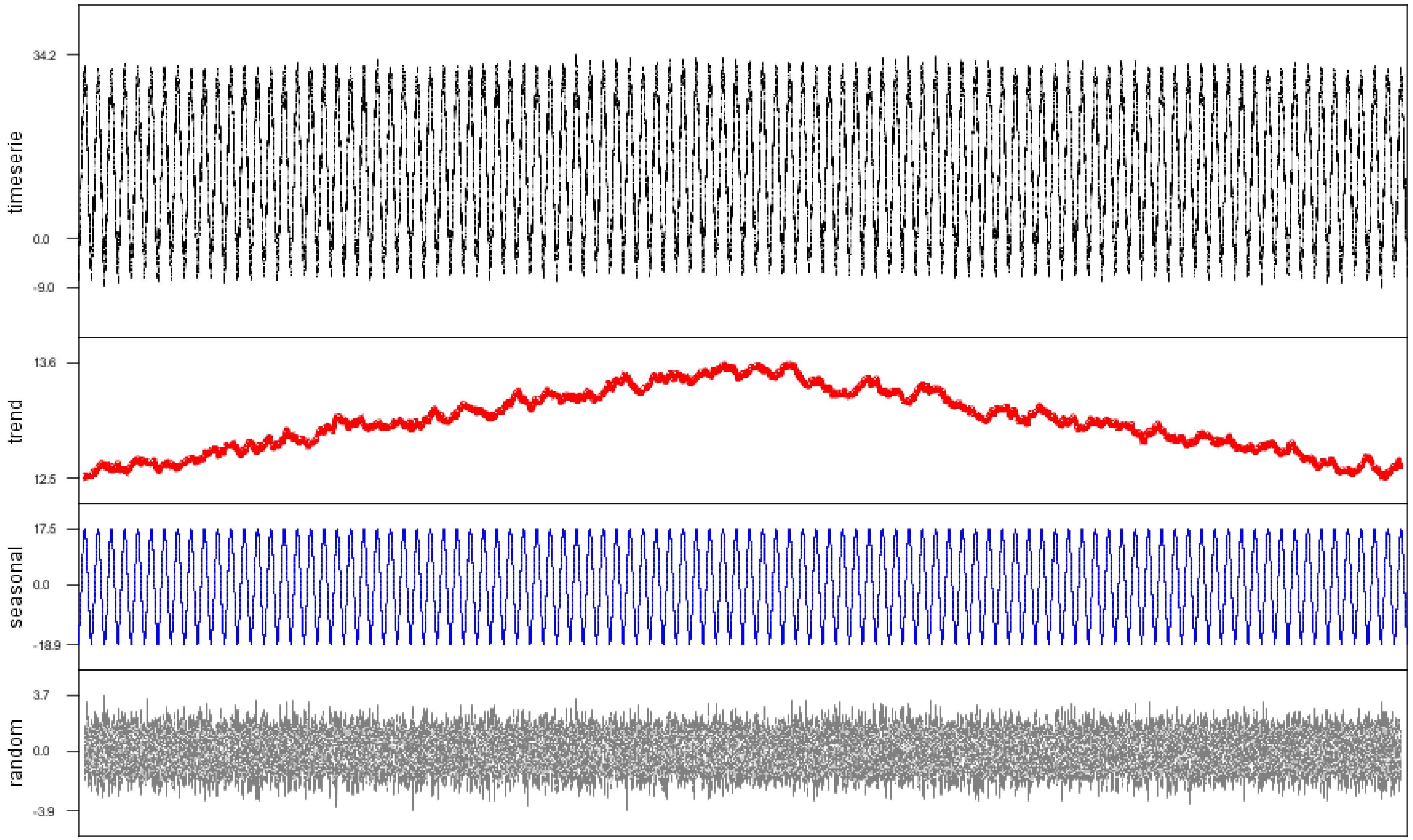
- We can extract an increase and decrease of 1 degree during 50 years even though the amount of noise is more than twice as large than the actual trend.

Example



5 years period, trend ±8 degrees, noise amount ±2 degrees

Example



100 years period, trend ±1 degree, noise amount ±3 degrees

Summary

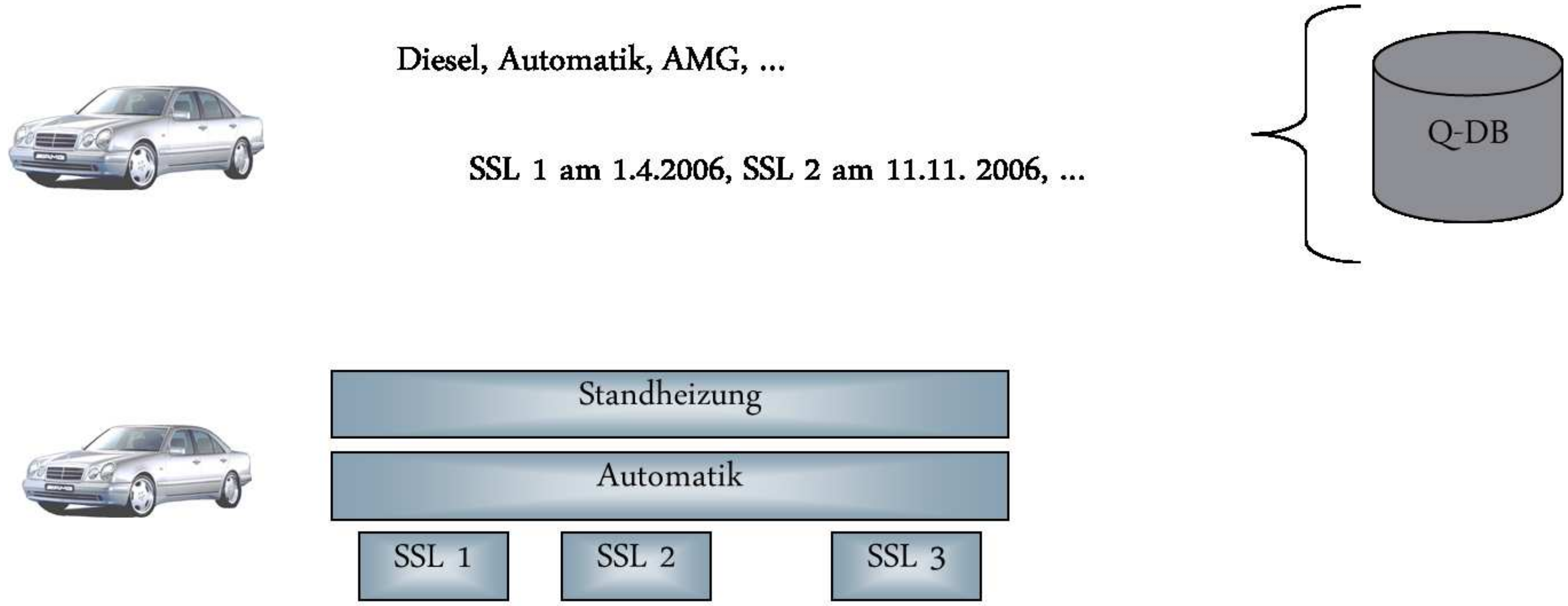
- **Definition of the problem domain**
  - Consider a time series to be composed of subcomponents.
  - Additive and multiplicative models.
- **Global and local approaches**
  - With and without seasonal components.
- **Robust to noise**
  - Noise can be higher than the trend component itself.

Häufige Muster in zeitbezogenen Daten

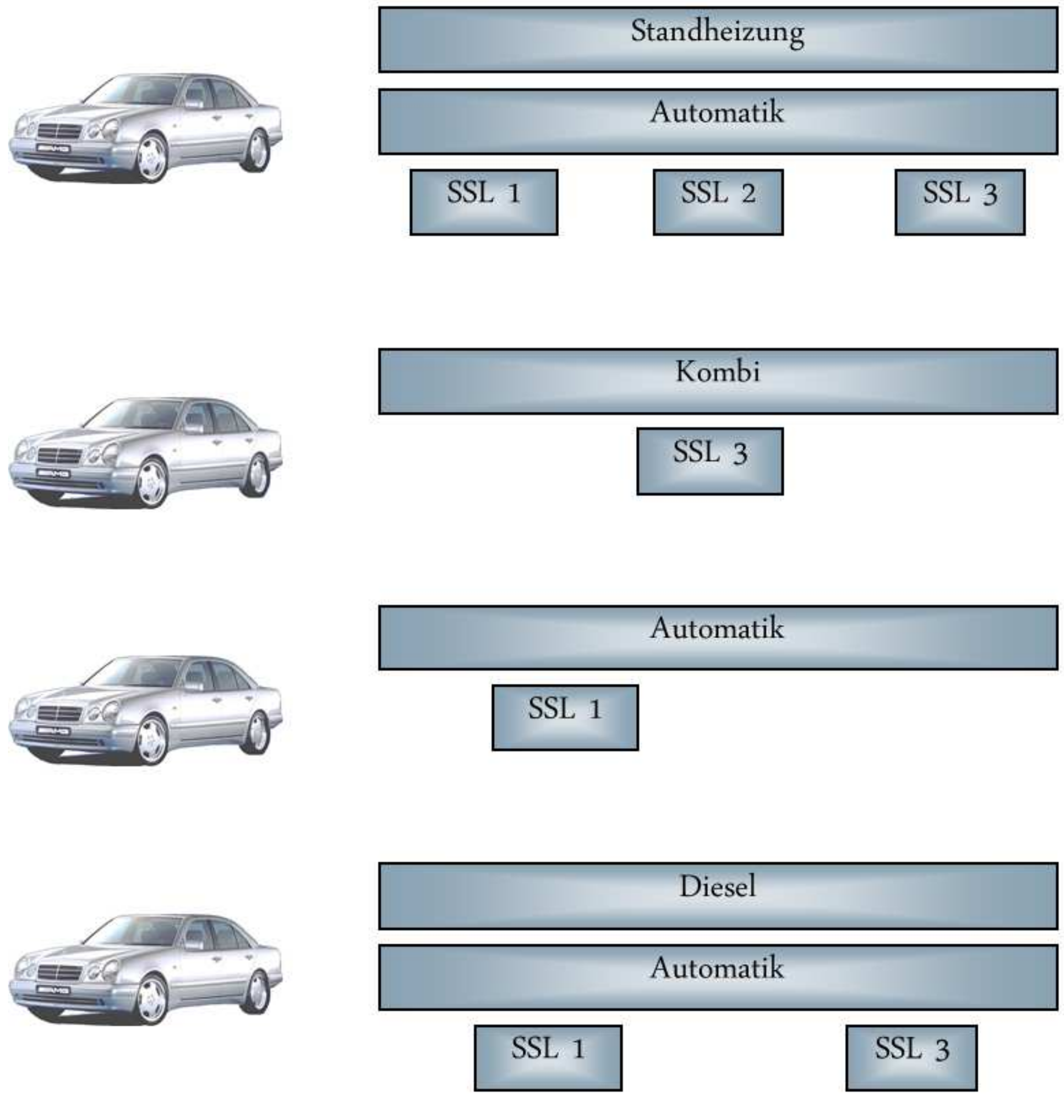
Inhalt

- Häufige Muster in zeitbezogenen Daten
    - Motivation / Problemstellung
    - Abgrenzung zu bekannten Verfahren
    - Algorithmen / Beispiel

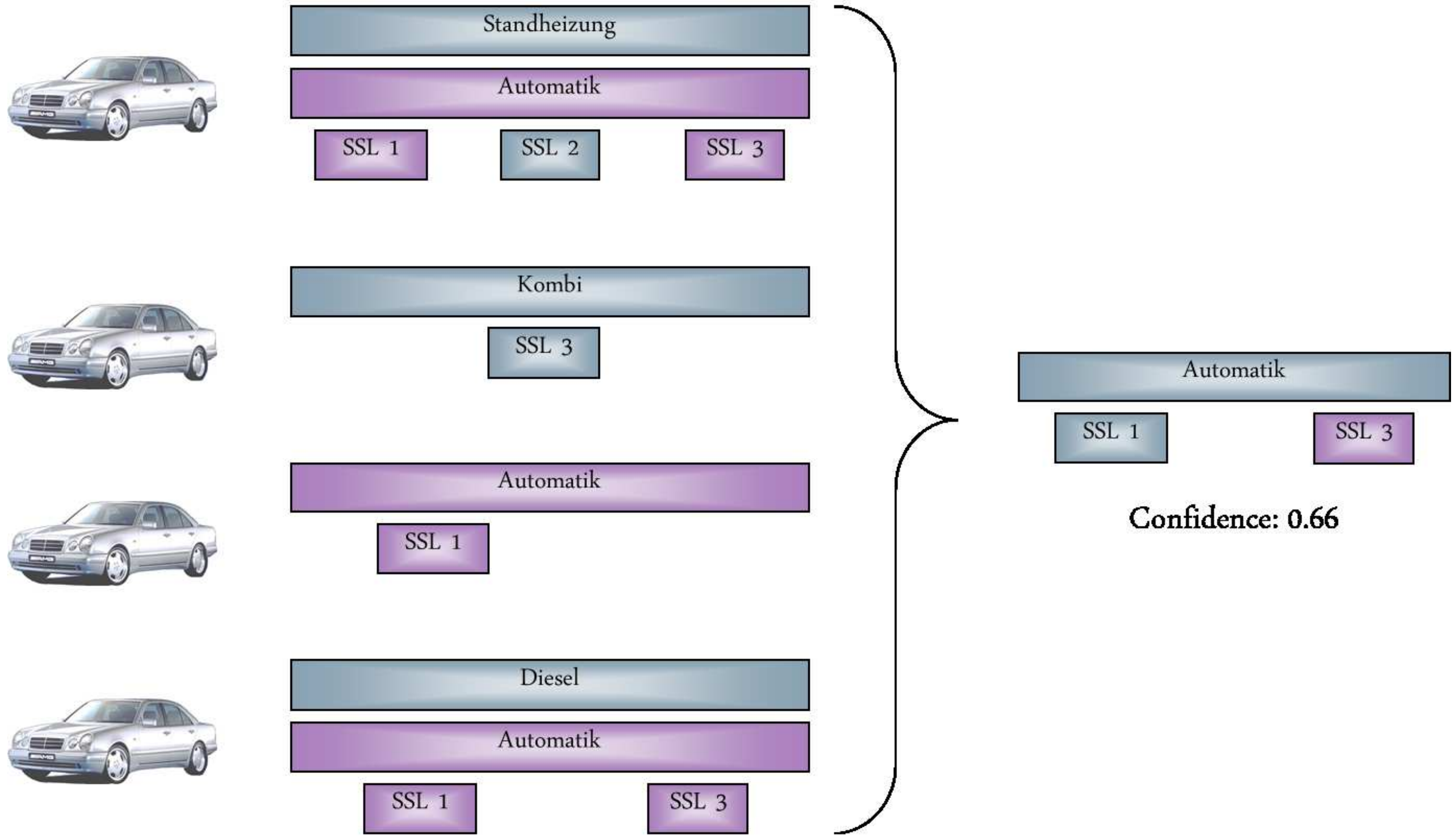
Qualitätsüberwachung von Fahrzeugen


Diesel, Automatik, AMG, ...
SSL 1 am 1.4.2006, SSL 2 am 11.11. 2006, ...
Q-DB
Standheizung
Automatik
SSL 1
SSL 2
SSL 3

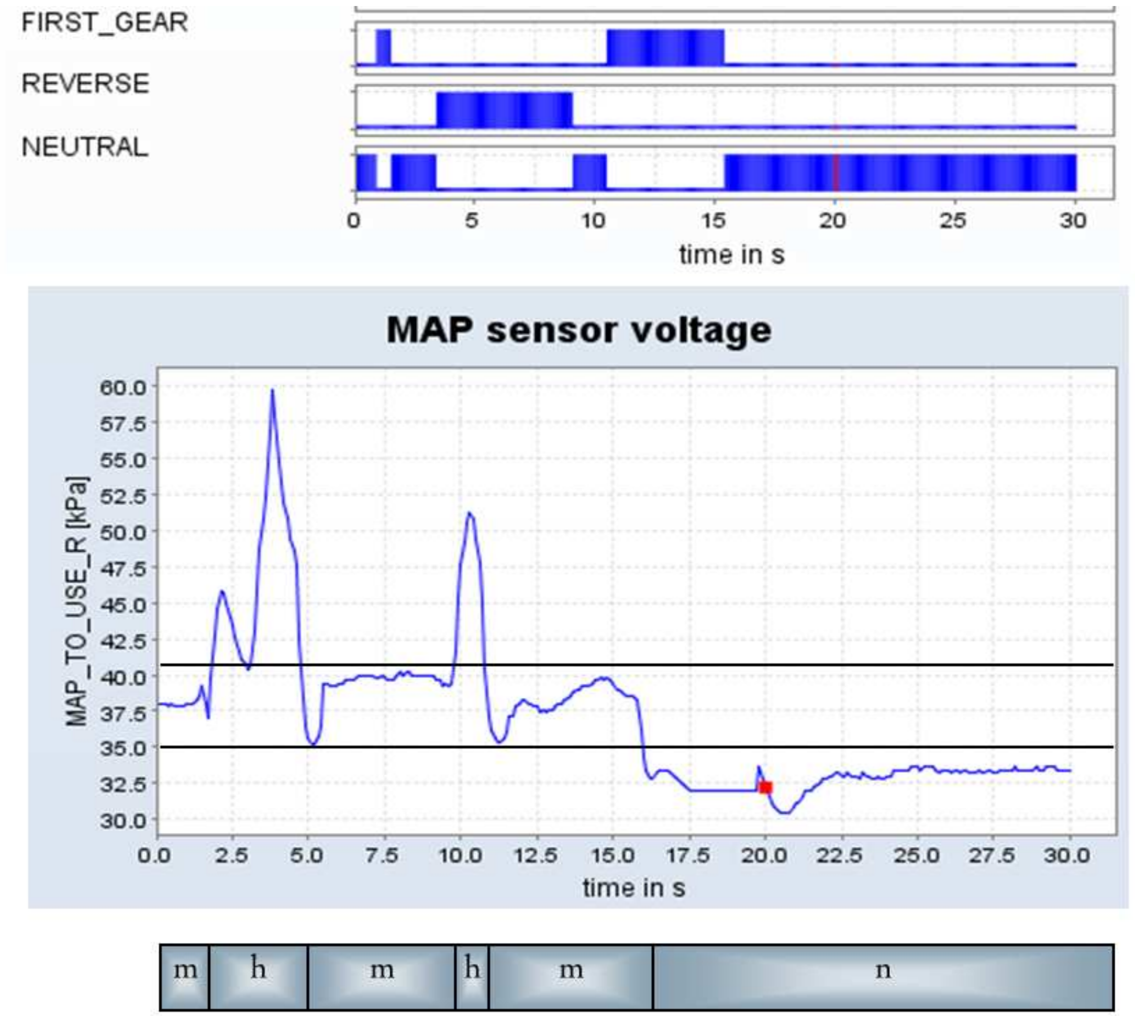
Qualitätsüberwachung von Fahrzeugen

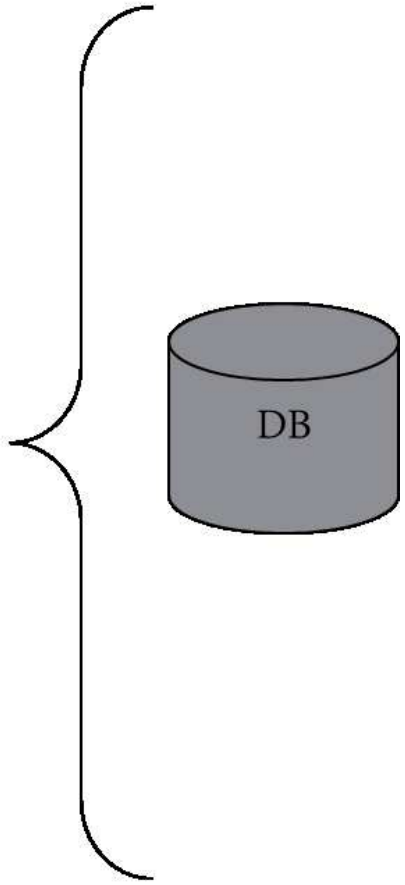


Qualitätsüberwachung von Fahrzeugen

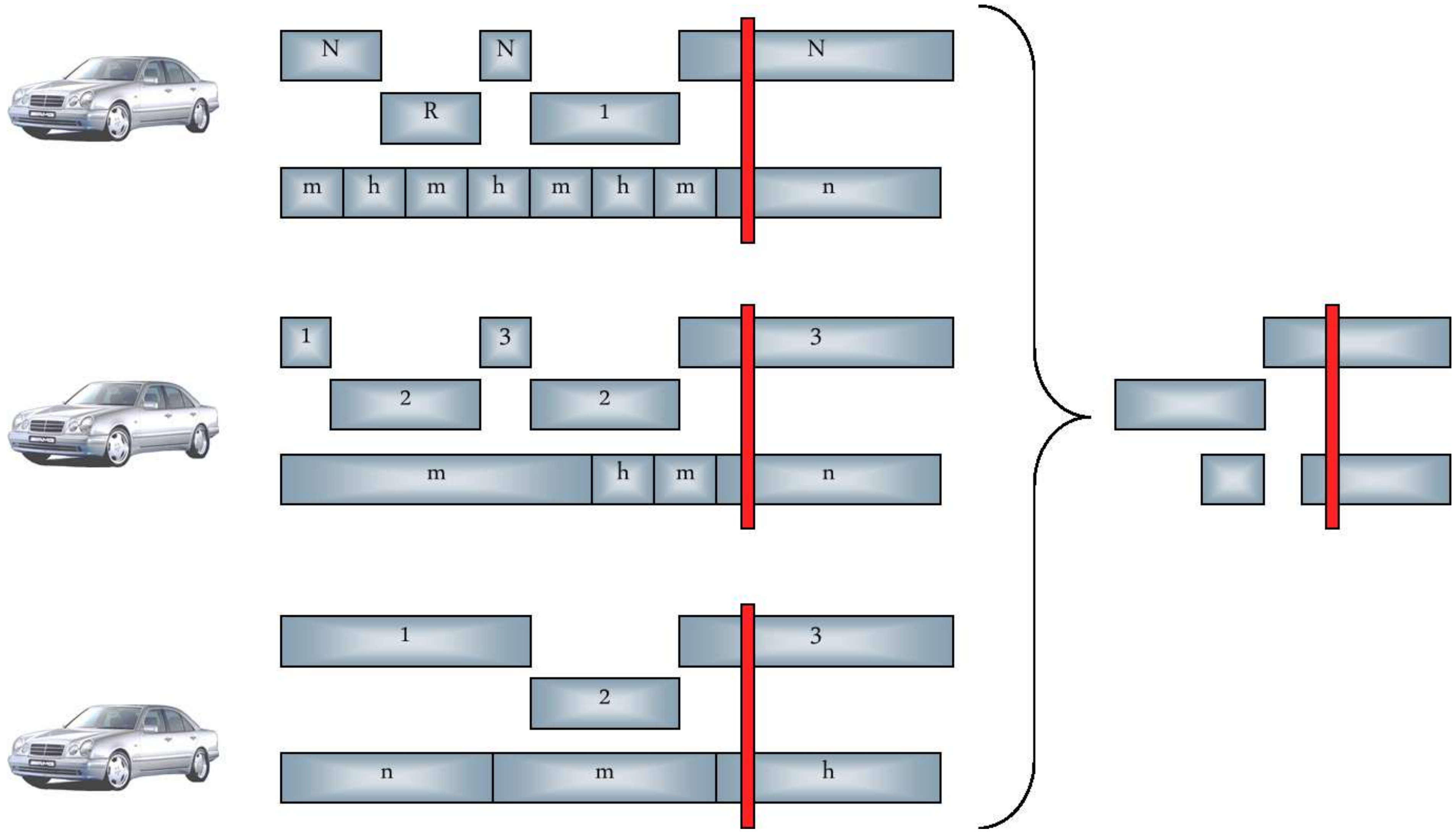


Vorserienfahrzeuge

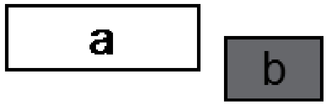
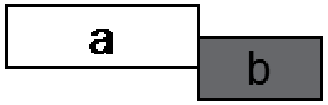
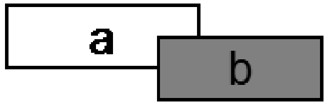
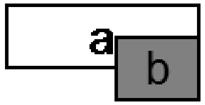
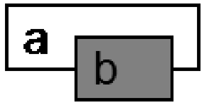
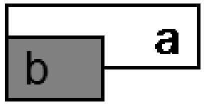
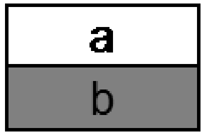




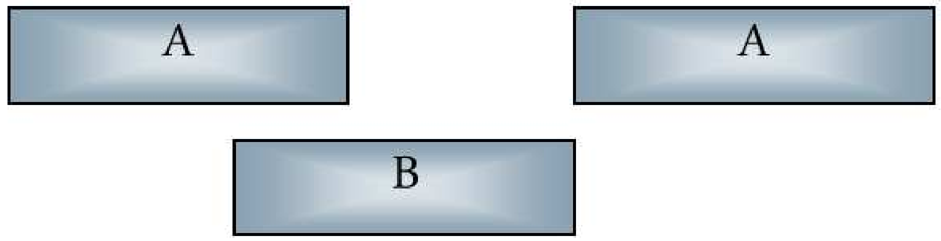
Vorserienfahrzeuge



Was ist ein temporales Muster?

Relation a to b		**Inverse Relation b to a**
a *before* b		b *after* a
a *meets* b		b *is-met-by* a
a *overlaps* b		b *is-overlaped-by* a
a *is-finished-by* b		b *finishes* a
a *contains* b		b *during* a
a *is-started-by* b		b *starts* a
a *equals* b		b *equals* a

Was ist ein temporales Muster?

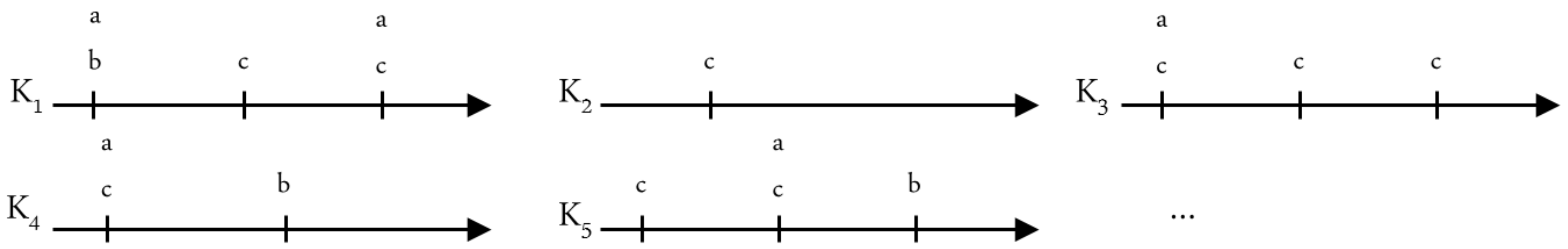


	A	B	A
A			
B			
A			

Inhalt

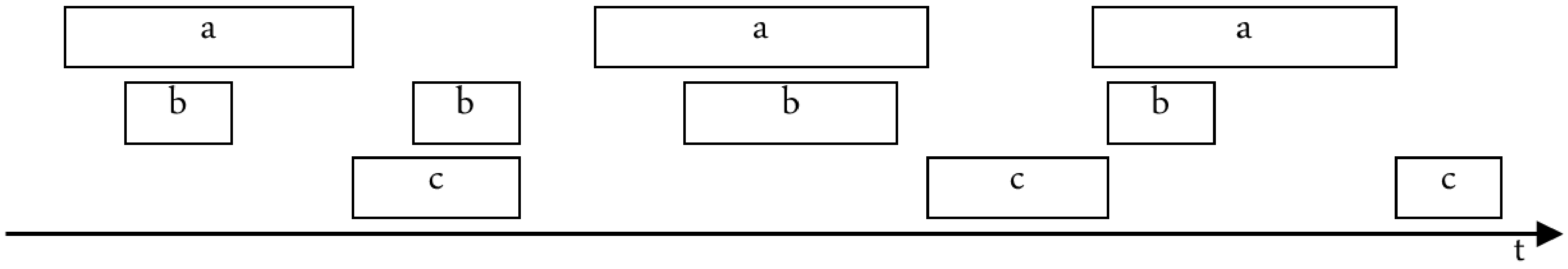
- Häufige Muster in zeitbezogenen Daten
  - Motivation / Problemstellung
  - **Abgrenzung zu bekannten Verfahren**
  - Algorithmen / Beispiel

Agrawal 1995



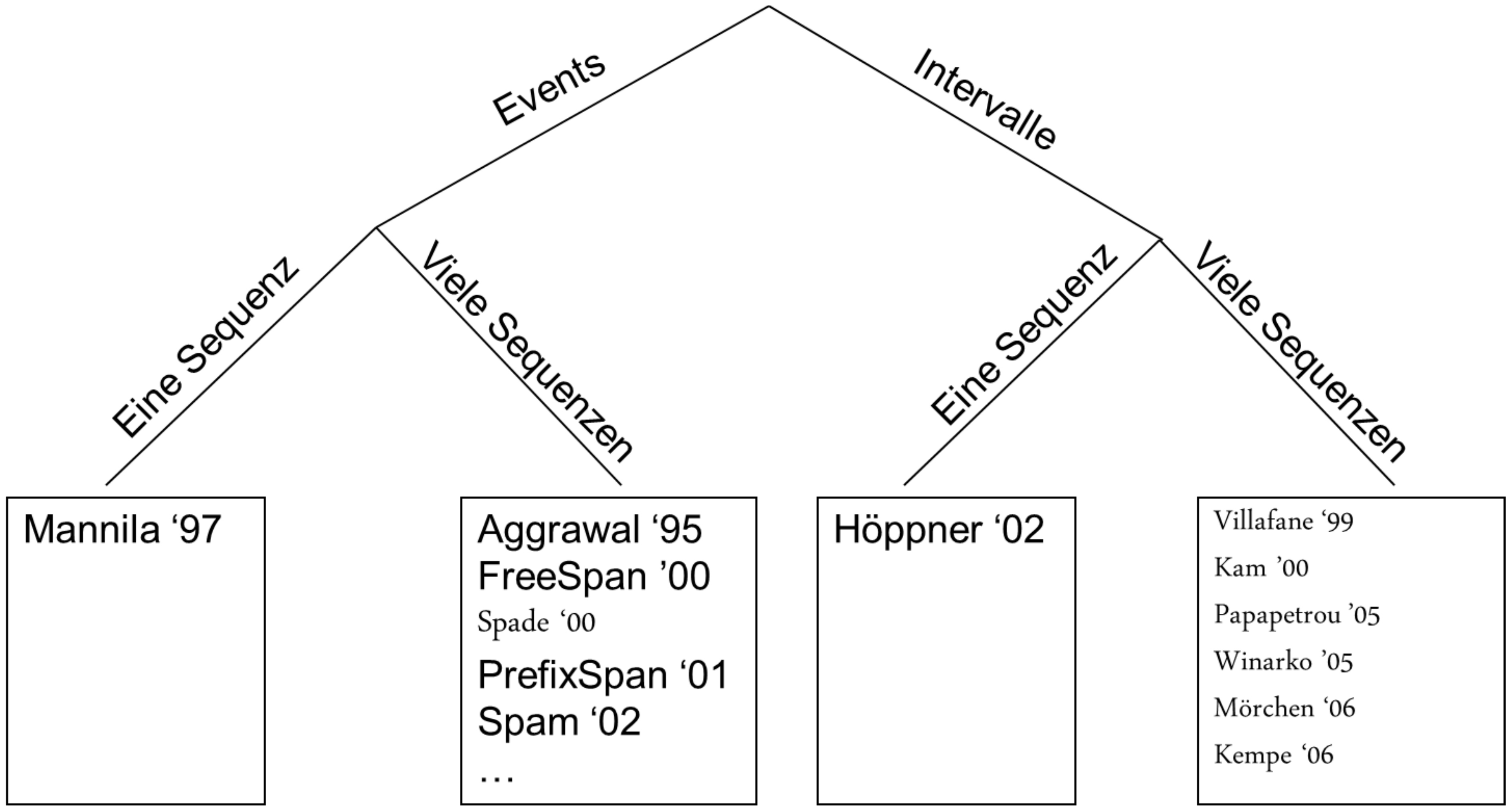
- Viele Kunden mit einer Historie von Itemsets
- Sucht Sequenzen
  - z.B. $\{a, b\} \to \{c\} \to \{b, c\}$
- Keine Regeln
- Kein Zeitfenster
- Support: Zählen
  - Supportzähler einer Sequenz wird inkrementiert, wenn sie bei einem Kunden vorkommt

Höppner 2002



- Ein einziger Zeitstrahl, Ereignisse haben zeitliche Ausdehnung
- Sucht Muster
  - a enthält b und trifft auf c
- Aus Mustern werden Regeln abgeleitet
- Benutzt ein Zeitfenster in dem Muster vorkommen muss
- Support: *Temporal Support*

Verwandte Verfahren



Inhalt

- Häufige Muster in zeitbezogenen Daten
  - Motivation / Problemstellung
  - Abgrenzung zu bekannten Verfahren
  - **Algorithmen / Beispiel**

Partielles Apriori-Kriterium

	A	B	A
A	e	o	b
B	io	e	m
A	a	im	e

- Alle Suffixe eines häufigen Musters sind häufig.

Kandidaten Generation

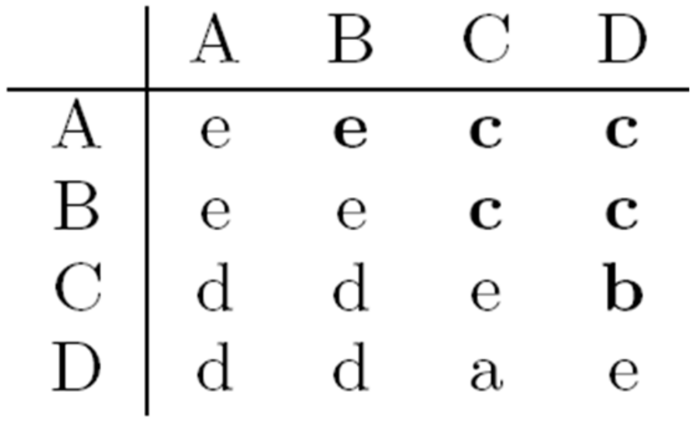
Pattern P	l_p	$l_1 \ldots l_{k-1}$
l_p	e	C
l_1		
$\vdots$	B	A
l_{k-1}		

Pattern Q	l_q	$l_1 \ldots l_{k-1}$
l_q	e	E
l_1		
$\vdots$	D	A
l_{k-1}		

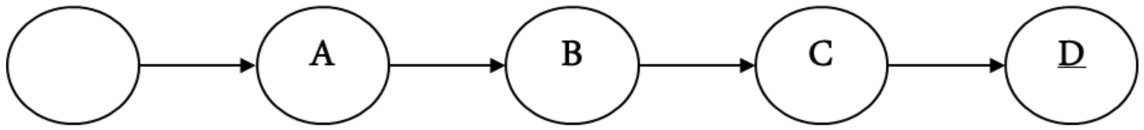
Pattern R	l_p	l_q	$l_1 \ldots l_{k-1}$
l_p	e	?	C
l_q	?	e	E
l_1			
$\vdots$	B	D	A
l_{k-1}			

Allens Intervall Relationen

Support Evaluation

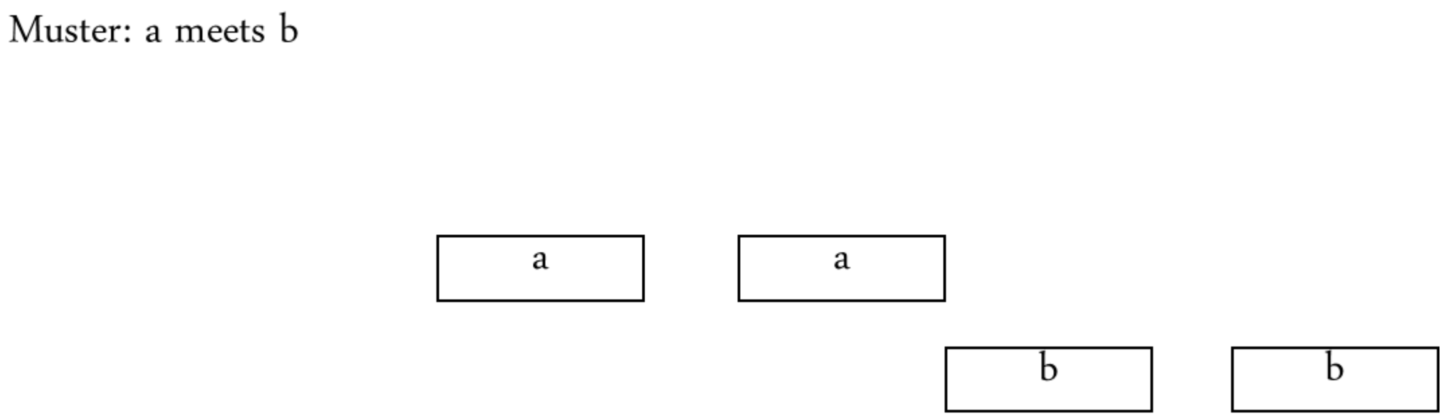


	A	B	C	D
A	e	**e**	**c**	**c**
B	e	e	**c**	**c**
C	d	d	e	**b**
D	d	d	a	e



- Ausnutzen der Eigenschaften von normalisierten Mustern, durch endliche Automaten

Endliche Automaten bleiben stecken

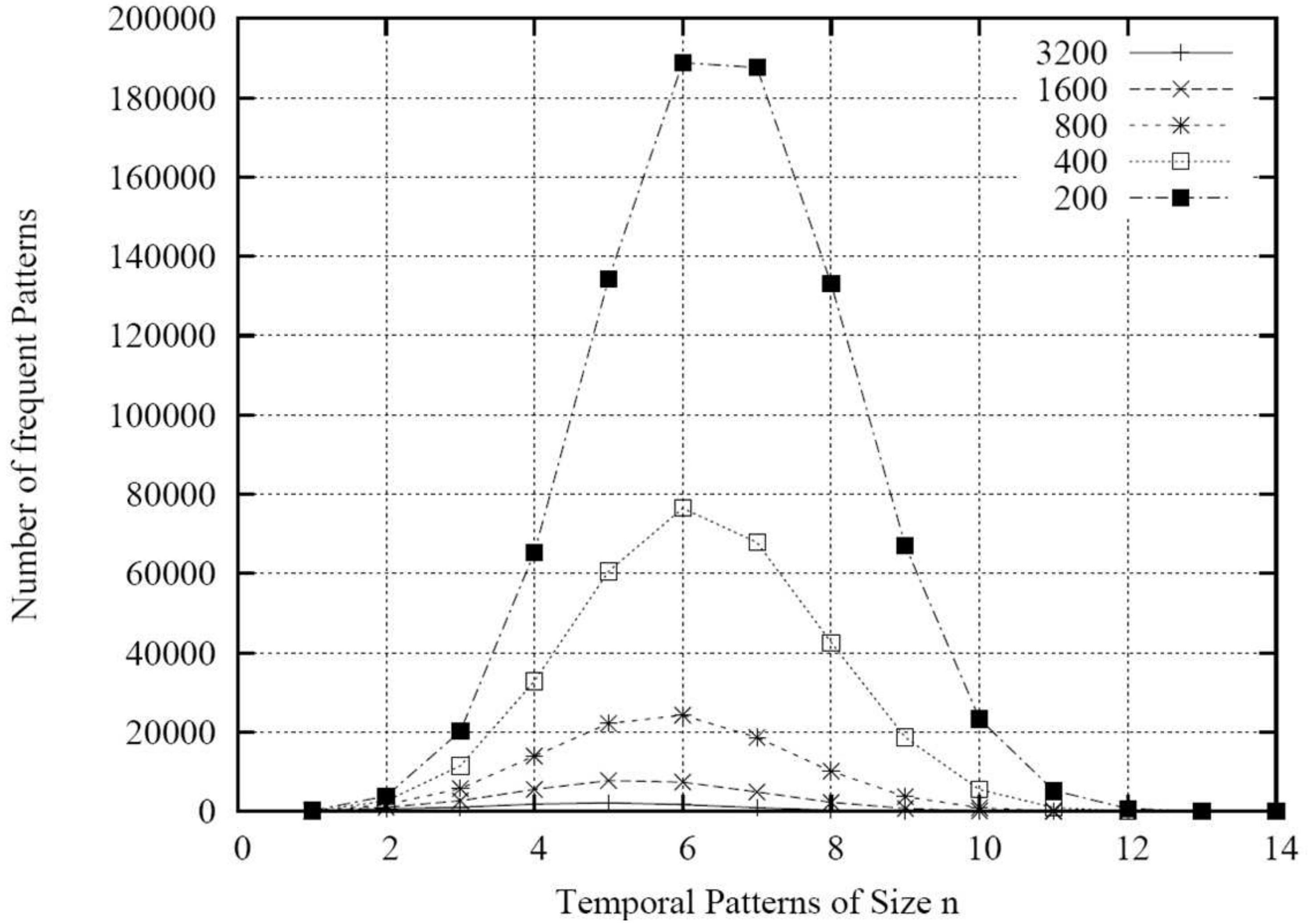


- Ein endlicher Automat bleibt nach akzeptieren des 1. a’s stecken.
- Lösung des Problems durch Anlegen von Kopien und Filtern der gefundenen Vorkommen

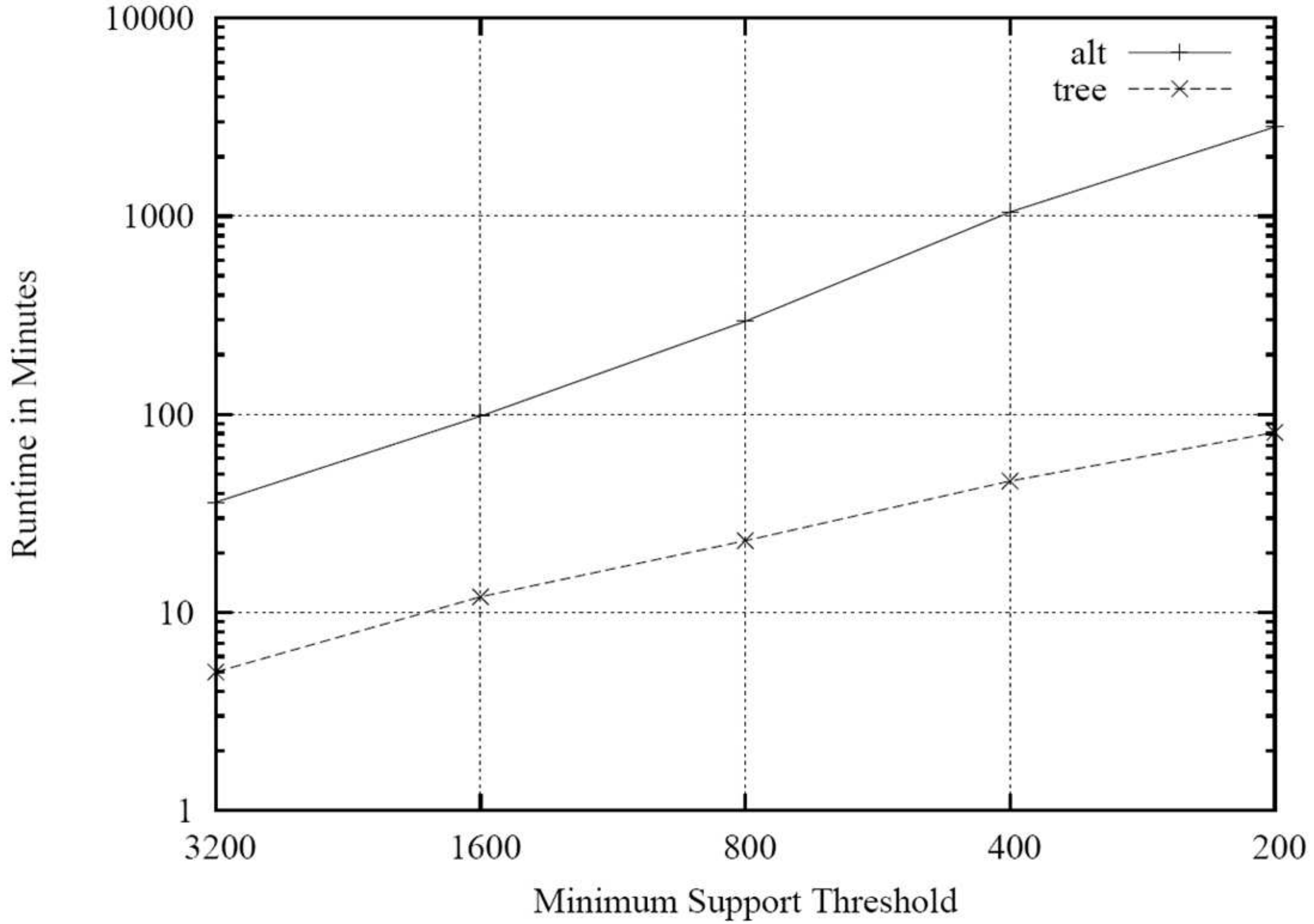
Beispiel: Qualitätsüberwachung von Fahrzeugen

- 101 250 Fahrzeuge
    - Werkstattaufenthalte
    - Fahrzeugkonfigurationsmerkmale
    - 1,4 Mio. temporale Intervalle

Anzahl häufiger Muster



Laufzeit



alt
tree
Runtime in Minutes
10000
1000
100
10
1
3200
1600
800
400
200
Minimum Support Threshold

Motive in Zeitreihen effizient finden

Inhalt

- Motive in Zeitreihen effizient finden
  - Data Mining in Zeitreihen
  - Speicherplatzeffiziente Repräsentationen
  - Symbolische Aggregat-Approximation (SAX)
  - Finden von Motiven in Zeitreihen mit SAX
  - Anwendungsbeispiel

Data Mining in Zeitreihen

- große Herausforderung: Finden von nützlichen Informationen aus Zeitreihen
- typische Problemen: Clustering, Klassifizierung, Entdeckung von häufigen Mustern und Regeln, Visualisierung, Erkennung von Anomalien
- Probleme reduzieren sich wegen großer Datenfülle häufig auf das Finden von wiederkehrenden, sich (einander) ähnelnden Teilsequenzen
- benötigt: Ähnlichkeitsmaß um Teilsequenzen miteinander zu vergleichen
- z.B. Euklidischer Abstand

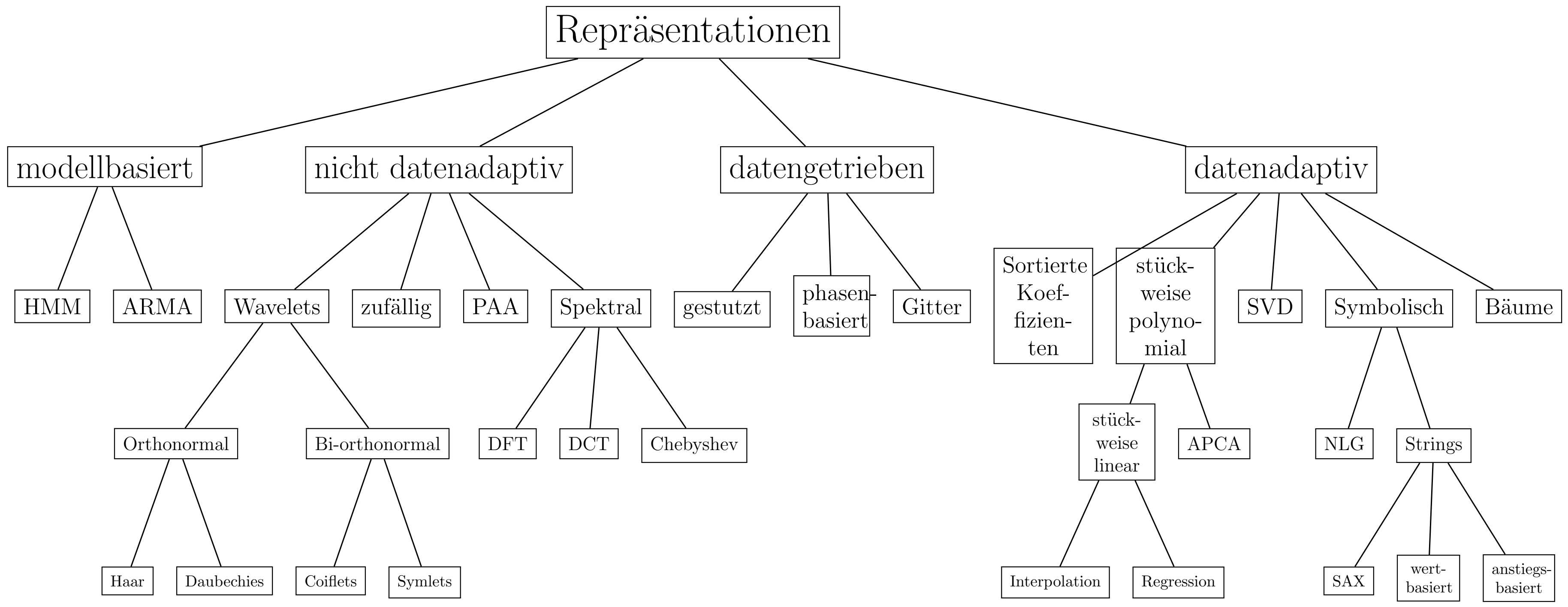
  $$d(Q, C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$

  zwischen 2 standardnormalverteilten Teilsequenzen $Q = (q_1, \ldots, q_n)^T$ und $C = (c_1, \ldots, c_n)^T$
- Problem: sehr viele Vergleiche und Kapazität des schnellen Arbeitsspeichers meist zu klein um alle Daten zu laden

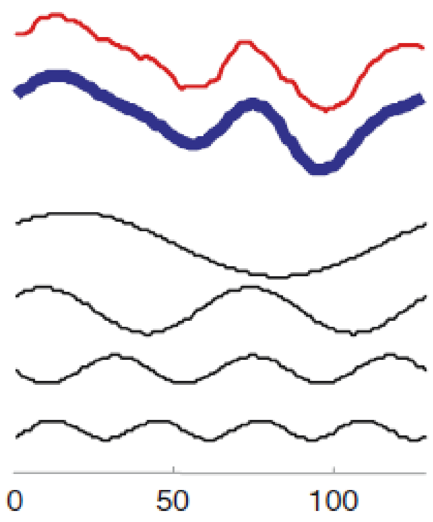
Speicherplatzeffiziente Repräsentationen

- Problem: viele, langsame Zugriffe auf Originaldaten
- Lösung: Approximation der Zeitreihen, die in Arbeitsspeicher passt und interessante Eigenschaften erhält
- z.B. diskrete Fourier-Transformation (DFT), diskrete Wavelet-Transformation (DWT), stückweise lineare (PLA) und adaptiv stückweise konstante Modelle (APCA), Singulärwertzerlegung (SVD)
- hier: symbolische Repräsentationen
- Vorteile: Algorithmen der Textverarbeitung und Bioinformatik anwendbar, z.B. Hashing, Markov-Modelle, Suffixbäume, usw.

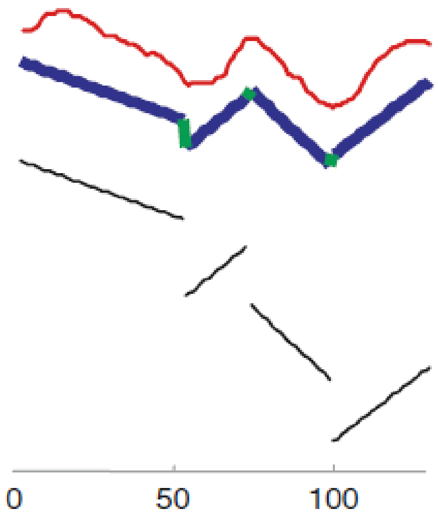
Zeitreihen-Repräsentationen



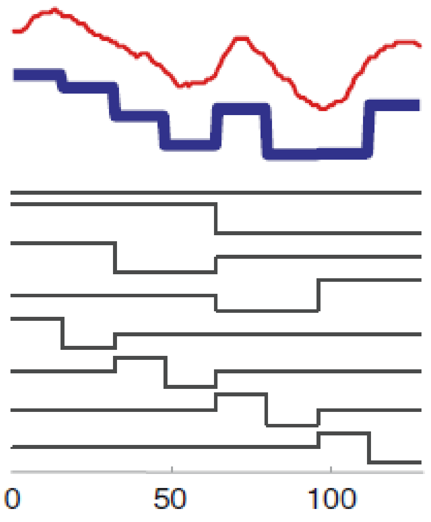
Die meist genutzten Repräsentationen



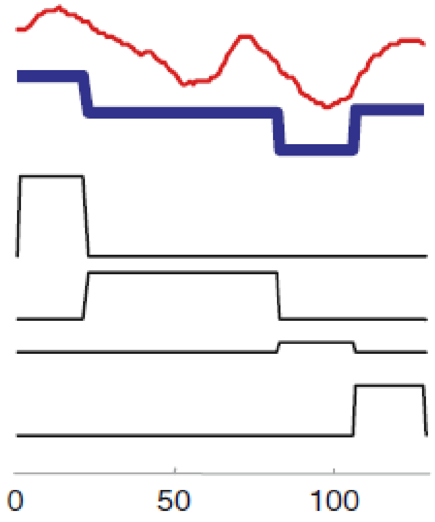
DFT



PLA

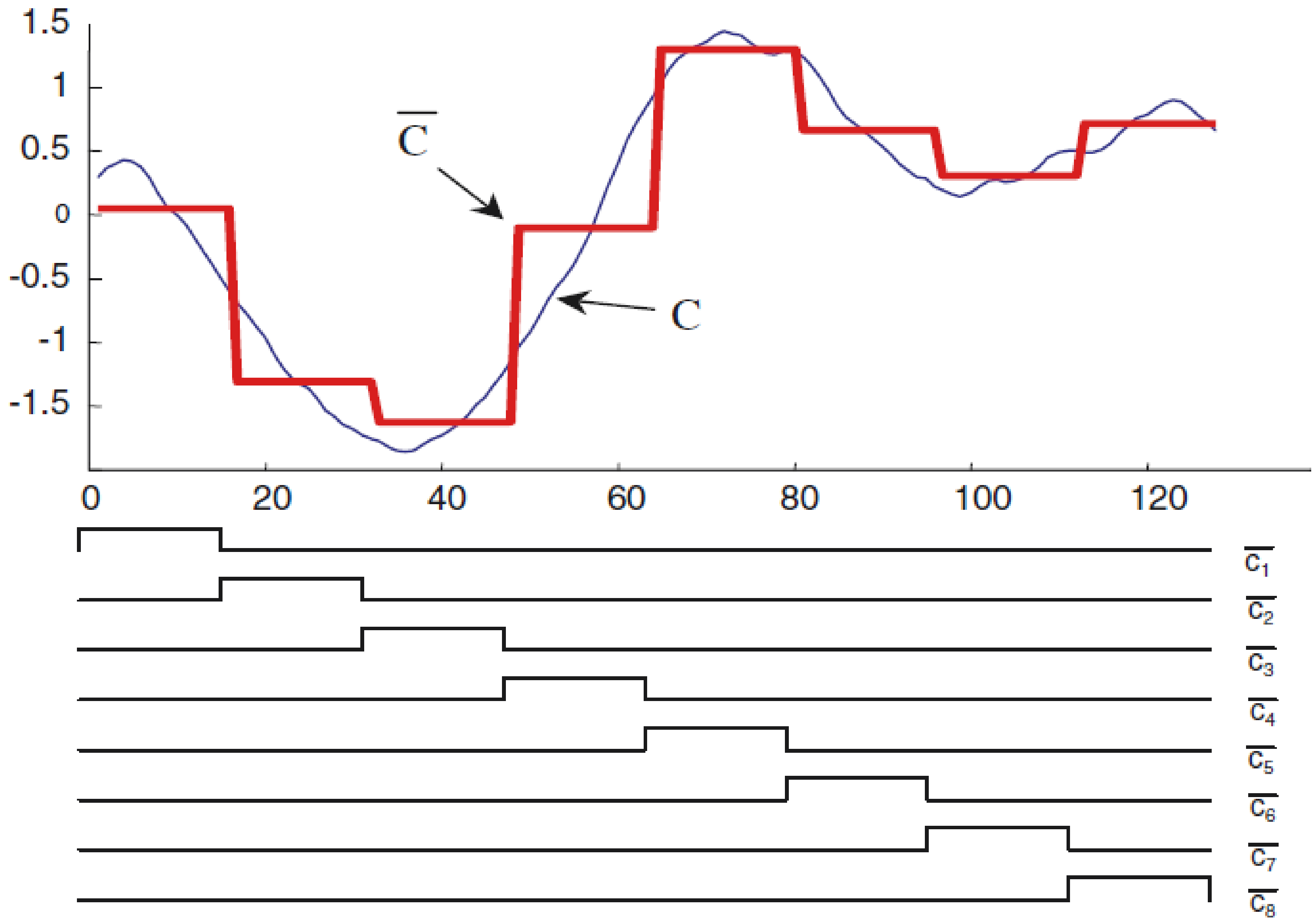


Haar wavelet



APCA

Stückweise aggregierte Approximation (PAA)



Reduktion von 128 auf 8 Datenpunkte

Symbolische Aggregat-Approximation (SAX)

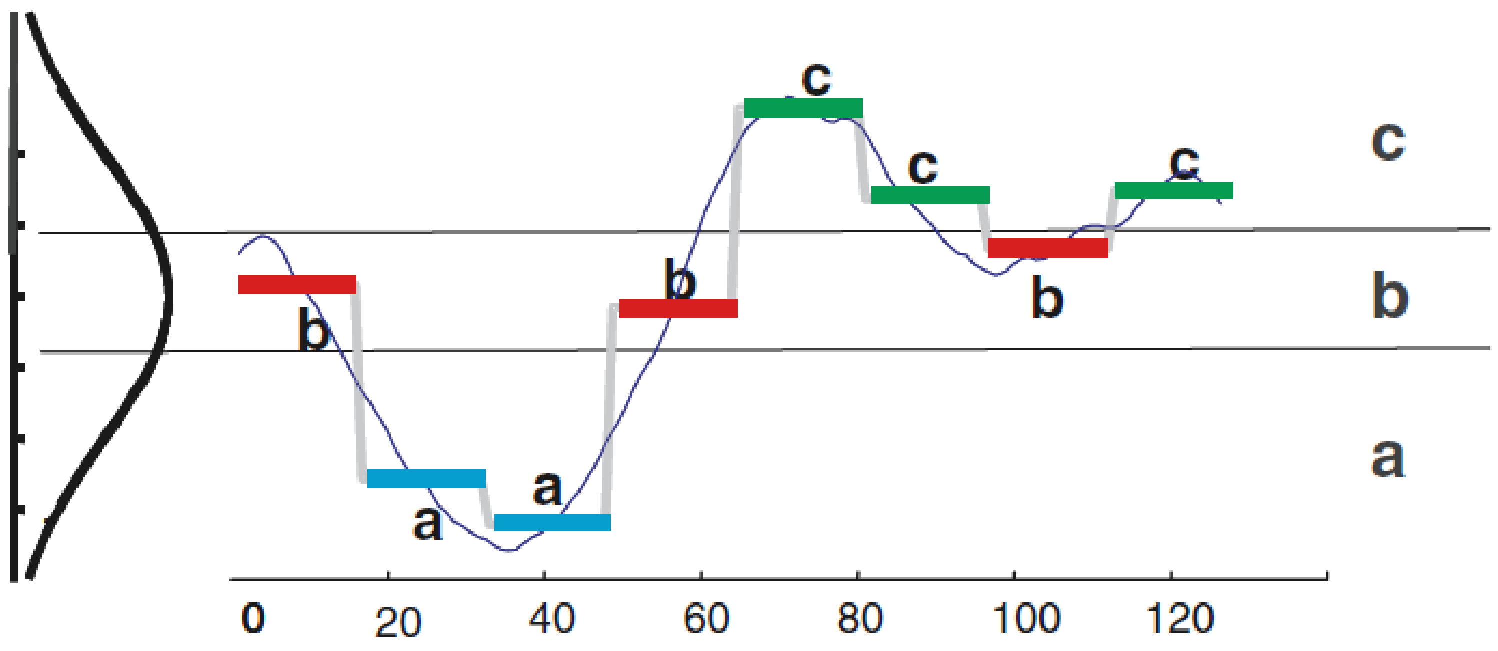
- jede Sequenz der Länge n wird ein Wort definierter Länge w über ein gewähltes Alphabet $A = \{\alpha_1, \dots, \alpha_a\}$ mit $|A| = a$
- simpler Algorithmus:
  1. Teile Teilsequenz in w gleichgroße Intervalle
  2. PAA: Bilde von jedem Intervall dessen Mittelwert (Repräsentant) $C = (c_1, \dots, c_n)^T$ wird abgebildet auf $\bar{C} = (\bar{c}_1, \dots, \bar{c}_w)$ durch
  $$\bar{c}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j$$
  3. Bilde jeden Mittelwert $\bar{c}_i$ von $\bar{C}$ auf einen der a Buchstaben durch
  $$\hat{a}_i = \alpha_j \Leftrightarrow \beta_{j-1} \leq \bar{c}_i \leq \beta_j$$
- Annahmen: Wertebereich der PAA-Sequenz sei normalverteilt und Auftreten jedes Buchstaben ist gleich wahrscheinlich
- Abbildung $\bar{c}_i \mapsto b \in A$ durch “Bruchstellen” $\beta_1, \dots, \beta_{a-1}$

“Bruchstellen” der Normalverteilung

$\lvert A\rvert$	3	4	5	6	7	8	9	10
β_1	$-.43$	$-.67$	$-.84$	$-.97$	-1.07	-1.15	-1.22	-1.28
β_2	0.43	0	0.25	0.43	0.57	0.67	0.76	0.84
β_3		0.67	0.25	0	$-.18$	$-.32$	$-.43$	$-.52$
β_4			0.84	0.43	0.18	0	$-.14$	$-.25$
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

- Bruchstellen teilen Normalverteilung in gleichwahrscheinliche Regionen

Beispiel: SAX



- hier: $n = 128, w = 8, a = 3$
- Ergebnis: **baabccbc**

Abstandsmaß für SAX

- PAA: untere Schranke zum Euklidischen Abstand durch

$$d_r(\bar{Q}, \bar{C}) = \sqrt{\frac{n}{w}}\sqrt{\sum_{i=1}^{w}(\bar{q}_i - \bar{c}_i)^2}$$

- SAX:

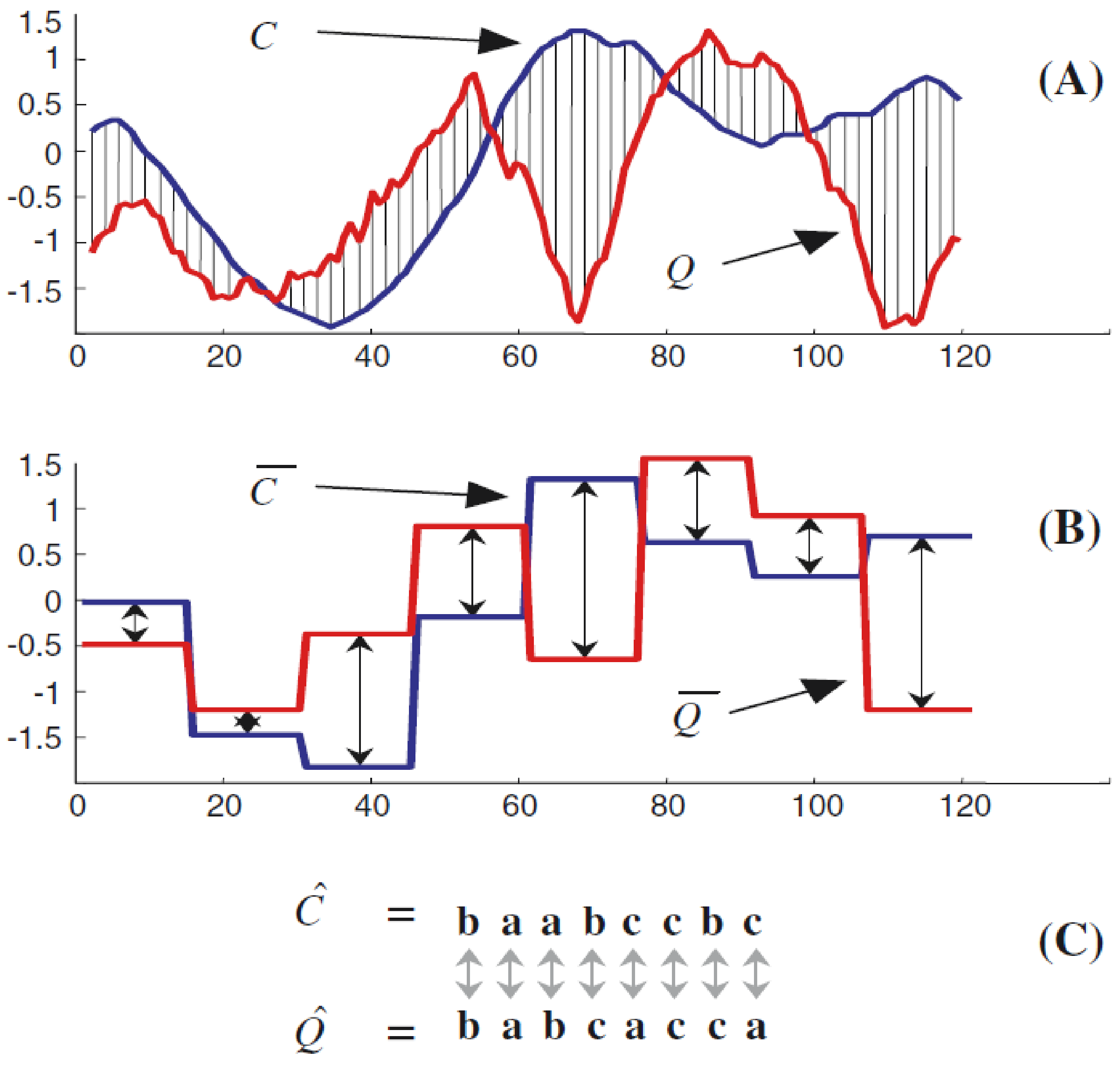
$$d^*(\hat{Q}, \hat{C}) = \sqrt{\frac{n}{w}}\sqrt{\sum_{i=1}^{w} d_a^*(\hat{q}_i, \hat{c}_i)^2}$$

- Abstandsmaß d_a^* sollte über Lookup-Tabelle definiert werden, z.B. für $a = 4$

	a	**b**	**c**	**d**
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.340	0.67	0	0

$$d_a^*(r, c) = \begin{cases} 0 & \text{falls } |r - c| \leq 1, \\ \beta_{\max(r,c)-1} - \beta_{\min(r,c)} & \text{sonst} \end{cases}$$

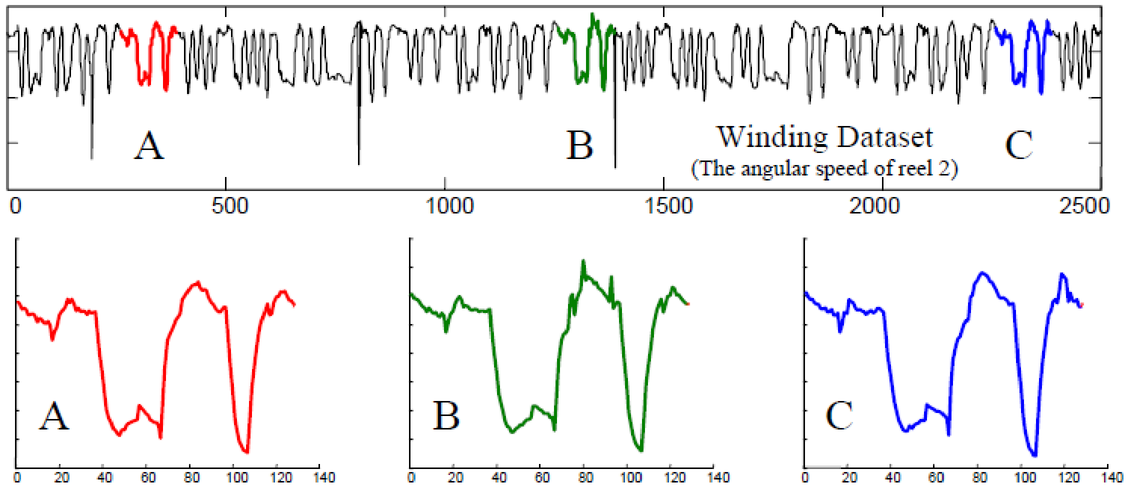
Vergleich der Abstandsmaße



SAX-Vorteil: untere Schranke

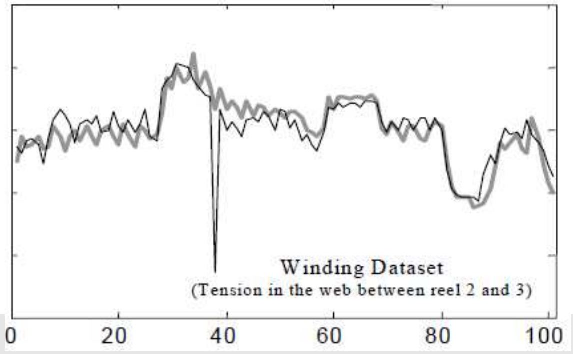
- $d^*(\hat{Q}, \hat{C})$ ist *untere Schranke* des Euklidischen Abstandes $d(Q, C)$ der Originalsequenzen Q und C
$$d^*(\hat{Q}, \hat{C}) \leq d(Q, C)$$
- falls $\hat{Q}$ und $\hat{C}$ zueinander unähnlich sind, so sind es auch Q und C
- SAX-basierte Algorithmen produzieren identische Resultate verglichen mit Algorithmen, die auf Originaldaten arbeiten
- "nur" ähnliche SAX-Worte sollten im Originalraum verglichen werden
- meist jedoch nur wenige Zugriffe auf Originaldaten

Finden von Motiven in Zeitreihen

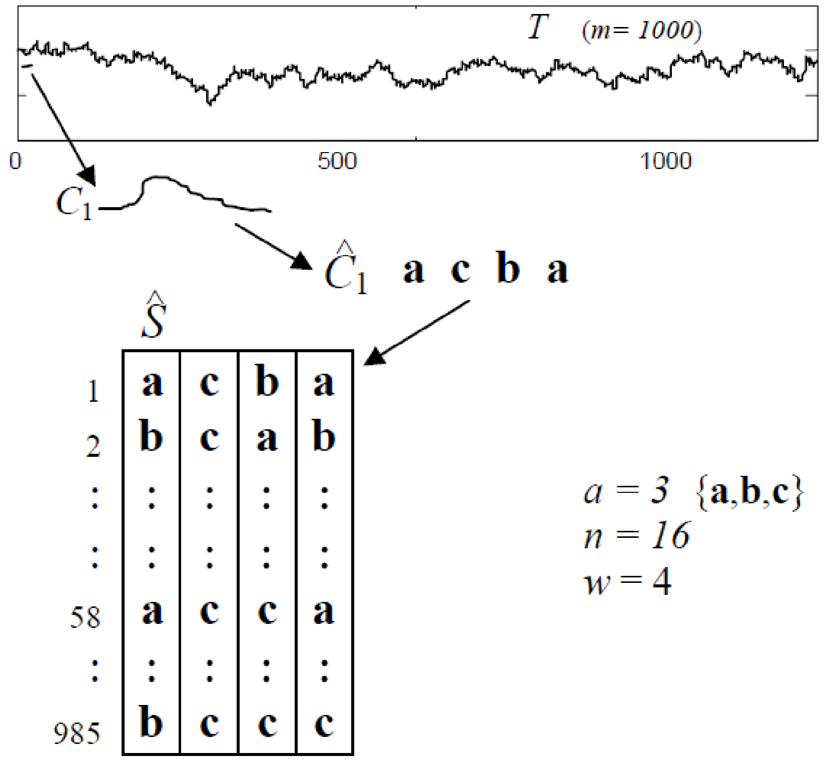


- Motive: Primitive, häufige (ähnliche) Muster, Prototypen

- Herausforderungen:
  - Motive sind vorher unbekannt
  - vollständige Suche ist kostspielig mit $O(n^2)$
  - Ausreißer beeinflüssen euklidischen Abstand



Erstellen der SAX-Matrix



- Finden aller Motive einer Zeitreihe der Länge m durch "Sliding Window"
- Fensterbreite n führt zu $(m - n + 1)$ Teilsequenzen
- Transformation jeder Sequenz in ein SAX-Wort der Länge w
- Speicherung in Zeilenmatrix (sog. SAX-Matrix)
- hat w Spalten und $(m-n+1)$ Zeilen

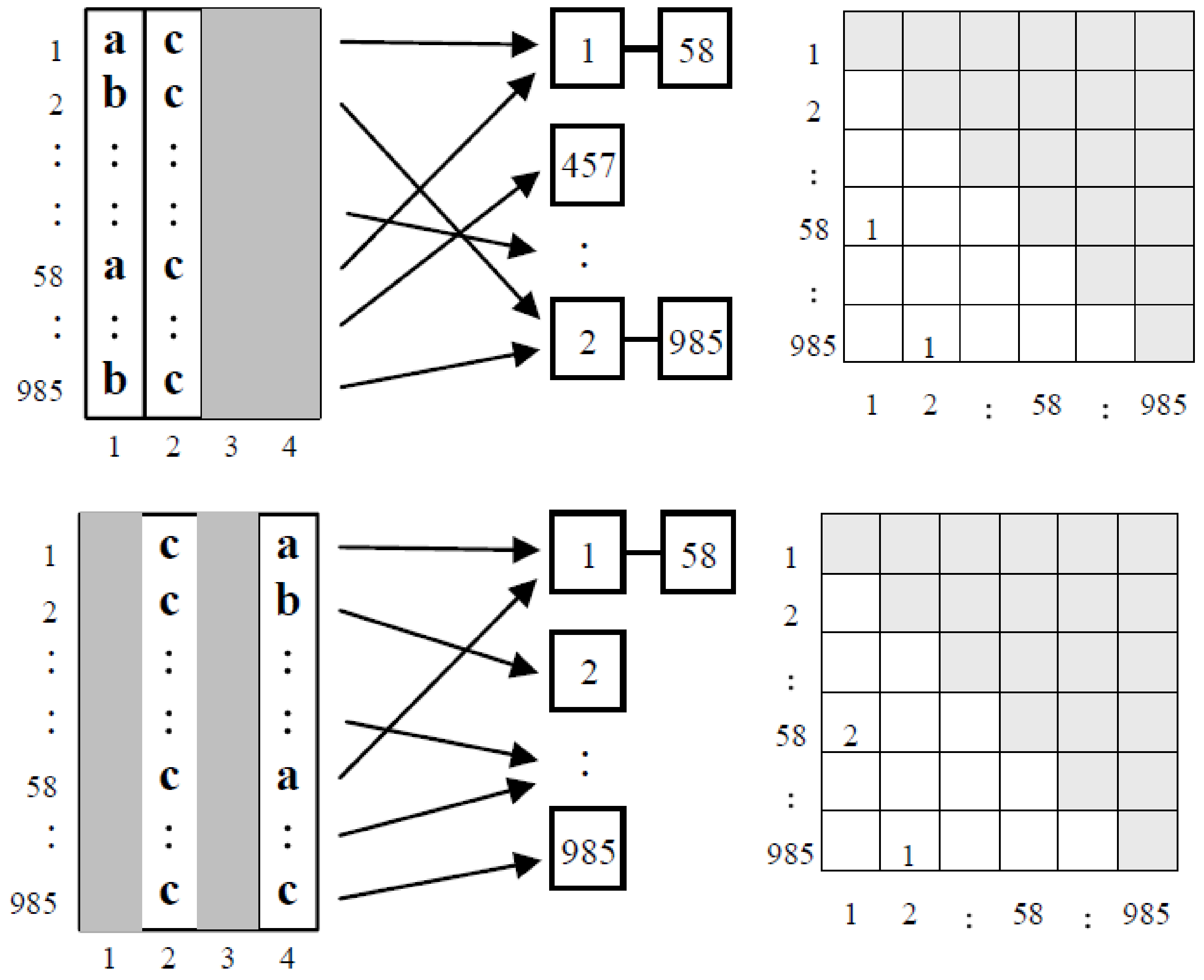
Zufallsprojektion

- Erraten der Motiv-Positionen durch sog. Zufallsprojektion
- paarweise Vergleiche der SAX-Worte
- Kollisionsmatrix M mit $(m - n + 1)^2$ Feldern für jeden der Vergleiche
- M durch Hash-Tabelle effizient implementieren!
- anfangs $M(i, j) = 0$ für $1 \leq i, j \leq m - n + 1$
- Idee: Zeichen für Zeichen zweier Worte in SAX-Matrix miteinander vergleichen
- bessere Annahme: “don’t care symbols” in Sequenzen mit unbekanntem Ort
- z.B. verrauschtes Motiv oder Streckung bzw. Stauchung einer Sequenz

Zufallsprojektion

- folglich: SAX-Matrix wird auf $1 \leq k < w$ zufällig ausgewählte Spalten projiziert
- Vergleich aller Zeilen der projizierten Matrix
- falls 2 projizierte SAX-Worte in Zeilen i und j identisch, so wird $M(i, j)$ inkrementiert
- Projizieren wird t mal wiederholt, weil einige Motive sich nach t Iterationen wahrscheinlich zusammen einen Eintrag in M teilen werden
- es ist auch unwahrscheinlich, dass viele zufällige Sequenzen zusammen mit einem bereits gefundenen Motiv kollidieren werden
- nutzerdefinierter Schwellenwert s mit $1 \leq s \leq k$ für Kollisionseinträge in M
- alle $M(i, j) \geq s$ wären demnach Kandidaten für Motive
- aber: in unmittelbarer Nachbarschaft einer Sequenz i kommen ähnliche Sequenzen vor (sog. triviale Übereinstimmungen)
- diese werden abschließend gestrichen

Zufallsprojektion



- ersten zwei Iterationen der Zufallsprojektion

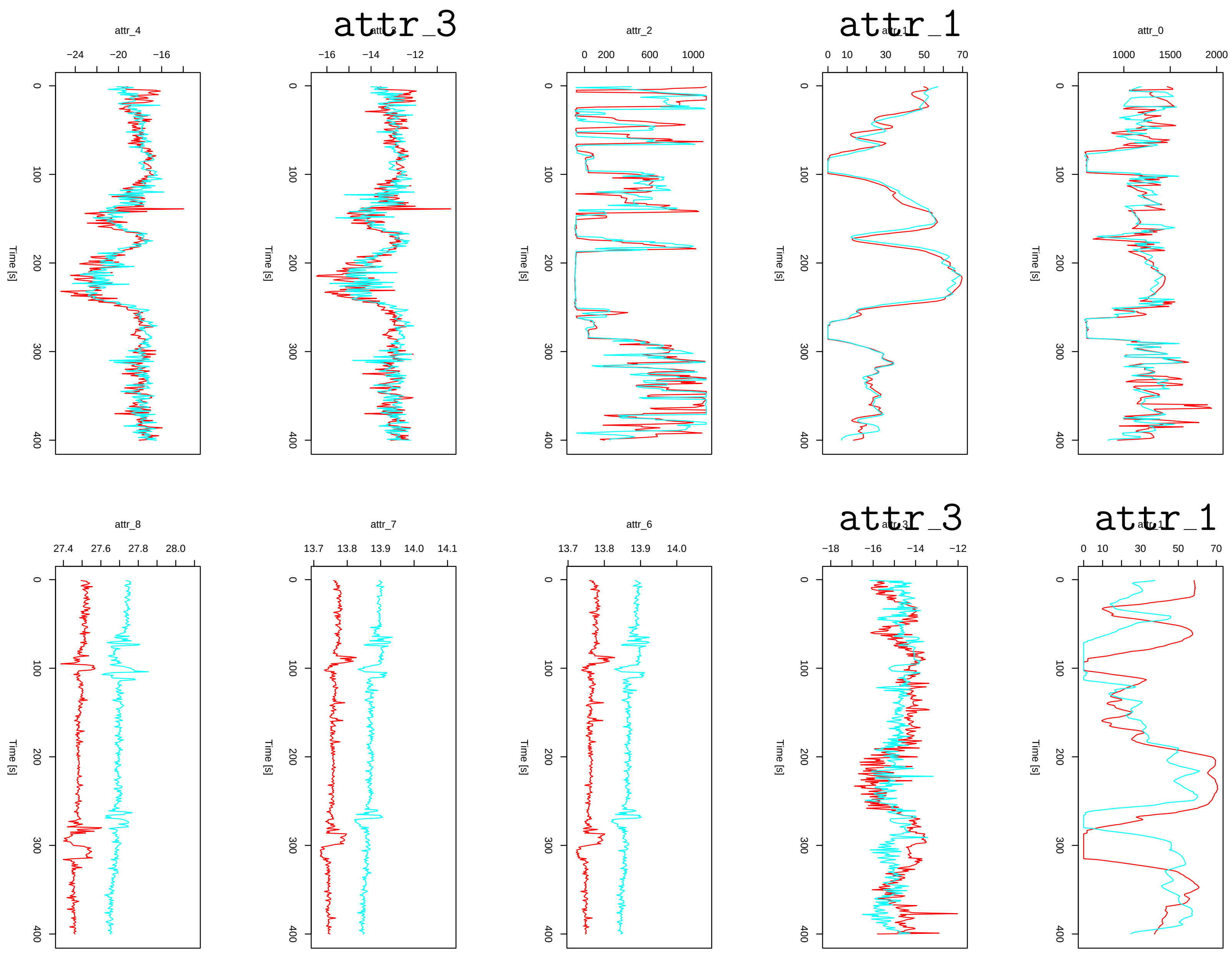
Subdimensionale Motive

- Zufallsprojektion für Motive in univariaten SAX-Zeitreihen kann auch multivariat verwendet werden
- Idee: Inkrementiere *pro Attribut* $j \in \{1, \ldots, p\}$ die Kollisionsmatrix M für jedes projiziertes SAX-Wort
- Problem: relevanten Dimensionen der potentiellen subdimensionalen Motive sind unbekannt
- Lösung:
    - Schätzen eine Verteilung $P(d_j)$ über Abstände zwischen nicht-trivialen Übereinstimmungen durch Ziehen einer Stichprobe
    - Bestimmung der Abstände $d_1^*, \ldots, d_p^*$ für jeden Eintrag $M(i, j) \geq s$
    - falls $P(d_j \leq d_j^*) < r_j^{\text{rel}}$ (nutzerspezifische *Dimensionsrelevanz*), dann ist j-tes Attribut relevant

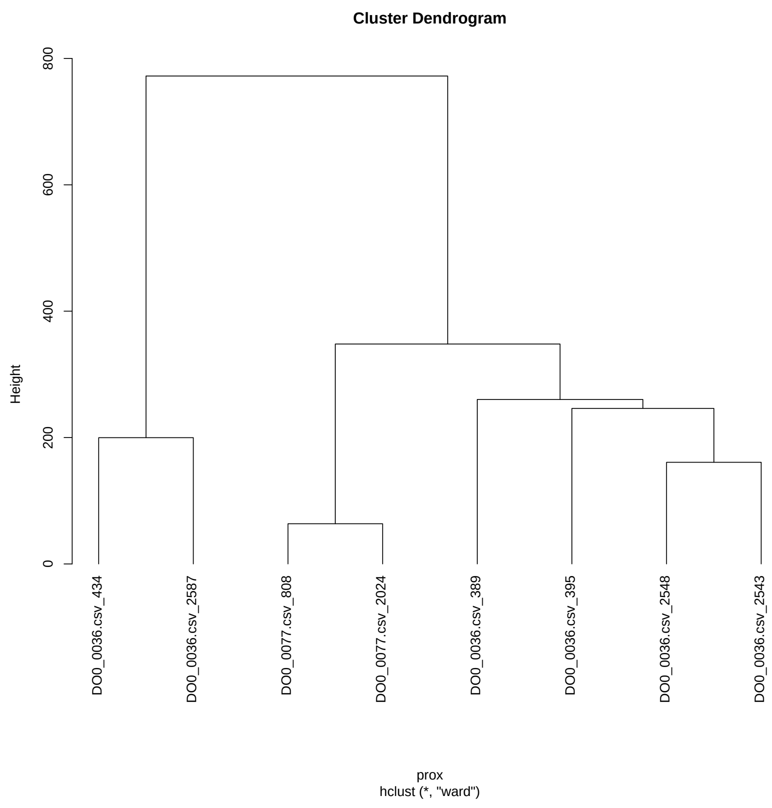
Anwendungsbeispiel

- Experte identifizierte $p = 9$ von insgesamt 130 Messkanälen als wichtig
- Motiv dauert mindestens $n = 400\,\mathrm{ms}$
- 10 Zeitreihen gegeben zum Suchen nach subdimensionalen Motiven

Subdimensionales Motiv in zwei Zeitreihen



Clustering der Motive



- Aufstellen einer Unähnlichkeitsmatrix durch paarweiser Vergleiche aller gefundenen Muster der 10 Zeitreihen basierend auf d^*
- Matrix ist symmetrisch, positiv und enthält auf Hauptdiagonale Nullelemente
- kann zum Gruppieren der Vorkommen genutzt werden, um Motive zu finden, die in mehreren Zeitreihen verteilt auftreten
- hier: hierarchisches Clustern aller Motive, die Attribute `attr_1` `attr_3` enthielten