Karlsruher Institut für Technologie

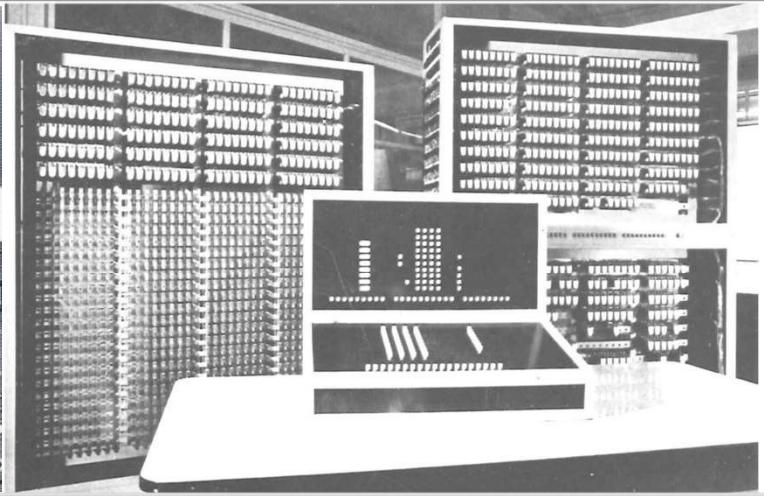
Institut für Technik der Informationsverarbeitung

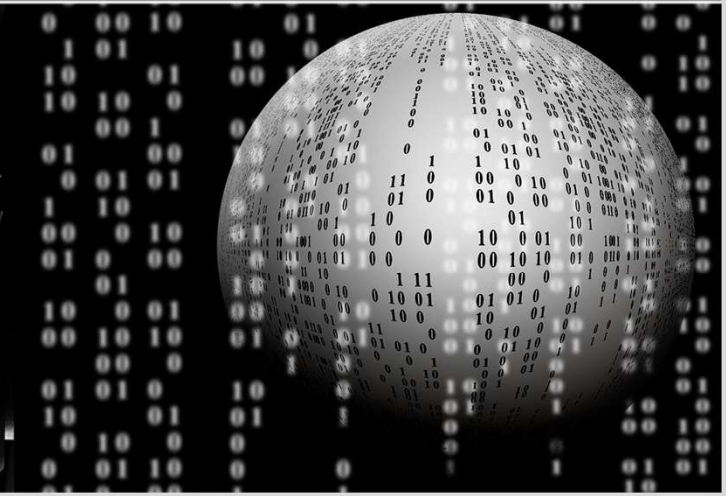
Institutsleitung
Prof. Dr.-Ing. J. Becker
Prof. Dr.-Ing. E. Sax
Prof. Dr. rer. nat. W. Stork

1. Einführung

Informationstechnik II

Prof. Dr.-Ing. Eric Sax



Who am I?

- Prof. Dr.-Ing. Eric Sax
- Dipl.-Ing. Etec.
  @Universität Karlsruhe
- 10 Jahre
  @Forschungszentrum Informatik
- 10 Jahre
  @MBtech Group
  - Projekte mit Mercedes, Getrag, BMW, Marquardt, Hella, Bosch, …
  - Leiter des Steuerkreises der EE-Entwicklung of für den SLS
- 5 Jahre
  @Daimler Buses
  - Weltweite EE Entwicklungsverantwortung



- @KIT
  - Leiter des Instituts für Technik der Informationsverarbeitung (ITIV)
  - Mehr als 70 Papers und 1 Buch zum Thema Hardware in the loop testing (HiL)
- @FZI
  - Direktor der Abteilung Elektronische Systeme und Mikrosysteme (ESM)

Ansprechpartner

- Vorlesung
  - Prof. Dr.-Ing. Eric Sax
    - Gebäude 30.10 - Raum 336
    - Telefon: 608-42500
    - eric.sax@kit.edu
    - http://www.itiv.kit.edu
    - Sprechstunden nach Vereinbarung
- Übungen und Tutorium
  - M.Sc. Nathalie Brenner
    - Wissenschaftliche Mitarbeiterin
    - Gebäude 30.10 - Raum: 126
    - Telefon 608-43093
    - nathalie brenner@kit edu





Meine persönlichen Meilensteine:

New Coaches and City Buses Euro6 (2009-2015)

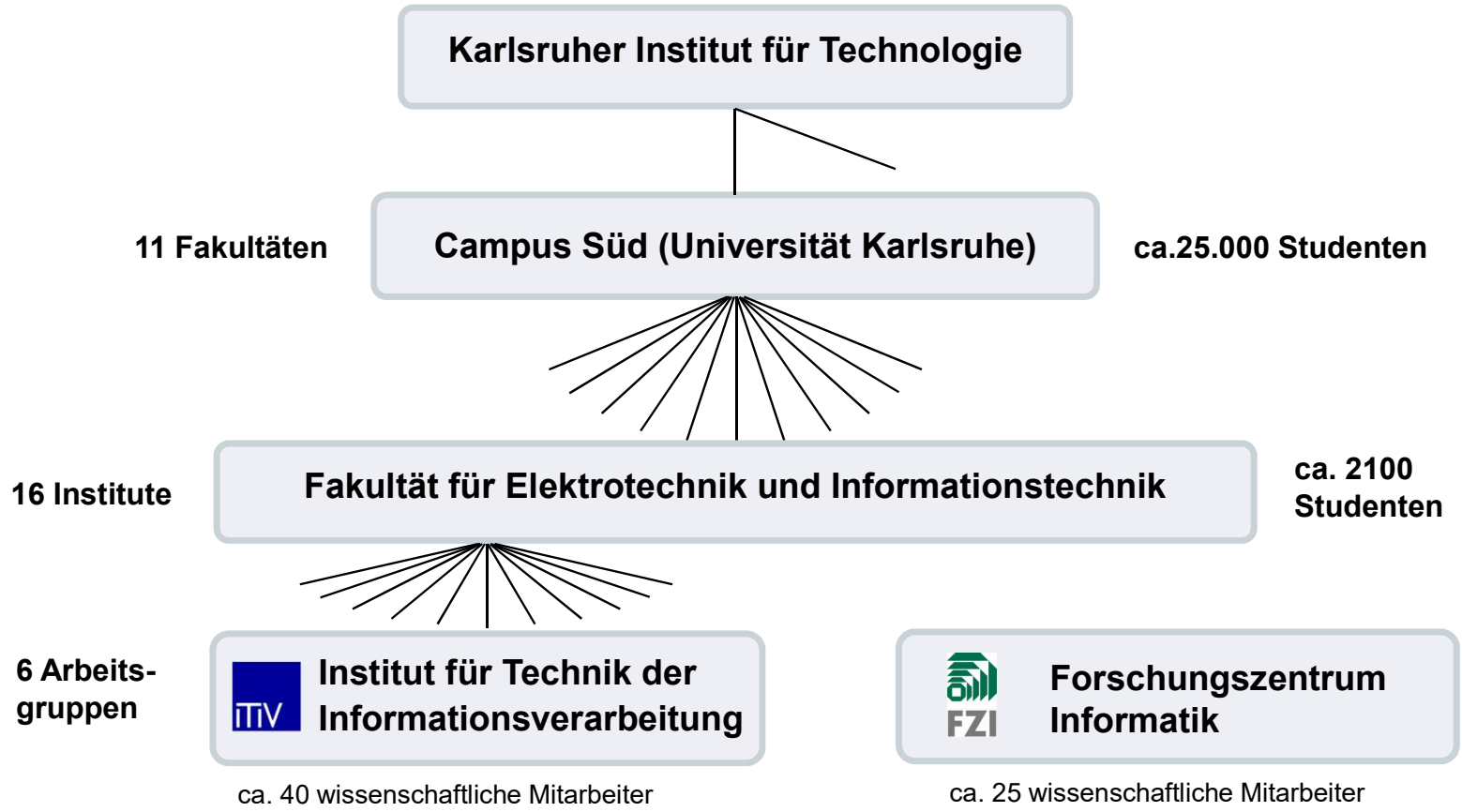








ITIV: Institut für Technik der Informationsverarbeitung



Doktoranden März, 2019



Karlsruhe



Um 1905 …



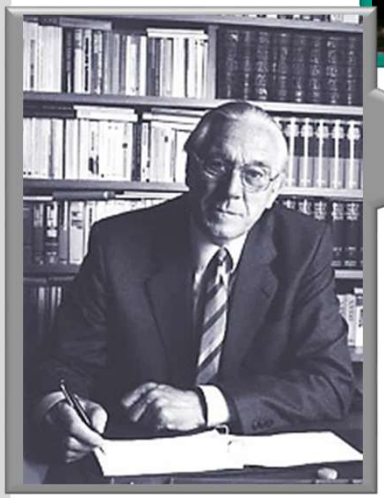






Um 1960 …







Nr. 267 – Mittwoch, 18. November 1998 – Seite 15

Fakultätentag trifft sich zum 25. in der Uni

Die Wiege der Informatik steht in der Fächerstadt

Auch der Name des Fachs stammt von Karl Steinbuch

Von unserem Redaktionsmitglied
Michael Nückel

„. . .mittels elektronischer Schaltungen in bisher unbekannter Geschwindigkeit Zahlenrechnung durchführen." Die Feuerzangenbowle? Nein. „Damit", so fährt der Sprecher fort, „begann die automatische Datenverarbeitung – wir nennen sie heute Informatik." Der „Sprecher" war kein geringerer als der renommierte Karlsruher Wissenschaftler Karl Steinbuch. Und er prägte bereits 1957 nicht nur den Begriff „Informatik", er war zusammen mit seinem Wissenschaftskollegen Karl Nickel der Gründungsvater der Karlsruher Informatik, die 1969 bundesweit erstmals als Institut und 1972 als Fakultät der Universität entstand.

„An dem guten Ruf und der führenden Stellung der Karlsruher Informatik-Fakultät [illegible] nichts verändert", sagt [illegible]ke. Klar, daß nach der [illegible]dung in Karlsruhe der [illegible]s neuen Studiengangs [illegible]iten am 20. November [illegible] Fächerstadt über die [illegible]s dahin noch namenlo- [illegible]Bezeichnung „Informa- [illegible]h das Festkolloquium [illegible]ntag Informatik" am [illegible]mber, 18 Uhr, findet [illegible]her Universität statt – [illegible]chen Informatik. Wo

[illegible]e Amerikaner die Nase [illegible]en 50ern nicht nur die [illegible]g in der Informations- [illegible] boten als erste auch [illegible]ungen für die junge [illegible]r Bezeichnung „Com- [illegible]rst 1969 forderte das [illegible] Bildung und Wissen- [illegible], Vorschläge zur Einrichtung von Forschungsgruppen einzubringen. Doch da verfügte die Karlsruher Fridericiana schon über ein langjähriges informatikbezogenes Lehrprogramm aus Mathematik, Physik und Elektrotechnik.

Nur so war es auch zu erklären, daß bereits im Frühjahr 1971, also kaum zwei Jahre nach der Gründung des Informatik-Instituts, die ersten Diplominformatiker Deutschlands die Fridericiana stolz verließen. Die „Seiteneinsteiger" machten es möglich. Sie hatten davor schon ein informatikorientiertes Grundstudium absolviert, jetzt studierten sie in dem neuen Fach zu Ende. Genau 115 Studierende begannen am 1. Januar 1969. Und im Nu war die Informatik gefragt wie kaum ein anderes Fach. Zunächst „vorfinanziert" durch die Universität, bis endlich das Geld vom Staat kam.

Wie aber der plötzlichen Nachfrage Herr werden? „Informatik ist zu einem Modestudium geworden", beklagte sich das frischgebackene Institut in einem Brief an „alle Höheren Schulen" und stellte warnend fest: „Informatik ist keine Ausbildung zum Programmierer". Auf der anderen Seite scheuten sich die Herren Professoren aber auch nicht, dann und wann einmal gut zwei Drittel der Studierenden bei den Prüfungen durchfallen zu lassen.

Was half's? 1992 platzte die Informatik mit dem Rekord von 2 500 Studis erneut aus allen Nähten, 500 bewarben sich neu. Danach sanken die Bewerberzahlen bis fast auf 200 und stiegen in diesem Semester erstmals wieder auf 283 an. Frauen? Auch sie gibt es mittlerweile unter den Informatikern, aber sie haben den Höchstanteil (1992) von neun Prozent nie überschritten. Immerhin gibt es den Arbeitskreis „Frauen in der Informatik". Und der ist auch beim Fakultätentag am Donnerstag vertreten. Die Festansprache zu dessen 25jährigem Bestehen über „Unternehmen im Info Age" hält aber selbstverständlich ein Mann.





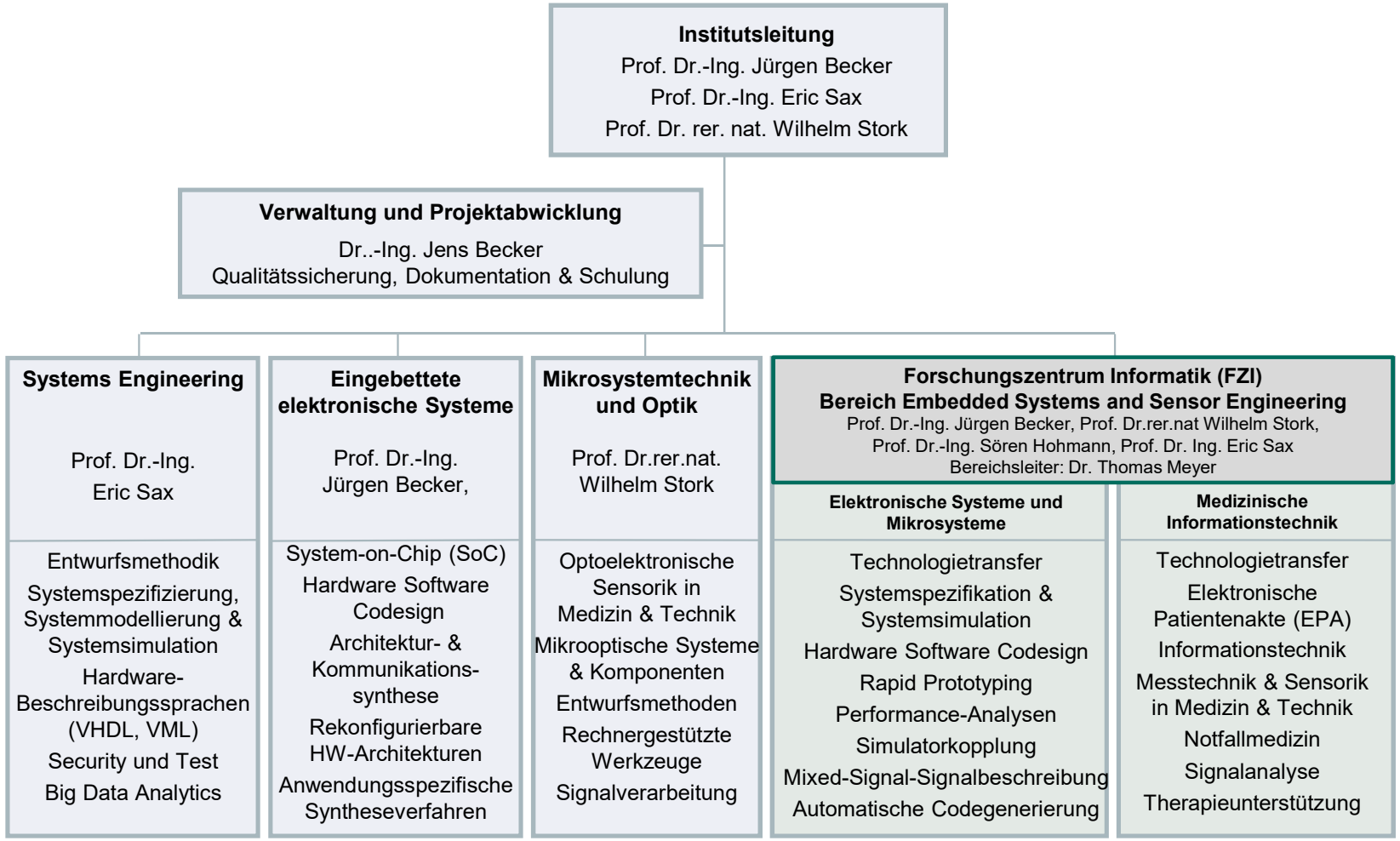


Um 1985: Forschungszentrum Informatik FZI





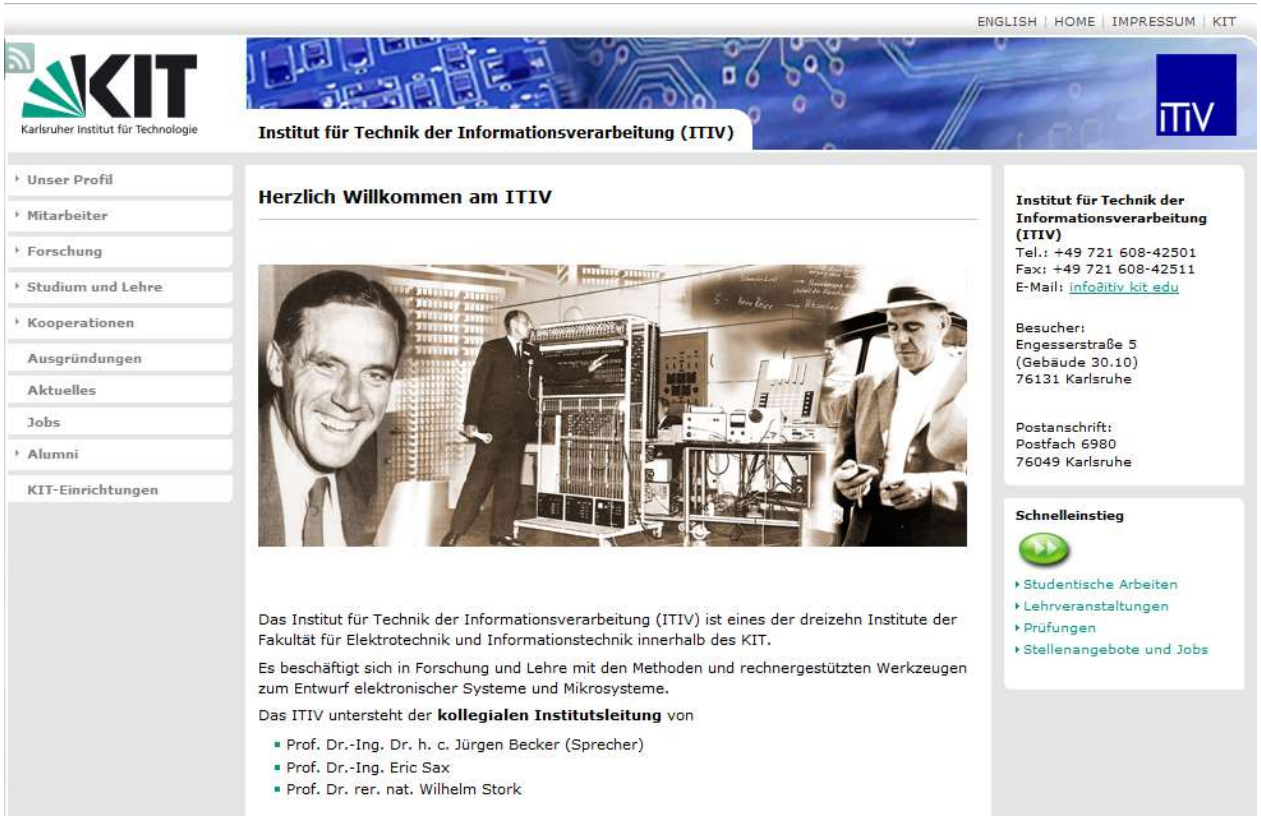
Organigramm ITIV



Das ITIV im Internet

- www.itiv.kit.edu

- Aktuelles
- Vorlesungsinformationen
- Abschlussarbeiten
- HiWi-Stellen



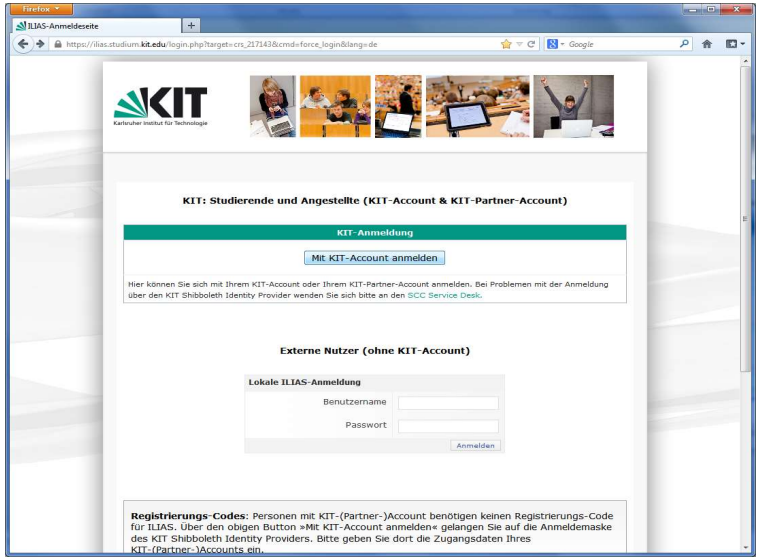
- Ab dem SS2019 werden die Veranstaltungen „2311622 Informationstechnik (IT)“ und „2311626 Praktikum Informationstechnik (PIT)“ nicht mehr angeboten.
- Stattdessen gibt es nun die Veranstaltungen „2311651 Informationstechnik I (IT1)“, „2311653 Informationstechnik I – Praktikum (PIT1)“ und „2311654 Informationstechnik II und Automatisierungstechnik (IT2)“.
- STO (Studienordnung) ETIT-2007 (ausgelaufen)
  - Es werden keine Prüfungen für IT und PIT mehr angeboten. Verwaltung lief in QISPOS.
- STO MIT-2012 (ausgelaufen)
  - Es werden keine Prüfungen für IT und PIT mehr angeboten. Verwaltung lief in QISPOS.
- STO ETIT-2015 (ausgelaufen)
  - Es werden noch folgende Prüfungen angeboten (Anmeldung über CAS):
    - IT:
      - WS 18/19, SS19, WS19/20 (nur für Wiederholer)
    - PIT:
      - Studierende können am aktuellen PIT1 teilnehmen
      - Zusätzlicher Aufgabenteil, um 3 LP zu erhalten
      - Prüfungen im SS19, SS20, SS21 (noch zu klären)
  - Übergangsregelung: Studierende, die IT und PIT nicht innerhalb der angebotenen Prüfungen bestehen konnten, müssen IT1, PIT1 und IT2 stattdessen prüfen, bekommen dafür nur die 7 LP von IT und PIT angerechnet.
  - Angebot: Studierende können IT2 prüfen und als interdisziplinäres Fach anrechnen lassen. Wenn sie das bisherige Modul IT bereits bestanden haben.

IT-Regelungen

- STO MIT-2016 (ausgelaufen)
  - Anmeldung über QISPOS. Prüfungsnummern:
    - IT: 969
    - PIT: 965
  - Es gelten die gleichen Regelungen wie für ETIT-2015.
- Studierende, die das alte IT-Modul bereits bestanden haben und in eine neue Prüfungsordnung wechseln, müssen IT1 nicht nochmal schreiben (in Diskussion).
- Studierende, die das alte IT-Modul bereits bestanden haben und nicht in eine neue Prüfungsordnung wechseln, können das Modul IT2 als „interdisziplinäres Fach“ anrechnen lassen (noch nicht final beschlossen in der SKOM).
- Andere Studiengänge
  - Wenn IT und PIT nicht durch IT1 und PIT1 und/oder IT2 ersetzt werden, gelten die gleichen Regelungen wie bei ETIT-2015.
  - Anmeldung über CAS oder blauer Schein.
- Auflagenprüfungen
  - Als Auflagenprüfungen sollten nur komplette Module also IT1 plus PIT 1 oder IT2 vergeben werden.

Unterlagen zu Informationstechnik II

- Nutzung der Lernplattform ILIAS
  - https://ilias.studium.kit.edu
  - Registrierung notwendig
    (falls noch nicht erfolgt)
  - Kurs: [2311654] Informationstechnik II und Automatisierungstechnik
    (SS 2019)
    - Passwort: ITII2019_DecisionTree
      (Groß-/Kleinschreibung beachten!)
- Inhalt
  - Vorlesungsfolien, Übungsblätter, Übungsfolien, Tutoriumsaufgaben, …



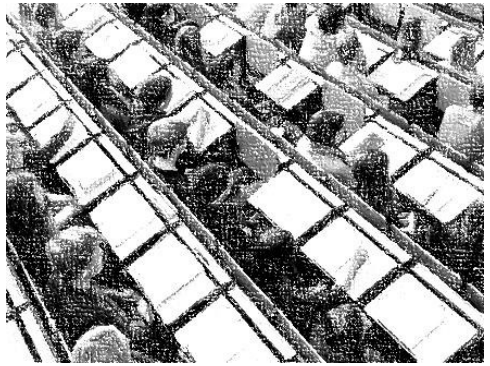
Magazin » Organisationseinheiten » Fakultät für Elektrotechnik und Informationstechnik » SS 2019 » [2311654] Informationstechnik II und Automatisierungstechnik (SS 2019)

[2311654] Informationstechnik II und Automatisierungstechnik (SS 2019)

Anmeldung zur Klausur

- Bachelor:
  - Verwendung der Selbstbedienungsfunktion des Studienbüros https://campus.studium.kit.edu/
- Andere oder nicht online freigeschaltete Studiengänge:
  - Direkte Anmeldung durch Abgabe des blauen Zettels oder Entsprechendem bei einem Betreuer (spätestens eine Woche vor der Prüfung)

Klausur (schriftlich)



- Inhalt: Vorlesung, Übung, Tutorium und Praktikum
- Termin: Donnerstag, 26.09.2019, 11:00 bis 13:00 Uhr
- Ort:
  - Voraussichtlich Fritz-Haller Hörsaal
- Dauer: 2 Stunden
- Mitzubringen: Fricard oder Lichtbildausweis und Schreibzeug
  - Nicht Bachelorstudenten zusätzlich: Studienzeitbescheinigung
- Hilfsmittel: 1x A4 Blatt (2x A4 Seiten, handschriftlich, lesbar ohne Hilfsmittel)
- Hörsaalverteilung (nach Nachnamen, bei Doppelnamen ist der erste Nachname maßgebend)
  - Details werden vor der Prüfung bekannt gegeben.

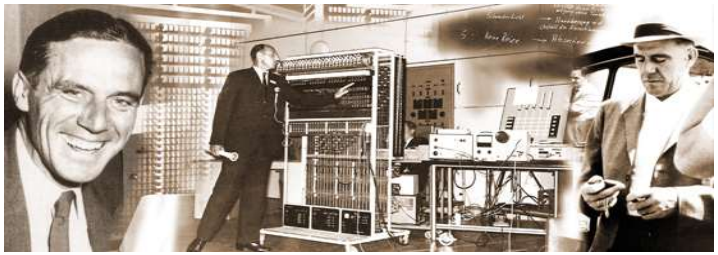
Inhalt IT1

1. **Einleitung und Motivation**
   - Organisatorisches
   - Begriffe
   - Typische Anwendungen
2. **Programmiersprachen**
   - Software-Komponenten
   - Was ist eine Programmiersprache?
   - Formale Sprachen
   - Höhere Sprachen
   - Programmierparadigmen
   - Wichtige Programmiersprachen
   - Elemente einer Programmiersprache
   - Weg zum ausführbaren Programm



Inhalt IT1

3. **Programmstrukturen**
   - Programmablaufpläne
   - Nassi-Shneiderman-Diagramme
   - Gegenüberstellung
4. **Objektorientierung**
   - Grundidee und Motivation
   - Klassen
   - Objekte
   - Datenkapselung
   - Vererbung
   - Polymorphie
5. **Betriebssysteme**
   - Threads
   - Zustände
   - Weg zum ausführbaren Programm



Inhalt IT1

6. Rechnerarchitekturen

- Einführung
- Allgemeiner Aufbau der internen Hardware Architektur
- Instruction Set Architecture (ISA) am Beispiel AVR8
- Klassifikation von Rechnerarchitekturen
- Performanzsteigerung
- ausgewählte Beispiele

.



Inhalt IT1

7.1 Datenstrukturen

- Array
- Liste
- Stack
- Queue
- Hash-Tabelle
- Baum
- Haufen

7.2 Datenbanken

- Motivation
- Begriffsklärung: Datenbank, Datenbankmanagementsystem und Datenbanksystem
- Nähere Erläuterung:
  - Datenbanksystem
  - Datenbank
  - Datenbankmanagementsystem
- Datenbankmanagementsysteme
  - SQL
  - Oracle

Inhalt IT1

8. Prozesse, Methoden, Tools

- Systems Engineering
- Software-Entwicklung
- Testing

9. Projektmanagement

- Definitionen und Begriffe
- Motivation für Projektmanagement
- Grundsätze
- Organisation (Projekt-/Zeitplan, Arbeitspakete, Meilensteine)
- Qualitäts- und Validierungsplan
- Fortschritts-Tracking
- Präsentation und Erstellung von Berichten
- Risikomanagement
- Projektabschluss



Inhalt IT2

1. Einführung
- Aufbau Modul IT 2 / Verbindung zu IT1 und anderen Lehrveranstaltungen
- Anwendungsbeispiele und Motivation
- Definitionen und Begriffe

2. Algorithmische Grundlagen/Eigenschaften
- Definition Algorithmus et al.
- Klassifikation von Algorithmen (Komplexität, Problemstellung, Anwendung)
- Merkmale
- Algorithmen-Analyse (Laufzeit, Effizienz, Effektivität)

3. Sortieralgorithmen
- Bubble Sort
- Insertion Sort
- Merge Sort
- Quick Sort



Inhalt IT2

4. Algorithmen auf Graphen

- Graphendefinition
- Binäre Suche
- Breitensuche
- Tiefensuche
- Zyklensuche
- Kürzester und kritischer Pfad

5. Optimierungsalgorithmen

- Partitionierung
- Anlagerungsverfahren
- Random Interchange
- Kernighan-Lin
- Greedy
- Simulated Annealing
- Evolutionäre Algorithmen



Inhalt IT2

6. Selbstlernende Algorithmen

- Statistik
- Evolutionäre Algorithmen (bisher Optimierungsalgorithmen)
- Decision Trees
- Clustering
- Neuronale Netze (MLP, RNN, CNN)

7. Big Data

- Definition, Verwendung (Charakterisitik, Risiken, Chancen)
- Hadoop, NoSQL, MapReduce (sehr schwierig, hier eine Wdh von Datenbanken aus IT1)
- Anwendungsfälle
- Big Data Prozesse:
  - Crisp, CDD
  - Data Science Process



Inhalt IT2

8. Anomalie-Erkennung

- Definition Anomalie, wozu benötigt?
- Verschiedene Anwendungsgebiete
- Verschiedene Methoden →Selbstlernende Verfahren
  - Clustering
  - MLP
- Anwendung bei Big Data

9. Cyber Security, Datenschutz

- Definition, Begriffe, Hacking
- Angreifertypen- und ziele, Angriffsziele
- Schutzmechanismus
  - Kryptographie
  - Firewall
  - Authentifizierung
  - Intrusion detection



Inhalt IT2

10. Automatisierung

To define



IT-Veranstaltungsplan SS2018 → neu IT I + II

Veranstaltungsplan Informationstechnik SS2017

Vorlesung/Übung:	Mi. 09.45-11.15	10.21 Benz (HMU)
Vorlesung/Übung:	Do. 14.00-15.30	30.46 Neue Chemie
Praktikum:	29. Mai - 28. Juli	

Woche	Datum Mi.	Vorlesung	Datum Do.	Übung	Praktikum	Inhalt Mi.	Inhalt Do.
1	26. Apr	VL	27. Apr	VL		VL0: Einleitung und Motivation	VL1: Rechnerarchitekturen
2	03. Mai		04. Mai	VL			VL2: Rechnerarchitekturen
3	10. Mai	VL	11. Mai	ÜB		VL3: Rechnerarchitekturen	ÜB1: Organisatorisches; Besprechung Aufg. 1; Variablen & Gültigkeitsbereich, Operatoren, Arrays, Kontrollstrukturen
4	17. Mai	VL	18. Mai	ÜB	Einführungsveranstaltung	VL4: Programmiersprachen, Weg zum ausführbaren Programm	ÜB2: Besprechung Aufg. 2; Funktionen, Zeiger & Header
5	24. Mai	VL	25. Mai	frei	Zusatz: HW-nahe Programmierung	VL5: Programmstrukturen	+ mehr Programmierinhalte
6	31. Mai	VL	01. Jun	VL	Projektmanagement	VL6: SW-Engineering	VL7: Objektorientierung
7	07. Jun	frei	08. Jun	frei		Pfingsten	Pfingsten
8	14. Jun	ÜB	15. Jun	frei		ÜB3: Besprechung Aufg. 3; Objektorientierung & Dynamische Speicherverwaltung	Fronleichnam
9	21. Jun	VL	22. Jun		Programmieren	VL8: Datenstrukturen	+ Big Data Analytics
10	28. Jun	VL	29. Jun	ÜB	Programmieren	VL9: Algorithmen	+ Selbstlernende Systeme/Funktionen
11	05. Jul	VL	06. Jul	ÜB	Programmieren	VL10: Sortieralgorithmen	ÜB5: Besprechung Aufg. 5; Dateiverarbeitung & STL
12	12. Jul	VL	13. Jul	ÜB	Test	VL11: Suchalgorithmen	ÜB6: Besprechung Aufg. 6; Sortieralgorithmen
13	19. Jul	VL	20. Jul		Test	VL12: Optimierungsalgorithmen	
14	26. Jul	VL	27. Jul	ÜB	Probefahrt	VL13: Puffer	+ Anomalie-Erkennung

IT 1

IT 2

VL	14
UB	7
Praktikum	9

Einführungsveranstaltung I Di. 16.05.2017, 14:00-15:30, HSaF
Einführungsveranstaltung II Di. 23.05.2017, 14:00-15:30, HSaF

KW= Kalenderwoche; SW = Semesterwoche; VL = Vorlesung; ÜB = Übung; T= Tutorium; P= Praktikum; T4 u. T5 wertvoll für P

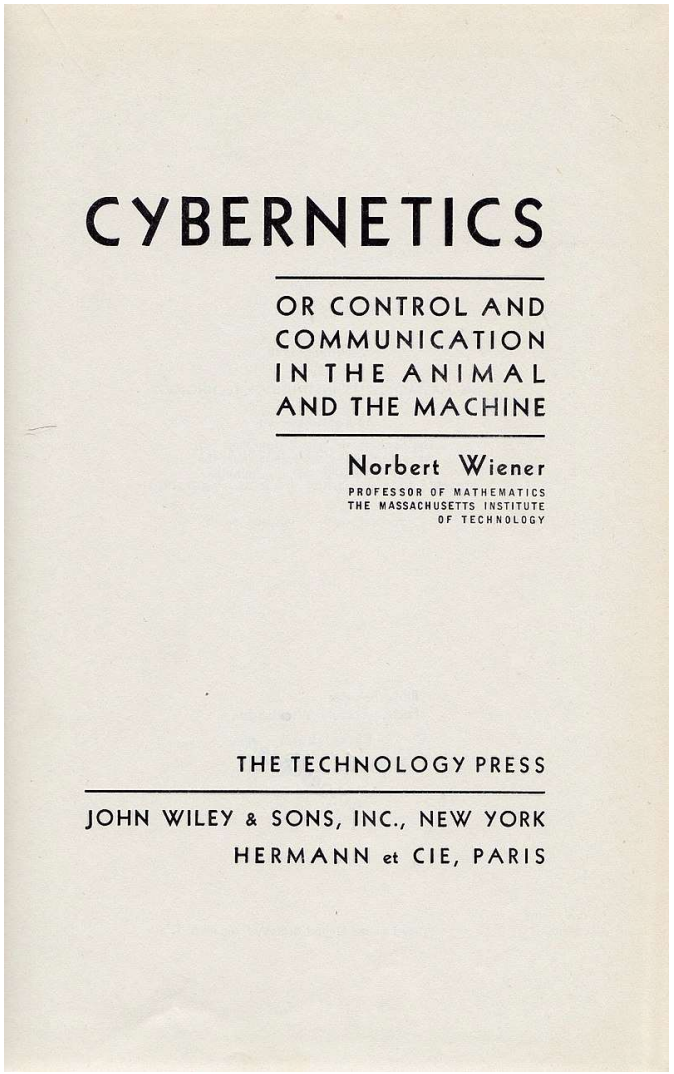
Veranstaltungsplan Informationstechnik 2 SS2019

Vorlesung/Übung:	Mi, 8:00 - 9:30 Uhr	Gebäude 30.22, Gaede-Hörsaal
Vorlesung/Übung:	Fr, 9:45 - 11:15 Uhr	Gebäude 30.22, Gaede-Hörsaal

Woche	Datum Mi.	Vorlesung	Datum Fr.	Übung	Inhalt Mi.	Inhalt Fr.
1.	24. Apr		26. Apr	VL		VL0: Einleitung und Motivation
2.	01. Mai	frei	03. Mai	VL	Tag der Arbeit	VL1: Algorithmische Grundlagen
3.	08. Mai	ÜB	10. Mai	VL	ÜB1: Organisatorisches; Besprechung Aufg. 1;	VL2: Sortieralgorithmen
4.	15. Mai		17. Mai	VL		VL3: Algorithmen auf Graphen
5.	22. Mai	ÜB	24. Mai	VL	ÜB2: Besprechung Aufg. 2;	VL4: Optimierungsalgorithmen
6.	29. Mai		31. Mai	VL		VL5: Big Data Einführung
7.	05. Jun	ÜB	07. Jun	VL	ÜB3: Besprechung Aufg. 3;	VL6: Big Data - Prozesse
8.	12. Jun	frei	14. Jun	frei	Pfingsten	Pfingsten
9.	19. Jun	ÜB	21. Jun	VL	ÜB4: Besprechung Aufg. 4;	VL7: Big Data - Modelling 1
10.	26. Jun	ÜB	28. Jun	VL	ÜB5: Besprechung Aufg. 5;	VL8: Big Data - Modellin 2
11.	03. Jul		05. Jul	VL		VL9: Big Data - Evaluation
12.	10. Jul	ÜB	12. Jul	VL	ÜB6: Besprechung Aufg. 6;	VL10: Big Data - Infrastruktur
13.	17. Jul	ÜB	19. Jul	VL	ÜB7: Besprechung Aufg. 7;	VL11: Anwendungen Institut
14.	24. Jul	ÜB	26. Jul	VL	ÜB8: Besprechung Aufg. 8;	VL12: Cyber Security

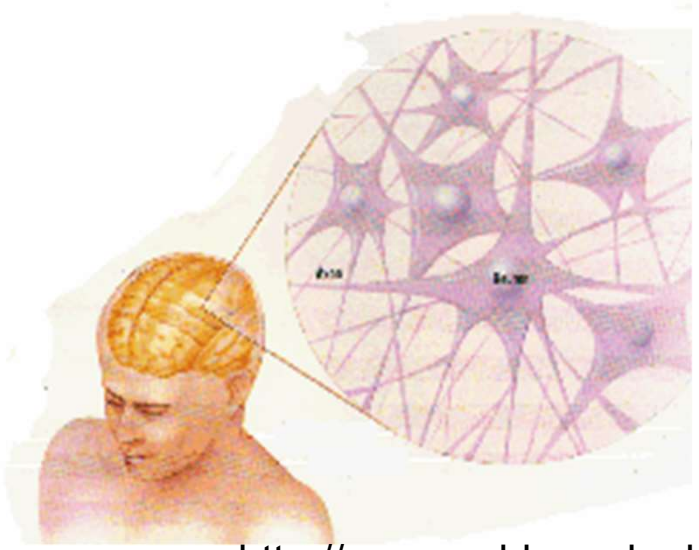
Kybernetik

- Nach Norbert Wiener, 1948
- Seit 1953 Lehrfach von Georg Klaus an der Humboldt-Universität Berlin
- Kommunikationswissenschaft, die sich mit der Übertragung und Steuerung von Informationen im weitesten Sinn befasst.
- Geht auf das griechische Wort für Steuermann (nach Homer) zurück
- Synonym: Allg. Informationstheorie



Neuronales Netz

- Hirn:
  - 10 bis 100 Mrd. Neuronen mit je 1.000 bis 10.000 Nachbarn
  - 1.000 Impulse pro Sekunde
- Neuronen wechseln Ihre Zustände in Abhängigkeit von der Summe der momentan anliegenden Verbindungsgewichte.
- Ein Zustandswechsel führt beim Überschreiten eines Schwellwertes zur Weitergabe eines Wertes an ein Nachbar-Neuron

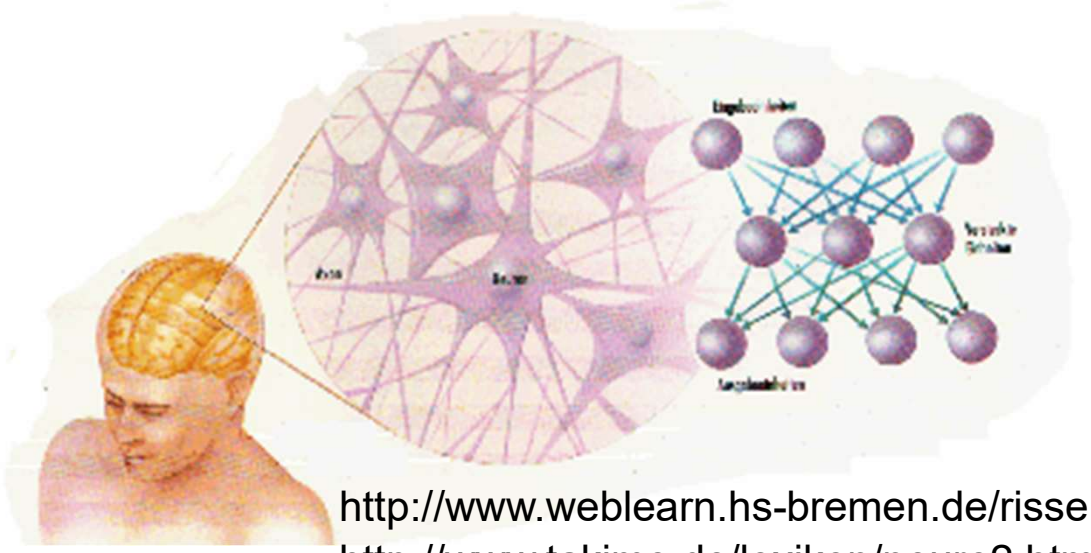


http://www.weblearn.hs-bremen.de/risse

http://www.takimo.de/lexikon/neuro2.html

Künstliches Neuronales Netz

- Wissensverarbeitendes System, das sich am menschlichen Hirn orientiert.
- Werden nicht im herkömmlichen Sinn durch ein Programm zu einem Verhalten angeleitet → sie werden mittels Testdaten trainiert!
- Realisiert durch Multiprozessor-Systeme mit einer sehr großen Anzahl sehr einfacher Prozessoren
  - Jeder Prozessor modelliert dabei ein Neuron.
- Das Perzeptron ist ein vereinfachtes künstliches neuronales Netz, das zuerst von Frank Rosenblatt 1958 vorgestellt wurde.
  - Es besteht in der Grundversion aus einem einzelnen künstlichen Neuron mit anpassbaren Gewichtungen und einem Schwellenwert.



http://www.weblearn.hs-bremen.de/risse
http://www.takimo.de/lexikon/neuro2.html

Künstliche Intelligenz
Das Gehirn als Vorbild

- Wenn von künstlicher Intelligenz die Rede ist, werden Daten wie in einem künstlichen neuronales Netz verarbeitet.
- Es handelt sich also um reine Software.
  - Die kann auch auf einem herkömmlichen Computer laufen, ist aber wesentlich leistungsstärker, wenn die Hardware für diese spezifische Form der Datenverarbeitung optimiert ist.
- Mittlerweile sind entsprechende Prozessoren erhältlich
  - Googles Tensor Flow Processing Units (TPU)
  - Bei Smartphones ist Huawei mit der Neural Processing Unit (NPU) auf dem aktuellen Chipsatz Kirin 970
    - Ihre Architektur ist an einem klassischen Grafikprozessor (GPU) orientiert, weil dieser für die parallele Berechnung vieler kleiner Einheiten optimiert ist.

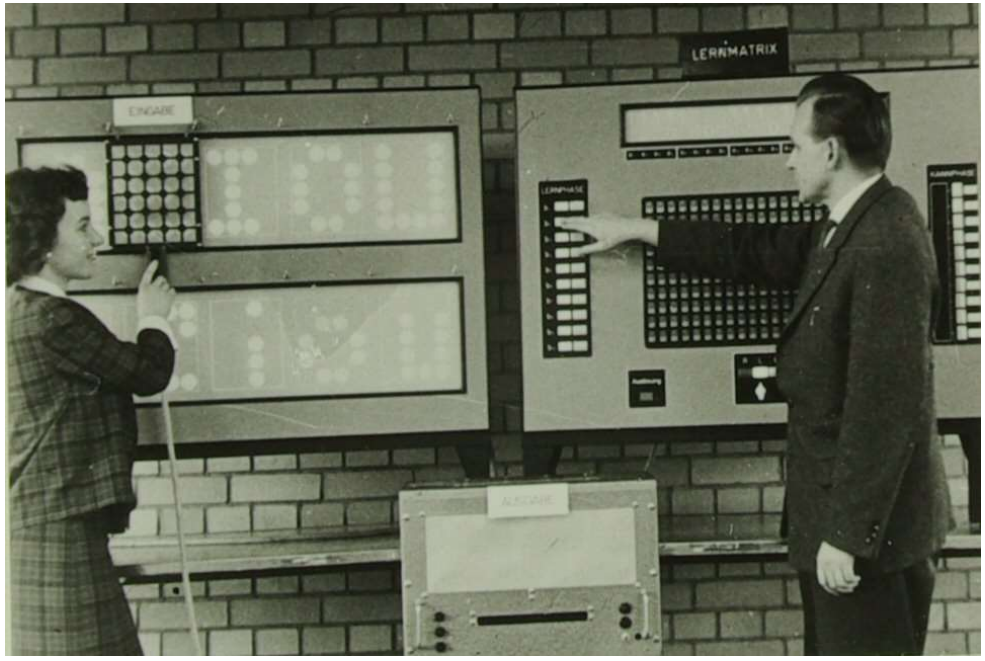
- Hinter KI stehen Konzepte der Informationsverarbeitung, die mehr als 50 Jahre alt sind und sich in ihrer Struktur am menschlichen Gehirn orientieren.
  - Sie unterscheiden sich grundlegend von einer klassischen Softwarearchitektur, die eintreffende Daten sequenziell nach dem immer gleichen Muster verarbeitet.
  - Diese liefert Ergebnisse mit hundertprozentiger Präzision, aber sie setzt Genauigkeit vor Geschwindigkeit und das Verarbeitungsschema ist statisch.




20th
CENTURY
FOX
THE
STAR
WARS
TRILOGY
.COM

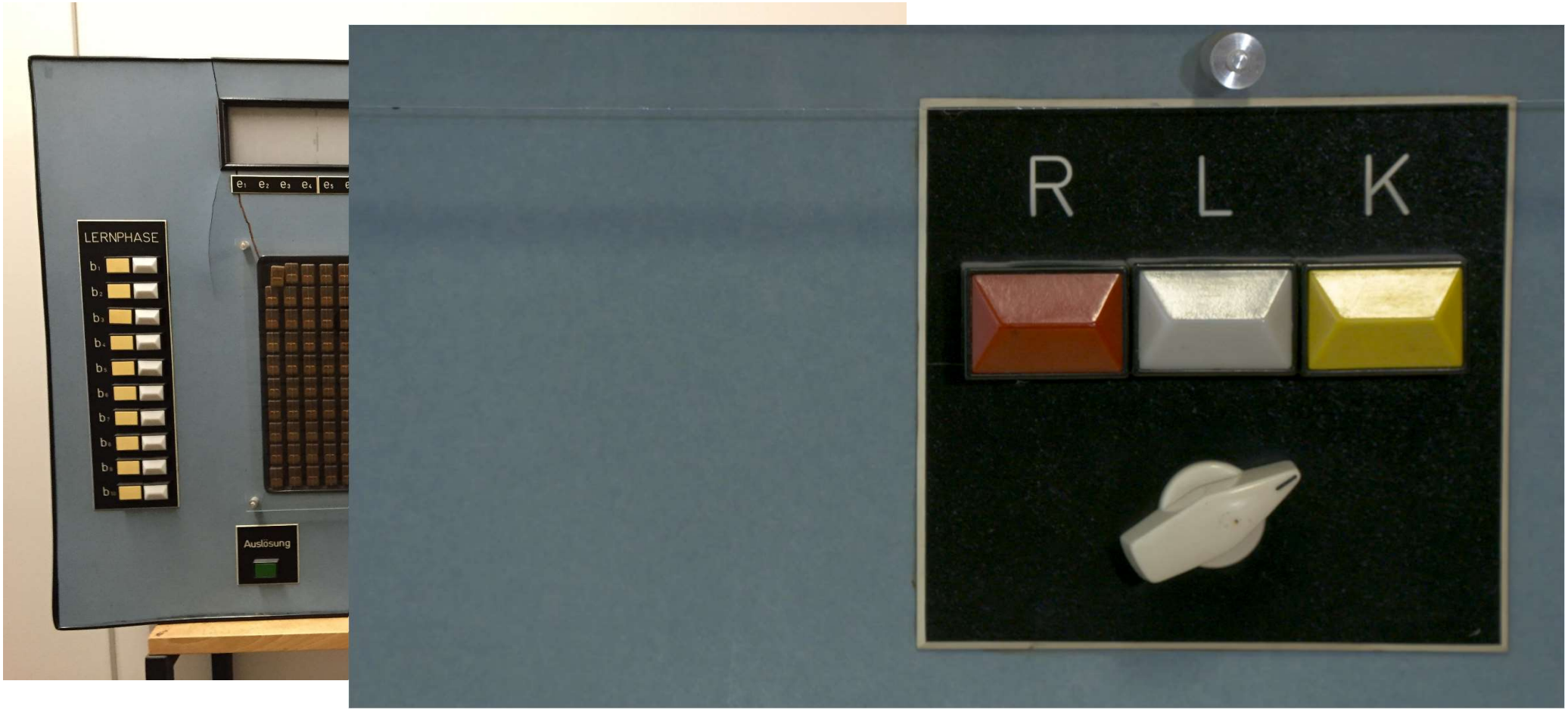
Lernmatrix

- Die Lernmatrix ist ein besonderer Typ eines künstlichen Neuronalen Netzes, die vom Informatik- und KNN-Pionier Karl Steinbuch um 1960 erfunden wurde.





Lernmatrix



Maschinelles Lernen

- Machine Learning, im Deutschen maschinelles Lernen, ist ein Teilgebiet der künstlichen Intelligenz.
  - Durch das Erkennen von Mustern in vorliegenden Datenbeständen sind IT-Systeme in der Lage, eigenständig Lösungen für Probleme zu finden.
  - Machine Learning ist ein Teilbereich der künstlichen Intelligenz.
  - Mithilfe des maschinellen Lernens werden IT-Systeme in die Lage versetzt, auf Basis vorhandener Datenbestände und Algorithmen Muster und Gesetzmäßigkeiten zu erkennen und Lösungen zu entwickeln.
  - Es wird quasi künstliches Wissen aus Erfahrungen generiert.
  - Die aus den Daten gewonnenen Erkenntnisse lassen sich verallgemeinern und für neue Problemlösungen oder für die Analyse von bisher unbekannten Daten verwenden.

Maschinelles Lernen

- Damit die Software eigenständig lernen und Lösungen finden kann, ist ein vorheriges Handeln von Menschen notwendig.
  - Beispielsweise müssen die Systeme zunächst mit den für das Lernen relevanten Daten und Algorithmen versorgt werden.
  - Zudem sind Regeln für die Analyse des Datenbestands und das Erkennen der Muster aufzustellen.
  - Sind passende Daten vorhanden und Regeln definiert, können Systeme mit maschinellem Lernen unter anderem folgendes:
    - Relevante Daten finden, extrahieren und zusammenfassen
    - Vorhersagen auf Basis der analysierten Daten treffen
    - Wahrscheinlichkeiten für bestimmte Ereignisse berechnen,
    - sich an Entwicklungen eigenständig anpassen
    - Prozesse auf Basis erkannter Muster optimieren.



Die verschiedenen Arten des Machine Learnings

- Algorithmen nehmen beim maschinellen Lernen eine zentrale Rolle ein. Sie sind für das Erkennen von Mustern und das Generieren von Lösungen verantwortlich und lassen sich in verschiedene Lernkategorien einteilen.
  - überwachtes Lernen
  - unüberwachtes Lernen
  - teilüberwachtes Lernen
  - bestärkendes Lernen
  - aktives Lernen
- Während beim überwachten Lernen im Vorfeld Beispielmodelle definiert und spezifiziert werden müssen, um die Informationen passend den Modellgruppen der Algorithmen zuzuordnen, werden die Modellgruppen beim unüberwachten Lernen automatisiert aufgrund eigenständig erkannter Muster gebildet.

Die verschiedenen Arten des Machine Learnings

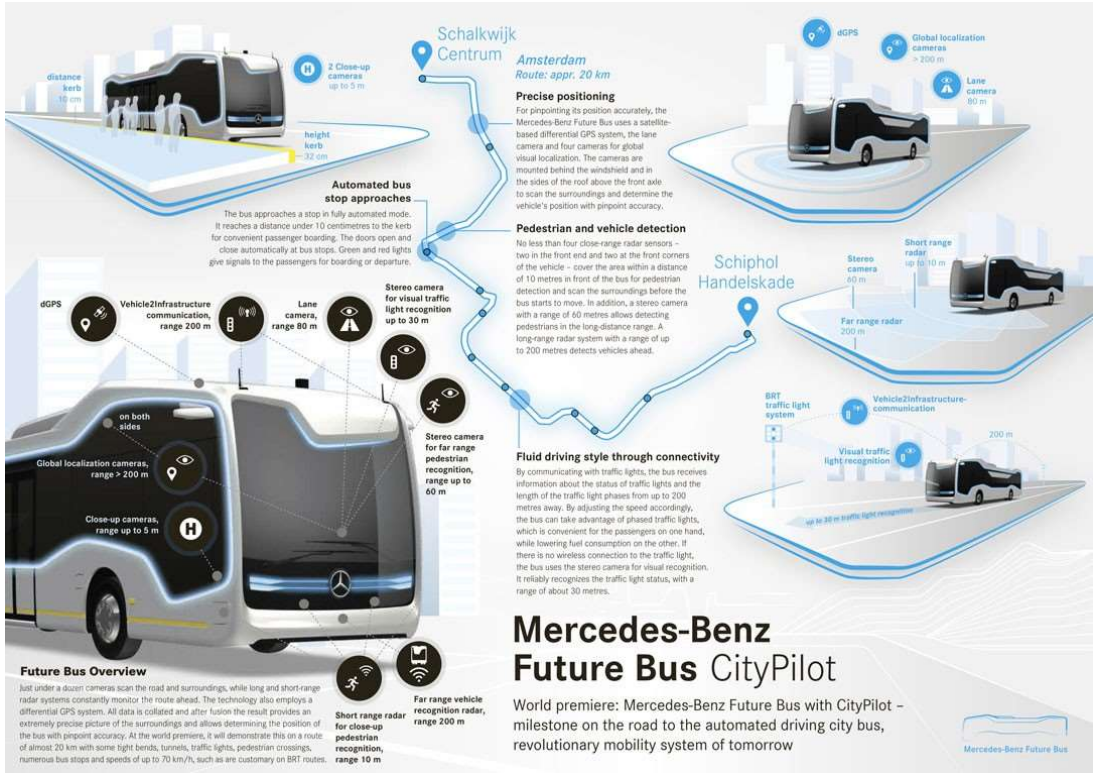
- Teilüberwachtes Lernen stellt eine Mischung aus beiden Methoden dar. Das bestärkende Lernen basiert auf Belohnungen und Bestrafungen.
  - Dem Algorithmus wird durch diese Interaktion mitgeteilt, wie er auf verschiedene Situationen zu reagieren hat.
  - Diese Lernweise ist dem menschlichen Lernen sehr ähnlich.
- Aktives Lernen schließlich bietet dem Algorithmus die Möglichkeit, für bestimmte Eingangsdaten die gewünschten Ergebnisse zu erfragen.
  - Um die Anzahl von Fragen zu minimieren, erfolgt zuvor eine Auswahl relevanter Fragen mit hoher Ergebnisrelevanz durch den Algorithmus selbst.
- Abhängig vom jeweiligen System kann die Datenbasis offline oder online vorliegen und wiederholbar oder nur einmalig für das maschinelle Lernen zur Verfügung stehen.
  - Ein weiteres Unterscheidungsmerkmal des Machine Learnings ist das gleichzeitige Vorhandensein der Ein- und Ausgabe-Paare oder deren zeitlich versetzte Entwicklung.
  - Je nach Art spricht man vom Batch-Lernen oder vom sequenziellen Lernen.

Deep Learning

- Abgrenzung des Deep Learnings vom rein maschinellen Lernen
  - Deep Learning ist ein Teilbereich des Machine Learnings
  - Der entscheidende Unterschied besteht darin, dass beim maschinellen Lernen der Mensch in die Analyse der Daten und den eigentlichen Entscheidungsprozess eingreift.
  - Beim Deep Learning sorgt der Mensch lediglich dafür, dass die Informationen für das Lernen bereitstehen und die Prozesse dokumentiert sind.
  - Die eigentliche Analyse und das Ableiten von Prognosen oder Entscheidungen überlässt er der Maschine selbst.
  - Der Mensch hat keinen Einfluss auf die Ergebnisse des Lernprozesses. Es lässt sich im Nachhinein nicht mehr vollständig zurückverfolgen, auf Basis welcher genauen Muster eine Maschine eine bestimmte Entscheidung getroffen hat.
  - Zudem werden die Entscheidungen ständig hinterfragt und die Entscheidungsregeln selbstständig optimiert.
- Deep Learning lehrt Maschinen zu lernen.
  - Die Maschine wird in die Lage versetzt, selbstständig und ohne menschliches Zutun ihre Fähigkeiten zu verbessern.
  - Das erreicht man, indem aus vorhandenen Daten und Informationen Muster extrahiert und klassifiziert werden.
  - Die gewonnenen Erkenntnisse lassen sich wiederum mit Daten korrelieren und in einem weiteren Kontext verknüpfen.
  - Schließlich ist die Maschine fähig, Entscheidungen auf Basis der Verknüpfungen zu treffen.

Referenz: Future Bus Amsterdam

- Vollautomatisiertes Fahren im Shuttle-Betrieb
- 3 Tage im regulären Service zwischen Schiphol und Amsterdam/Harlem
- Spurführung durch KIT (FZI)



Referenz: Automatisierter Betriebshof

- Projekte mit Verkehrsbetrieben in Stuttgart, Potsdam und München
- Fahr-Modellerstellung zur Demonstration der Automatisierungsidee in 1:87





Referenz: Masterarbeit im Europapark (Rust)

- Automatisiertes Fahren als Besucherspaß!

MA
Masterarbeit

Umfeld:

Der Europapark ist der größte Freizeitpark Europas und besteht seit über 40 Jahren. Neben neuesten Attraktionen gibt es folglich auch die ein oder andere Bahn, die etwas in die Jahre gekommen ist. Dazu zählt ein Fahrspaß, der es erlaubt mit mechanisch geführten Fahrzeugen Rennen zu fahren. Zu untersuchen gilt es nun, inwieweit automatisiertes Fahren mit neuester Sensor- und Kommunikationstechnik einen Ersatz für die Mechanik erlaubt. Alternative Lösungsideen, die nach Machbarkeit und Wirtschaftlichkeit zu untersuchen sind, sollten am Ende der Arbeit in ein umsetzbares Konzept münden.





Aufgabe:

- Erfassung und Gegenüberstellen der einsetzbaren Technologien (Sensoren etc.)
- Konzepterstellung (Nutzungsmöglichkeit bestehender Fahrzeuge, Besucher-Kapazität, Praktikabilität,…)
- Darstellung der möglichen Fahrmanöver
- Vergleich Vor- und Nachteile zu bestehenden Systemen (klassisch ohne automatisiertes Fahren)
- Betriebswirtschaftlichkeitsberechnung, ggf. auch lediglich Kostenaufstellung
- Ermittlung der technischen und operativen Voraussetzungen (z.B. Anforderungen an Wartung, Attraktionsmitarbeiter usw.)

Kapazität, Praktikabilität,…)
- Darstellung der möglichen Fahrmanöver
- Vergleich Vor- und Nachteile zu bestehenden Systemen (klassisch ohne automatisiertes Fahren)
- Betriebswirtschaftlichkeitsberechnung, ggf. auch lediglich Kostenaufstellung
- Ermittlung der technischen und operativen Voraussetzungen (z.B. Anforderungen an Wartung, Attraktionsmitarbeiter usw.)

Voraussetzungen:

- Endphase eines Masterstudiums in Info, ETIT, WiWi o.ä.
- Erfahrung im Umgang mit Hardware (Sensoren, HW-Plattformen) und erste Programmiererfahrung
- Eigenständige und lösungsorientierte Arbeitsweise
- Bereitschaft zur Arbeit und Analyse auch vor Ort im Europapark

Bei Interesse bitte melden bei: **eric.sax@kit.edu**, Raum: 326

Referenz: Durchführbarkeitsstudie INTERACt
Autonomes Fahren in Terminalbereichen

- Motivation:
- Derzeitige Herausforderungen im Transportsektor:
  - gesteigerter Wettbewerb
  - erhöhter Kostendruck
  - steigendes Verkehrsaufkommen
  - Mangel an qualifiziertem Fahrpersonal
  - erhöhte Anforderungen und Bewusstsein für Verkehrssicherheit und Umweltschutz
- Autonome Lkw können einen Beitrag zur Bewältigung dieser Herausforderungen leisten.
- Inhalt:
  - Analyse welche technischen, operativen und rechtlichen Anforderungen an Fahrzeuge, Transportdienstleister und Terminals zu stellen sind
  - Überprüfung inwieweit autonom fahrende LKWs in den Containertransportablauf integriert werden können

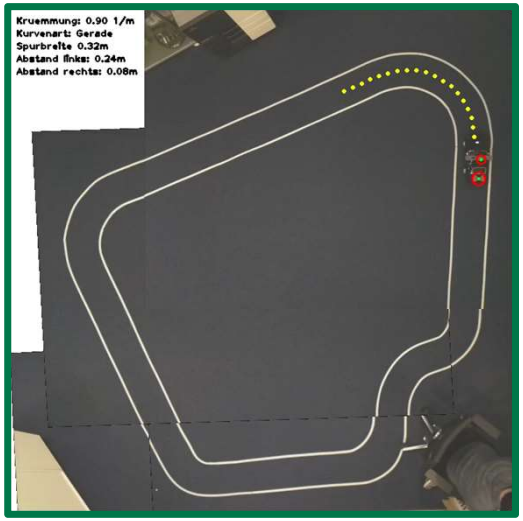


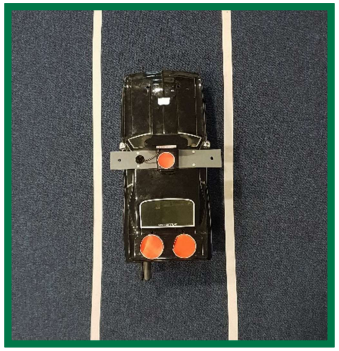




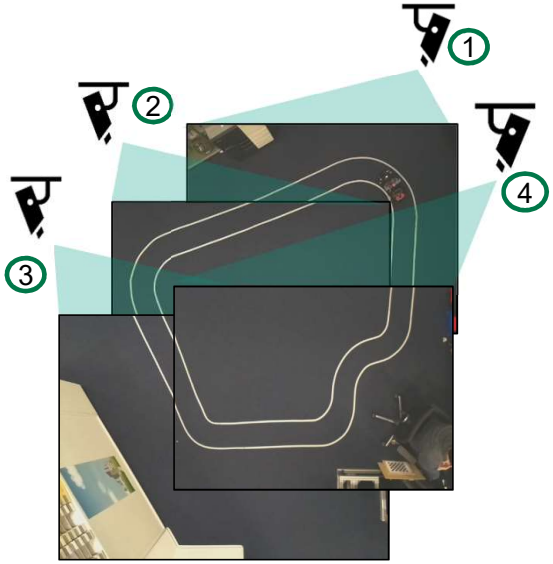
Demonstrator Teststrecke

- Die Teststrecke ist mit 4 Raspberry Pi Deckenkameras ausgestattet
- Die 4 einzelnen Bilder werden mit einem Stitching-Algorithmus zusammengefügt
- Das Demonstrator Fahrzeuge wird über farbige Marker lokalisiert





Demonstrator mit roten Markern



Remote Access





10 Jahre später …

- **Aufteilung**
- **Klausur**
- **Organisatorisches**



Backup

Anwendungen Big Data und selbst lernende Systeme



Karlsruher Institut für Technologie

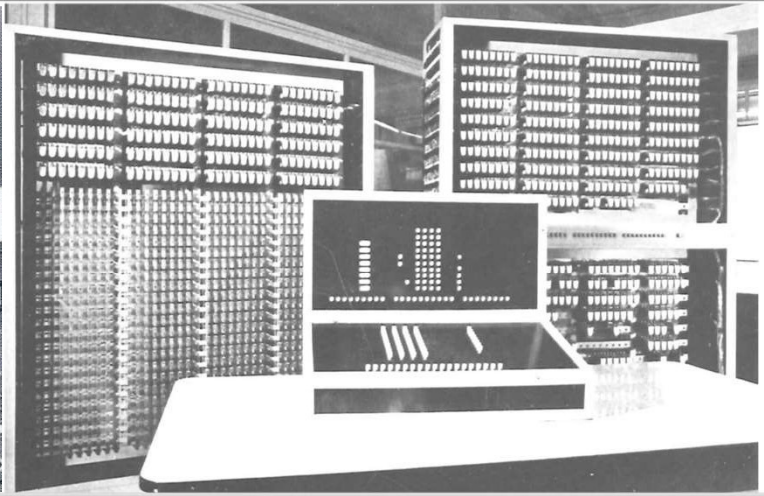
Institut für Technik der
Informationsverarbeitung

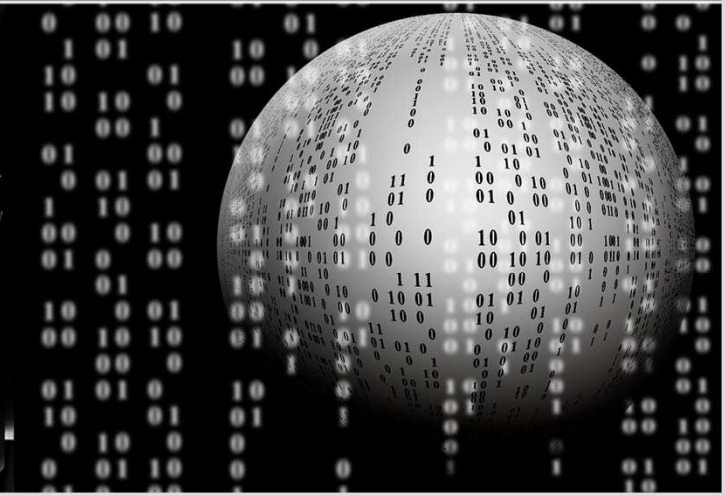
Institutsleitung
Prof. Dr.-Ing. J. Becker
Prof. Dr.-Ing. E. Sax
Prof. Dr. rer. nat. W. Stork

Algorithmische Grundlagen/Eigenschaften

Informationstechnik II

Prof. Dr.-Ing. Eric Sax



Inhalt IT2

1. Einführung

- Aufbau Modul IT 2 / Verbindung zu IT1 und anderen Lehrveranstaltungen
- Anwendungsbeispiele und Motivation
- Definitionen und Begriffe

Algorithmische Grundlagen/Eigenschaften

- Definition Algorithmus et al.
- Klassifikation von Algorithmen (Komplexität, Problemstellung, Anwendung)
- Merkmale
- Algorithmen-Analyse (Laufzeit, Effizienz, Effektivität)

3. Sortieralgorithmen

- Bubble Sort
- Insertion Sort
- Merge Sort
- Quick Sort



Definition: Algorithmus

- Wer war Musa al-Chwarizimi? (أبو جعفر محمد بن موسى الخوارزمي)
  - Auf ihn geht der Begriff Algorithmus zurück.
  - Er führte Ziffer "null" aus dem indischen in das arabische Zahlensystem ein.
- Algorithmen sind …
  - … genau definierte Handlungsvorschriften zur Lösung eines Problems oder einer bestimmten Art von Problemen in endlich vielen Schritten
- Beispiele:
  - Kochrezept
  - Gebrauchsanleitungen
  - Hilfen zum Ausfüllen von Formularen
  - Waschmaschinenprogramme



Denkmal des Abu Dscha'far Muhammad ibn Musa al-Chwarizmi

Definition: Algorithmus

- Algorithmus im engeren Sinne der Informatik
  - Gegenstand einiger Spezialgebiete der Theoretischen Informatik (Algorithmentheorie, Komplexitätstheorie, Berechenbarkeitstheorie)
  - Als Computerprogramme und elektronische Schaltkreise steuern sie in Form von konkreten Funktionen Computer und andere Maschinen.



Formale Definition

- Eine Berechnungsvorschrift zur Lösung eines Problems heißt genau dann Algorithmus, wenn folgende Eigenschaften gelten:
  - Finitheit: das Verfahren muss in einem endlichen Text eindeutig beschreibbar sein
  - Ausführbarkeit: jeder Schritt des Verfahrens muss tatsächlich ausführbar sein
  - Dynamische Finitheit: das Verfahren darf zu jedem Zeitpunkt nur endlich viel Speicherplatz benötigen (Platzkomplexität)
  - Terminierung: das Verfahren darf nur endlich viele Schritte benötigen (Zeitkomplexität)

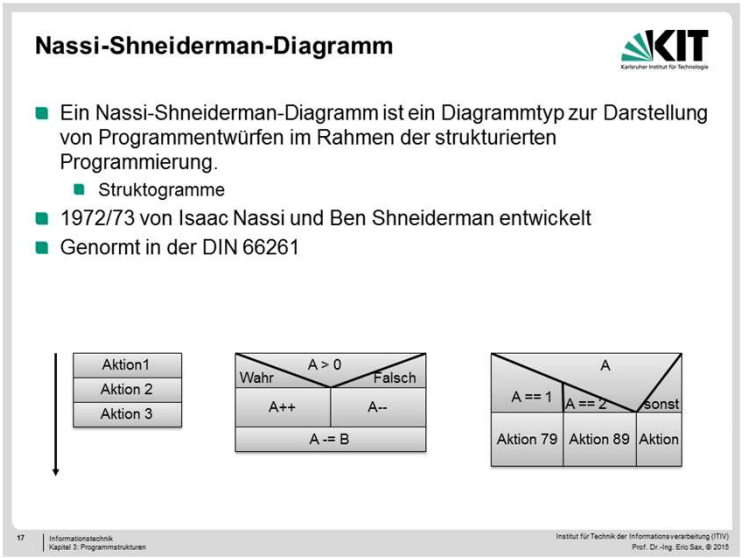
Praktische Eigenschaften

- Der Begriff Algorithmus ist in praktischen Bereichen oft auf die folgenden Eigenschaften eingeschränkt:
  - Determiniertheit: der Algorithmus muss bei den selben Voraussetzungen das gleiche Ergebnis liefern
  - Determinismus: die nächste anzuwendende Regel im Verfahren ist zu jedem Zeitpunkt eindeutig definiert

Beschreibung von Algorithmen



- Lösung eines Problems:
  - Finden einer algorithmischen Beschreibung des Lösungswegs
- Beschreibung mittels:
  - der natürlichen Sprache
  - als Computerprogramm
  - als Hardwareentwurf
  - mit Pseudo Code
  - grafisch, z.B: Nassi Shneiderman Diagrammen



Klassifikation von Algorithmen

- Einteilung der Algorithmen in Klassen
- Beschreibung der Algorithmen durch ihre Klassenmerkmale
- Klassifikationsmerkmale von Algorithmen:



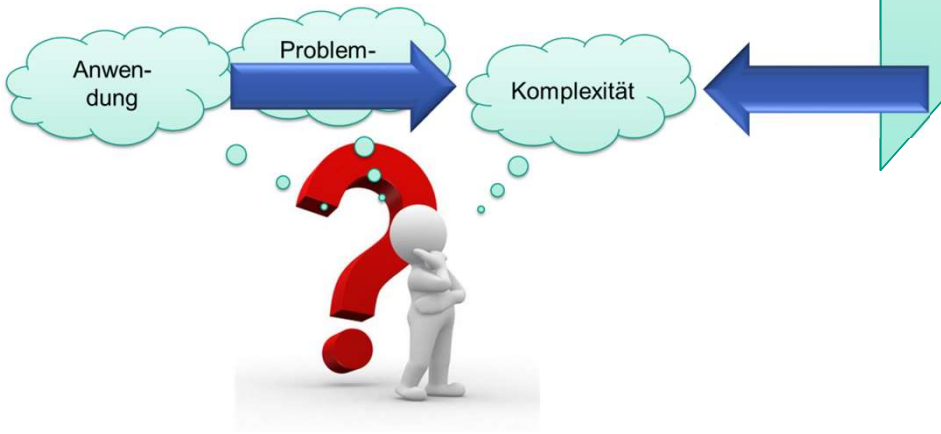
Klassen nach Komplexität

Platzkomplexität

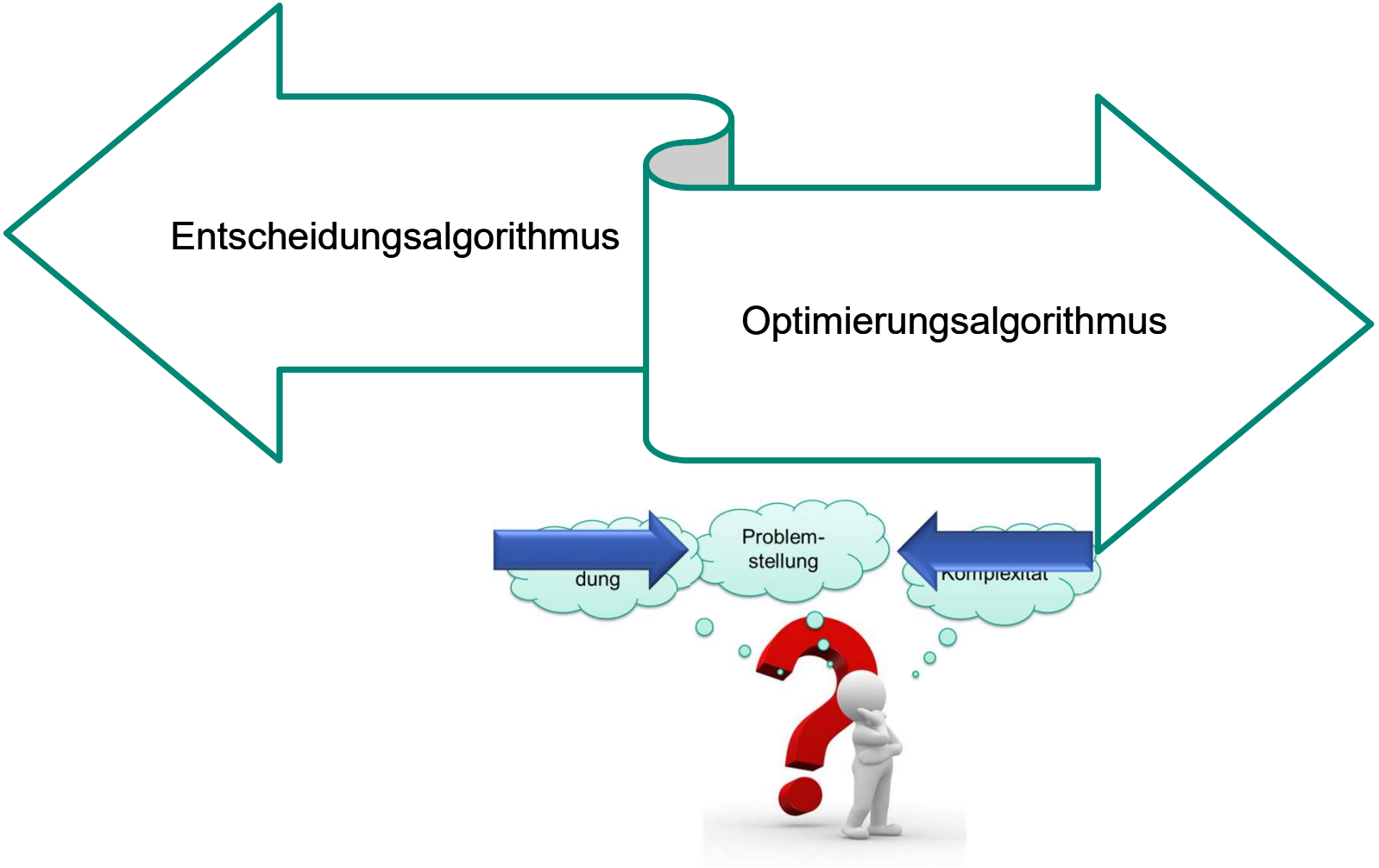
- Linear platzbeschränkt
- Logarithmisch platzbeschränkt
- Polynomial platzbeschränkt
- Exponentiell platzbeschränkt

Zeitkomplexität

- Linear zeitbeschränkt
- Logarithmisch zeitbeschränkt
- Polynomial zeitbeschränkt
- Exponentiell zeitbeschränkt

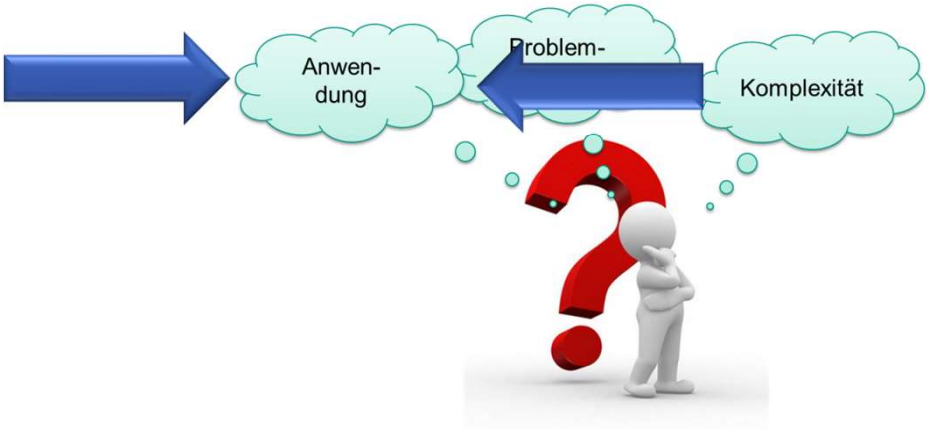


Klassen nach Problemstellung



Klassen nach Anwendung

- Suchalgorithmen
  - für Listen, Arrays: Lineare Suche, Binäre Suche, Interpolationssuche
  - für Graphen, Bäume: Breitensuche, Tiefensuche
- Sortieralgorithmen
  - Bubblesort, Heapsort, Insertionsort, Mergesort, Quicksort
- Graphentheorie
  - Kürzester Pfad, Breitensuche, Tiefensuche, Handlungsreisender
- Kryptographie: Verschlüsselungsalgorithmen
- Detektion: Bilderkennung
- ...



Weitere Merkmale von Algorithmen

Iterative Algorithmen

- Ein Algorithmus ist iterativ, wenn in seiner Realisierung keine Rekursionen vorkommen.



Rekursive Algorithmen

- Ein Algorithmus ist rekursiv, wenn dieser sich in seiner Realisierung nur selbst aufruft, jedoch keine Schleifen vorkommen.

Beispiel

Freie Universität Berlin

Rekursion vs. Iteration

Die Funktion **sum** berechnet für ein gegebenes **n>0** die Summe aller Zahlen von **1** bis **n**

Iterativ

```
def sum(n):
    sum = 0
    for i in range(n+1):
        sum += i
    return sum
```

Rekursiv

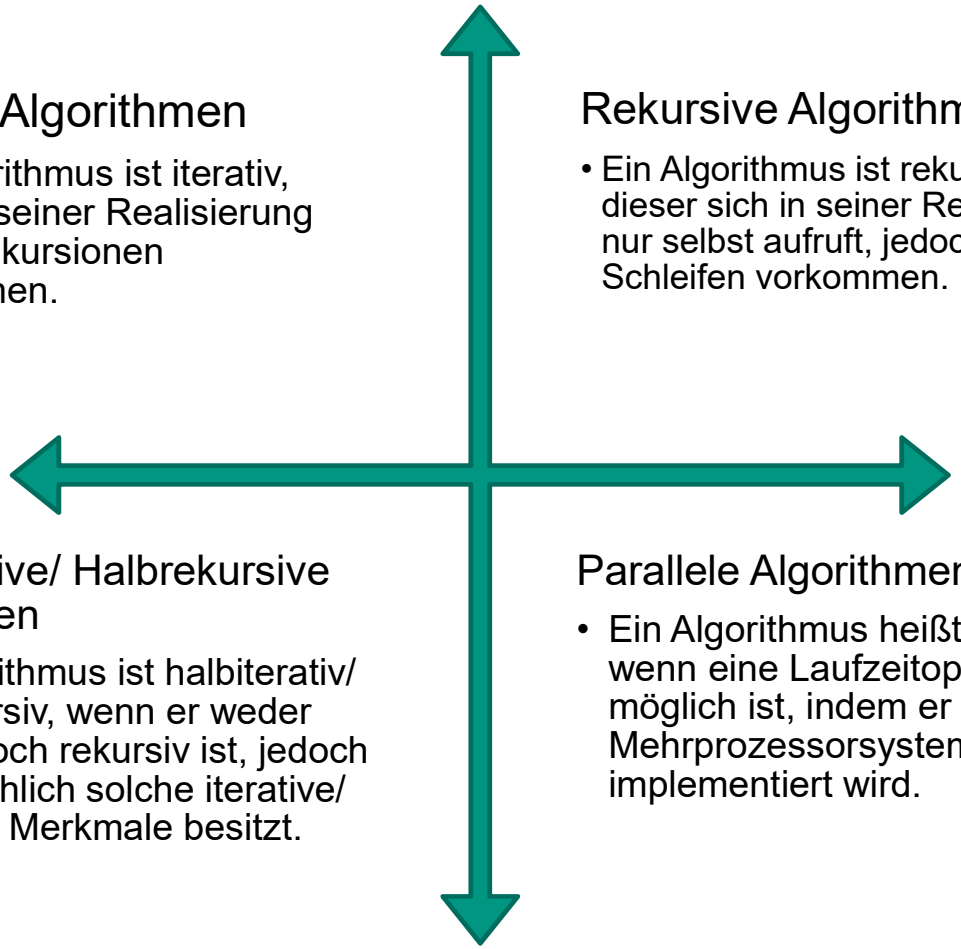
```
def sum_rec(n):
    if n==0:
        return 0
    else:
        return n+sum_rec(n-1)
```

direkt

Formel von Gauß

```
def sum(n):
    return n*(n+1)//2
```

Weitere Merkmale von Algorithmen



Iterative Algorithmen

- Ein Algorithmus ist iterativ, wenn in seiner Realisierung keine Rekursionen vorkommen.

Rekursive Algorithmen

- Ein Algorithmus ist rekursiv, wenn dieser sich in seiner Realisierung nur selbst aufruft, jedoch keine Schleifen vorkommen.

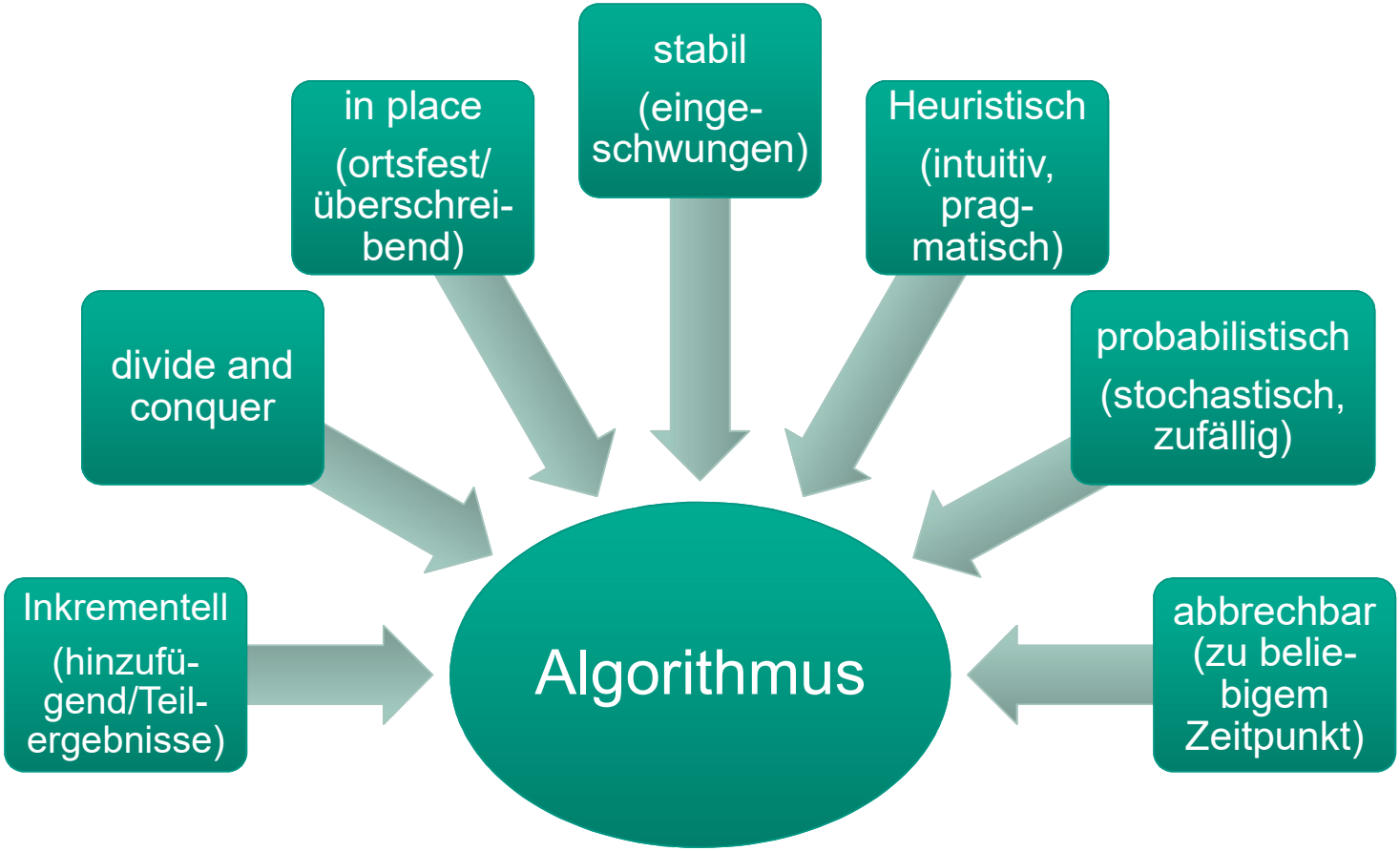
Halbiterative/ Halbrekursive Algorithmen

- Ein Algorithmus ist halbiterativ/ halbrekursiv, wenn er weder iterativ noch rekursiv ist, jedoch hauptsächlich solche iterative/ rekursive Merkmale besitzt.

Parallele Algorithmen

- Ein Algorithmus heißt parallel, wenn eine Laufzeitoptimierung möglich ist, indem er auf einem Mehrprozessorsystem implementiert wird.

Verschiedene Merkmale von Algorithmen (1)



Verschiedene Merkmale von Algorithmen (2a)

- **Inkrementell:** bedeutet im allgemeinen, dass der Algorithmus zunächst ein Teilergebnis bringt und dann während er läuft weitere Ergebnisse hinzufügt (inkrementell=hinzufügend).
  - Inkrementell kann auch bedeuten, dass ein Algorithmus neue Daten hinzufügend verarbeiten und das Ergebnis aktualisieren kann, ohne alles komplett neu zu berechnen.
- **divide and conquer**: Grundprinzip „teile und herrsche“
  - ein Problem wird solange in kleinere Teilprobleme zerlegt, bis man diese lösen (beherrschen) kann. Dabei wird ausgenutzt, dass bei vielen Problemen der Lösungsaufwand sinkt, wenn man das Problem in kleinere Teilprobleme zerlegt. Anschliessend wird aus den Teillösungen die Gesamtlösung (re)konstruiert.
- **in place:** Ein Algorithmus arbeitet in-place, wenn er außer dem für die Speicherung der zu bearbeitenden Daten benötigten Speicher nur eine konstante, also von der zu bearbeitenden Datenmenge unabhängige, Menge von Speicher benötigt.
  - Der Algorithmus überschreibt die Eingabedaten mit den Ausgabedaten. Hierbei ist auch von Ortsfestigkeit die Rede.
  - Das Gegenteil ist out-of-place.
- **stabil:** Ein Algorithmus sortiert stabil, wenn bei Gleichheit der Schlüssel 2er Elemente, die Reihenfolge der Elemente nicht verändert wird.
  - Ein (numerischer) Algorithmus heißt stabil wenn er für alle erlaubten und in der Größenordnung der Rechengenauigkeit gestörten Eingabedaten akzeptable Resultate produziert.

Verschiedene Merkmale von Algorithmen (2b)

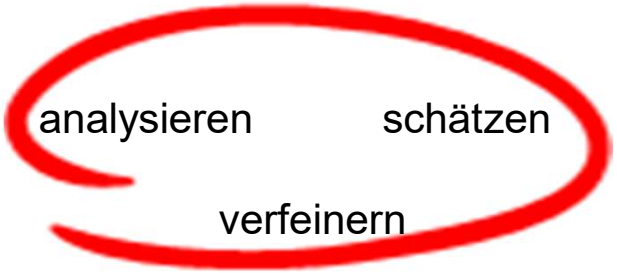
- **heuristisch:** heuristische Algorithmen versuchen mit geringem Rechenaufwand und kurzer Laufzeit zulässige Lösungen für ein bestimmtes Problem zu erhalten.
  - Klassische Algorithmen versuchen, einerseits die optimale Rechenzeit und andererseits die optimale Lösung zu garantieren.
    - Heuristische Verfahren verwerfen einen oder beide dieser Ansprüche, um bei komplexen Aufgaben einen Kompromiss zwischen dem Rechenaufwand und der Güte der gefundenen Lösung einzugehen.
  - Dazu wird versucht, mithilfe von Schätzungen, „Faustregeln“, intuitiv-intelligentem Raten oder unter zusätzlichen Hilfsannahmen eine gute Lösung zu erzeugen, ohne optimale Eigenschaften zu garantieren.
    - Heuristische Verfahren werden eingesetzt, wenn der erforderliche Rechenaufwand im Entscheidungsfindungsprozess zu umfangreich ist oder dieser den Rahmen des Möglichen sprengt.
    - Dabei wird die Anzahl der in Betracht zu ziehenden Möglichkeiten reduziert, indem man aussichtslos erscheinende Varianten von vornherein ausschließt.
  - Die Alternative zu heuristischen Verfahren ist die Brute-Force-Methode, bei der alle in Frage kommenden Möglichkeiten ausnahmslos durchgerechnet werden.
    - Die bekannteste und einfachste Heuristik ist die Lösung eines Problems mittels „Versuch und Irrtum“ (Englisch: by trial and error).

Verschiedene Merkmale von Algorithmen (2c)

- **probabilistisch:** Ein probabilistischer Algorithmus (auch stochastischer oder randomisierter Algorithmus) verwendet - im Gegensatz zu einem deterministischen Algorithmus - Zufallsbits um seinen Ablauf zu steuern.
  - Es wird nicht verlangt, dass ein randomisierter Algorithmus immer effizient eine richtige Lösung findet.
  - Randomisierte Algorithmen sind in vielen Fällen einfacher zu verstehen, einfacher zu implementieren und effizienter als deterministische Algorithmen für dasselbe Problem. Wir unterscheiden drei verschiedene Arten von probabilistischen Algorithmen.
  - Algorithmus 1. Art (Macao Algorithmus):
    - mindestens bei einem Schritt der Prozedur werden einige Zahlen zufällig ausgewählt (nicht definit). Sonst deterministisch. Diese Algorithmen liefern immer eine korrekte Antwort. Benutzt werden sie, wenn irgendein bekannter Algorithmus zur Lösung eines bestimmten Problems im mittleren Fall viel schneller als im schlechtesten Fall läuft.
  - Algorithmus 2. Art (Monte-Carlo Algorithmus):
    - gleich wie Algorithmus 1. Art (nicht definit). Zusätzlich: Ausgabe ist korrekt mit einer Wahrscheinlichkeit von $1 - \varrho$, wobei ϱ sehr klein ist (nicht endlich). Diese Algorithmen liefern immer eine Antwort, wobei die Antwort nicht unbedingt richtig ist. $\varrho \rightarrow 0$ falls $t \rightarrow \infty$. Das Problem bei solchen Algorithmen liegt darin, zu entscheiden, ob die Antwort korrekt ist.
  - Algorithmus 3. Art (Las-Vegas Algorithmus):
    - gleich wie Macao-Algorithmus (nicht definit). Eine Folge von zufälligen Wahlen kann unendlich sein (mit einer Wahrscheinlichkeit $\varrho \rightarrow 0$) (nicht endlich). Diese Algorithmen liefern nie eine unkorrekte Antwort, jedoch besteht die Möglichkeit dass keine Antwort gefunden wird.
- **abbrechbar:** Algorithmus der so entworfen ist, dass er zu jedem beliebigen Zeitpunkt abbrechbar ist und das bis dahin erreichte Ergebnis ein gültiges (meist nicht optimales) Ergebnis ist, das sofort ausgeben werden kann.

Algorithmen-Analyse z.B. Laufzeit

- Die Erforschung und Analyse von Algorithmen ist eine Hauptaufgabe der Informatik
  - Dies wird meist theoretisch (ohne konkrete Umsetzung in eine Programmiersprache) durchgeführt.
- Algorithmen werden zur Analyse in eine stark formalisierte Form gebracht und mit den Mitteln der formalen Semantik untersucht.
- Schrittweises Vorgehen
  - Charakterisierung der Eingabedaten
  - Bestimmung der abstrakten Operationen
  - Iterative Verbesserung:



- Problem für Laufzeitanalyse:
  - Abschätzung unter Verzicht auf Einzelheiten.
    - 90/10-Regel: 90% der Laufzeit bewirkt durch nur 10% des Programm-Codes
    - 1:50 Regel: 1% des Programm-Codes bewirkt 50% der Laufzeit

Teilgebiete der Algorithmenanalyse

- Komplexitätstheorie:
  - Verhalten von Algorithmen bezüglich Ressourcenbedarf wie Rechenzeit und Speicherbedarf
  - Ressourcenbedarf wird dabei in Abhängigkeit von der Länge der Eingabe ermittelt, z.B.:
    - Anzahl Elemente eines Feldes, einer Matrix
    - Länge einer Zeichenkette
    - Grad eines Polynoms
    - Anzahl der Knoten in einer dynamischen Datenstruktur (Liste, Baum)
  - → Die meisten Algorithmen besitzen einen Hauptparameter n, der die Anzahl der zu verarbeitenden Datenelemente angibt.
- Ziel: Bestimmung der Laufzeit unabhängig von
  - Hardware
  - Betriebssystem
  - Programmiersprache
  - Verwendete Bibliotheken
  - Compiler-Optimierungen

Relativer Zeitaufwand für „Integer“ Operationen
Feinanalyse

Elementare Aktionen und Operationen mit *integer*-Datenobjekten und ihr relativer Zeitaufwand bezogen auf die Zuweisung

Elementare Aktion/Operation		Relativer Zeitaufwand
Zuweisung (= Basis)	=	1.0
Addition oder Subtraktion	+/-	1.4
Multiplikation	*	2.3
Division	/	8.0
Vergleich (inkl. Sprung bei *if* oder *while*)	<>	1.5
Indizierung einer Matrix	[ ]	4.2

Werte wurden ermittelt mit einem PASCAL-Programm,
Borland PASCAL-Compiler Version 7.0,
auf PC unter Betriebssystem Microsoft XP
mit Prozessor Intel Pentium 4 mit 2GHz Taktfrequenz

Relative Laufzeitberechnung für Matrix[j,k] = 0

Algorithmus $A1$	Anzahl	Gewicht
i := 1	1	1.0
while i ≤ n1 do	$n1 + 1$	1.5
j := 1	$n1$	1.0
while j ≤ n2 do	$n1 \cdot (n2 + 1)$	1.5
⟹ matrix[i, j] := 0.0	$n1 \cdot n2$	4.2
j := j + 1	$n1 \cdot n2$	2.4
end -- while		
i := i + 1	$n1$	2.4
end -- while		
Summen		

=	1.0
+/-	1.4
*	2.3
/	8.0
<>	1.5
[]	4.2

Feinanalyse

Relative Laufzeitberechnung für Matrix[j,k] = 0

Algorithmus $A1$	Anzahl	Gewicht	Gesamt = Anzahl · Gewicht			
			$n1 \cdot n2$	$n1$	$n2$	konst.
i := 1	1	1.0				1.0
while i ≤ n1 do	$n1 + 1$	1.5		1.5		1.5
j := 1	$n1$	1.0		1.0		
while j ≤ n2 do	$n1 \cdot (n2 + 1)$	1.5	1.5	1.5		
matrix[i, j] := 0.0	$n1 \cdot n2$	4.2	4.2			
j := j + 1	$n1 \cdot n2$	2.4	2.4			
end -- while						
i := i + 1	$n1$	2.4		2.4		
end -- while						
Summen			8.1	6.4	0.0	2.5

=	1.0
+/-	1.4
*	2.3
/	8.0
<>	1.5
[]	4.2

Feinanalyse

Relative Laufzeit Algorithmus 1: $t_{A1}(n1, n2) = 8{,}1 \cdot n1 \cdot n2 + 6{,}4 \cdot n1 + 0 \cdot n2 + 2{,}5$

Relative Laufzeitberechnung für Matrix[j,k] = 0

Alternative Realisierung

Algorithmus $A2$	Anzahl	Gewicht	Gesamt = Anzahl · Gewicht			
			$n1 \cdot n2$	$n1$	$n2$	konst.
l := 1	1	1.0				1.0
c := 1	1	1.0				1.0
i := 1	1	1.0				1.0
noe := n1 * n2	1	3.3				3.3
while i ≤ noe do	$n1 \cdot n2 + 1$	1.5	1.5			1.5
matrix[l, c] := **0.0**	$n1 \cdot n2$	4,2	4.2			
if c = n2 then	$n1 \cdot n2$	1.5	1.5			
l := l + 1	$n1$	2.4		2.4		
c := 1	$n1$	1,0		1.0		
else						
c := c + 1	$n1 \cdot n2 - n1$	2.4	2.4	−2.4		
end -- if						
i := i + 1	$n1 \cdot n2$	2.4	2.4			
end -- while						
Summen			12.0	1.0	0.0	7.8

=	1.0
+/-	1.4
*	2.3
/	8.0
<>	1.5
[]	4.2

Feinanalyse

Relative Laufzeit Algorithmus 2: $t_{A2}(n1, n2) = 12{,}0 \cdot n1 \cdot n2 + 1{,}0 \cdot n1 + 0 \cdot n2 + 7{,}8$

Vergleich Algorithmus 1 und 2

- Laufvariable = Indexvariable
- Geschachtelte While-Schleife

Algorithmus $A1$

```
i := 1
while i ≤ n1 do
   j := 1
   while j ≤ n2 do
      matrix[i, j] := 0.0
      j := j + 1
   end -- while
   i := i + 1
end -- while
Summen
```

- Laufvariable ≠ Indexvariable
- 1 While-Schleife und 1 if/else-Abfrage

```
l := 1
c := 1
i := 1
noe := n1 * n2
while i ≤ noe do
   matrix[l, c] := 0.0
   if c = n2 then
      l := l + 1
      c := 1
   else
      c := c + 1
   end -- if
   i := i + 1
end -- while
Summen
```

Vergleich der relativen Laufzeiten

- Algorithmus 1 $t_{A1} = 8{,}1 \cdot n1 \cdot n2 + 6{,}4 \cdot n1 + 2{,}5$
- Algorithmus 2 $t_{A2} = 12{,}0 \cdot n1 \cdot n2 + 1{,}0 \cdot n1 + 7{,}8$
- Zum einfacheren Vergleich des Laufzeitverhaltens
  - Einführung neue Problemgröße:
  - grobe Maßzahl für die Anzahl der Durchläufe

  $\Rightarrow n = \sqrt{n1 \cdot n2}$
- … und für großes n: $t'_{A1} = 8{,}1 \cdot n^2$ $t'_{A2} = 12{,}0 \cdot n^2$
- Analyseergebnis:
  - Algorithmus 1 und 2 zeigen ein quadratisches Wachstum mit n.
  - Algorithmus 2 braucht zum Lösen der Aufgabe für große n (> 2) etwa 50% mehr Rechenzeit
- Überprüfung mit Zeitmessung auf genanntem Rechner
  - für n1 = n2 = n = 100 und 10.000 Wiederholungen
  - Bei 10.000 Wdh. → Laufzeit 1,37 Sekunden bzw. 2,09 Sekunden

Laufzeitkomplexität → Grobanalyse

- Feinanalyse (s. zuvor) wird i.A. nur bei sicherheitsrelevanten Systemen mit harten Realzeitanforderungen durchgeführt
- In den meisten praktischen Anwendungen nur Grobanalyse
  - Identifiziere und analysiere die Teile des Algorithmus, die das Laufzeitverhalten signifikant beeinflussen
  - Suche innerste (am häufigsten zu durchlaufende) Schleife
  - Analysiere prinzipielles Laufzeitverhalten in Abhängigkeit der Problemgröße n
  - Analysiere Abhängigkeit von Inhalt und Ausprägung der verarbeitenden Datenobjekte:
    - Günstigster Fall (best case) → minimale Zahl an Arbeitsschritten
    - Ungünstigster Fall (worst case) → maximale Zahl an Arbeitsschritten
    - Durchschnittlicher Fall (average case)
      - typische, über viele Anwendungen hinweg betrachtete und gemittelte Zahl an Arbeitsschritten
      - häufig nur auf Basis von Annahmen und mit Hilfe Wahrscheinlichkeitsrechnung

Effizienz versus Effektivität

- Effektivität: Wirksamkeit des Algorithmus zur möglichst guten Lösung der Aufgabenstellung unabhängig vom Aufwand
- Effizienz: Wirtschaftlichkeit des Algorithmus.

  Verhältnis von Aufwand zum Grad der Zielerreichung

  → Effizienz setzt Effektivität voraus

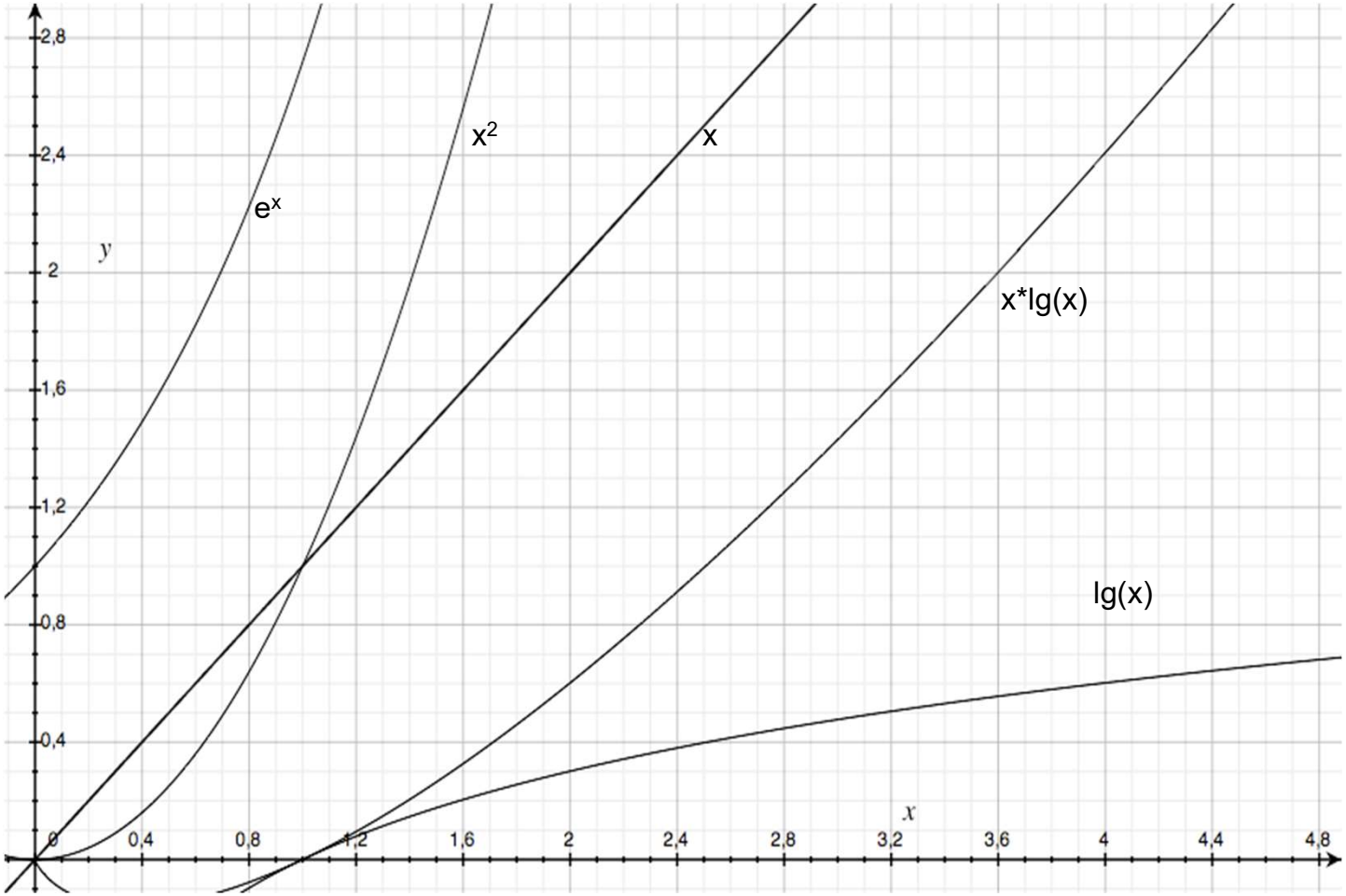
O-Notation

- Ursprung
  - Die *O*-Notation (gesprochen: „groß oh“, als Symbol für „*Ordnung von*“) wird grundsätzlich auf die Veröffentlichung *Analytische Zahlentheorie* aus dem Jahr 1892 von Peter Bachmann (1837 - 1920) zurückgeführt.
  - Einige Zeit später hat der Zahlentheoretiker Edmund Landau (1877 -1938) davon Gebrauch gemacht und neben ‚*O*’ auch ‚*o*’ betrachtet.
  - Aufgrund dessen werden Notationen auch als Gebrauch Landau’scher Symbole bezeichnet.
  - Remark: *Auf Landau geht auch das Symbol für die Menge der ganzen Zahlen zurück.*
- Zweck
  - Die *O*-Notation wird genutzt, um Abschätzungen über Aufwände von Algorithmen vorzunehmen.
  - obere Schranken erkennen, da Probleme vorwiegend durch die hohen Aufwände für Speicher und Laufzeit entstehen.
  - Ermittlung unterer Schranken und arithmetischer Mittel. Untere Schranken können von Bedeutung sein, um zu zeigen, dass es keinen Algorithmus gibt, der schneller läuft als $g(n)$.

Laufzeitkomplexitätsklassen

Asymptotische Laufzeit-komplexität	Bezeichnung	Erläuterung am Beispiel der Verdoppelung der Problemgröße und typische Algorithmen mit dieser Ordnung
$O(1)$	konstant	Laufzeit ist unabhängig von der Problemgröße, optimaler Fall, der praktisch nicht auftritt
$O(\log n)$	logarithmisch	Verdoppelung der Problemgröße bewirkt Anstieg der Laufzeit um $\log 2$, also um eine Konstante (um 1 für *logarithmus dualis*), sehr günstig und daher erstrebenswert, z.B. *binäre Suche* (siehe Kapitel 6)
$O(n)$	linear	Verdoppelung der Problemgröße bewirkt Verdoppelung der Laufzeit, immer noch zufrieden stellend, z.B. *sequenzielle Suche* (siehe Kapitel 6)
$O(n \log n)$	–	Fast so gut wie linear, v ältnis zu n klein ist, z.B. gute Sortierverfahren wie *Quicksort* (siehe Kapitel 7)
$O(n^2)$	quadratisch	Verdoppelung der Problemgröße bewirkt Vervierfachung der Laufzeit, ungünstig, z.B. schlechte Sortierverfahren wie *Bubblesort* (siehe K
$O(n \log n)$	–	Fast so gut wie linear, weil $\log n$ im Verhältnis zu n klein ist, z.B. gute Sortierverfahren wie *Quicksort*
$O(n^2)$	quadratisch	Verdoppelung der Problemgröße bewirkt Vervierfachung der Laufzeit, ungünstig, z.B. schlechte Sortierverfahren wie *Bubblesort*
$O(n^3)$	kubisch	Verdoppelung der Problemgröße bewirkt Verachtfachung der Laufzeit, sehr unbefriedigend, z.B. einfache *Matrizenmultiplikation*
$O(k^n)$	exponentiell	Verdoppelung der Problemgröße bedeutet Quadrierung (weil $k^{2n} = (k^n)^2$) der Laufzeit, katastrophal, z.B. *Backtracking*- oder *Exhaustionsalgorithmen*

Beispiel: Komplexitätsverhalten (kleine n)



Zwischenübung: Typische Wachstumsgesetze

- Annahme: Verwendet wird ein Rechner mit 10^9 Instruktionen pro Sekunde (1.000 MIPS) und ein Algorithmus mit 1.000 Instruktionen.
- Wieviele Algorithmen mit Wachstum g(n) (s. Zeilen) können in der Zeit t (s. Spalten) umgesetzt werden?

g(n)	1 Sekunde	1 Minute	1 Stunde	1 Tag	1 Monat	1 Jahr	1 Jahrhundert
$\log_2 n$							
$\sqrt{n}$		$= 10^9 \div 10^3 = 10^6 = \sqrt{n} \rightarrow n = 10^{12}$					
n		$= 10^9 \div 10^3 = 10^6 = n$					
$n \log_2 n$							
n^2		$= 10^9 \div 10^3 = 10^6 = n^2 \rightarrow n = 10^3$					
n^3							
2^n							
$n!$							

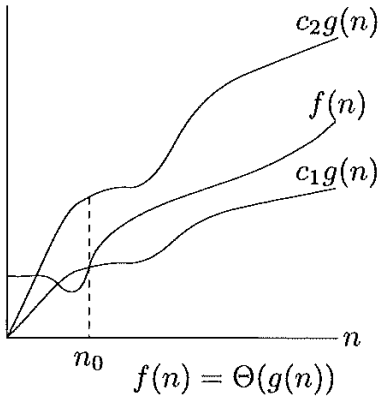
Zwischenübung: Typische Wachstumsgesetze Lsg.

- Annahme: Verwendet wird ein Rechner mit 10^9 Instruktionen pro Sekunde (1.000 MIPS) und ein Algorithmus mit 1.000 Instruktionen.
- Wieviele Algorithmen mit Wachstum g(n) (s. Zeilen) können in der Zeit t (s. Spalten) umgesetzt werden?

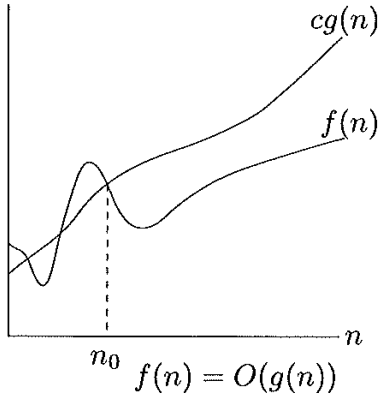
g(n)	1 Sekunde	1 Minute	1 Stunde	1 Tag	1 Monat	1 Jahr	1 Jahrhundert
$\log_2 n$	2^{10^6}	$\approx\infty$	$\approx\infty$	$\approx\infty$	$\approx\infty$	$\approx\infty$	$\approx\infty$
$\sqrt{n}$	1E12	$= 10^9 \div 10^3 = 10^6 = \sqrt{n} \rightarrow n = 10^{12}$					9,9E30
n	1E6	$= 10^9 \div 10^3 = 10^6 = n$			2,7E12	31,5E12	3,15E15
$n \log_2 n$	62746	2,8E6	1,3E8	2,8E9	7,5E10	8,0E11	6,9E13
n^2	1000	$= 10^9 \div 10^3 = 10^6 = n^2 \rightarrow n = 10^3$				6	5,6E7
n^3	100	391	1532	4420	13924	31581	146589
2^n	19	$= 2^{20} = 1.048.576$		36	41	44	51
$n!$	9	11	12	13	15	16	17

Notationen zur Definition der Ordnung bzw. von Schranken für das Wachstum von Funktionen

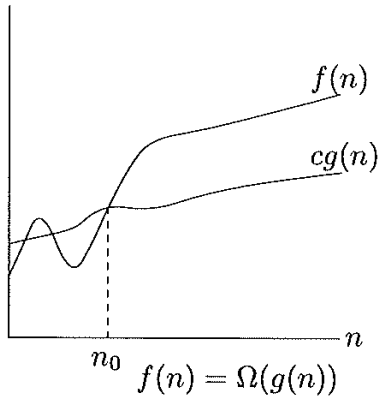
Notation	Definition	Interpretation
O-Notation	$f(n) = O(g(n))$ bedeutet $\lim_{n\to\infty} \frac{f(n)}{g(n)} \leq c$	f ist *höchstens* von der Ordnung g g definiert obere Schranke für f
Ω-Notation	$f(n) = \Omega(g(n))$ bedeutet $\lim_{n\to\infty} \frac{g(n)}{f(n)} \leq c$	f ist *mindestens* von der Ordnung g g definiert untere Schranke für f
Θ-Notation	$f(n) = \Theta(g(n))$ bedeutet $c_1 \leq \lim_{n\to\infty} \frac{f(n)}{g(n)} \leq c_2$	f ist *genau* von der Ordnung g g definiert eine Bandbreite für f



$f(n) = \Theta(g(n))$



$f(n) = O(g(n))$



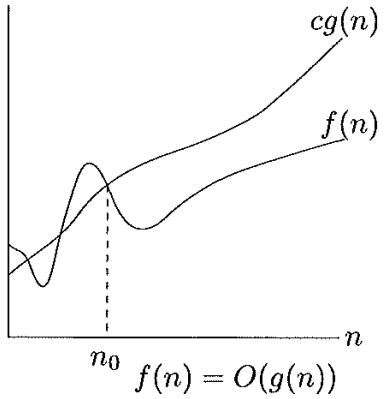
$f(n) = \Omega(g(n))$

O-Notation: asymptotische Laufzeitkomplexität

- Obere asymptotische Schranke
- Bei einer gegebenen Funktion g(n) bezeichnen wir mit O(g(n)) (ausgesprochen „groß O von g von n“) die Menge der Funktionen

  O(g(n)) = {f(n), so dass für positive Konstanten c und n_0 gilt

  $0 < f(n) < cg(n)$ für alle $n > n_0$ }



gesucht :
- möglichst “kleine” und “einfache” Funktionen g(n)
- mit möglichst geringen konstanten Faktoren c
- als asymptotische obere Schranke für f(n)

Beispiel asymptotische Laufzeitkomplexität

- Ermittlung der asymptotischen Laufzeitkomplexität des Algorithmus A1

$$t_{A1}(n1, n2) = 8{,}1 \cdot n1 \cdot n2 + 6{,}4 \cdot n1 + 2{,}5$$

- o.B.d.A. mit Problemgröße $n1 = n2 = n$

$$t_{A1}(n) = 8{,}1 \cdot n^2 + 6{,}4 \cdot n + 2{,}5 = n^2 \cdot (8{,}1 + 6{,}4 / n + 2{,}5 / n^2)$$

O(g(n)) = {f(n), so dass für positive Konstanten c und n_0 gilt

$0 < f(n) < cg(n)$ für alle $n > n_0$ }

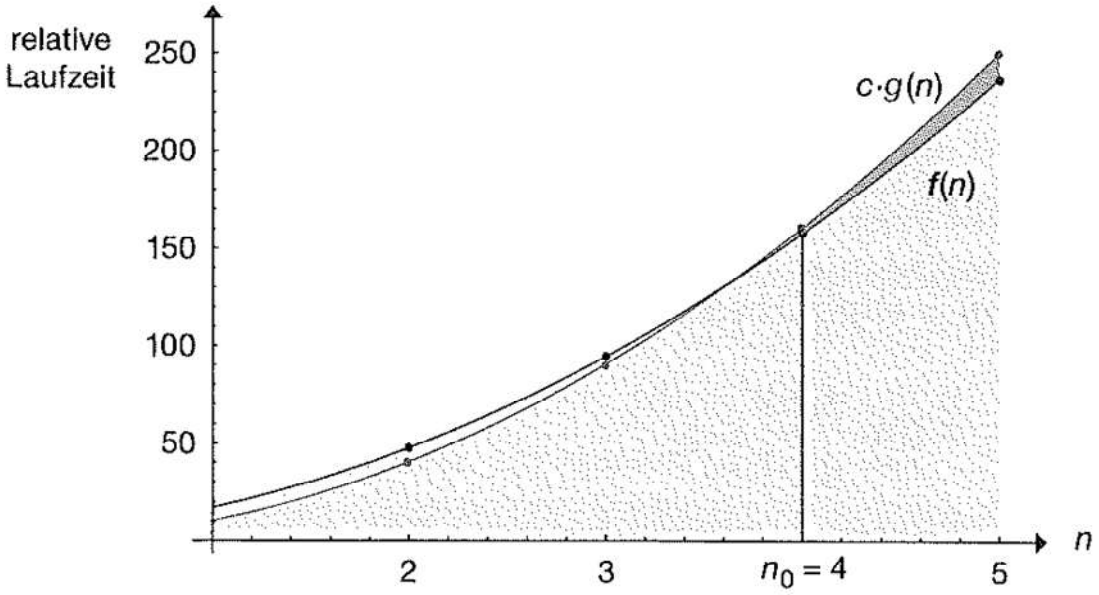
- Daraus ist ersichtlich, dass der Wert der Konstanten c gemäß Definition der O-Notation auf jeden Fall größer als 8,1 sein muss.

Beispiel asymptotische Laufzeitkomplexität

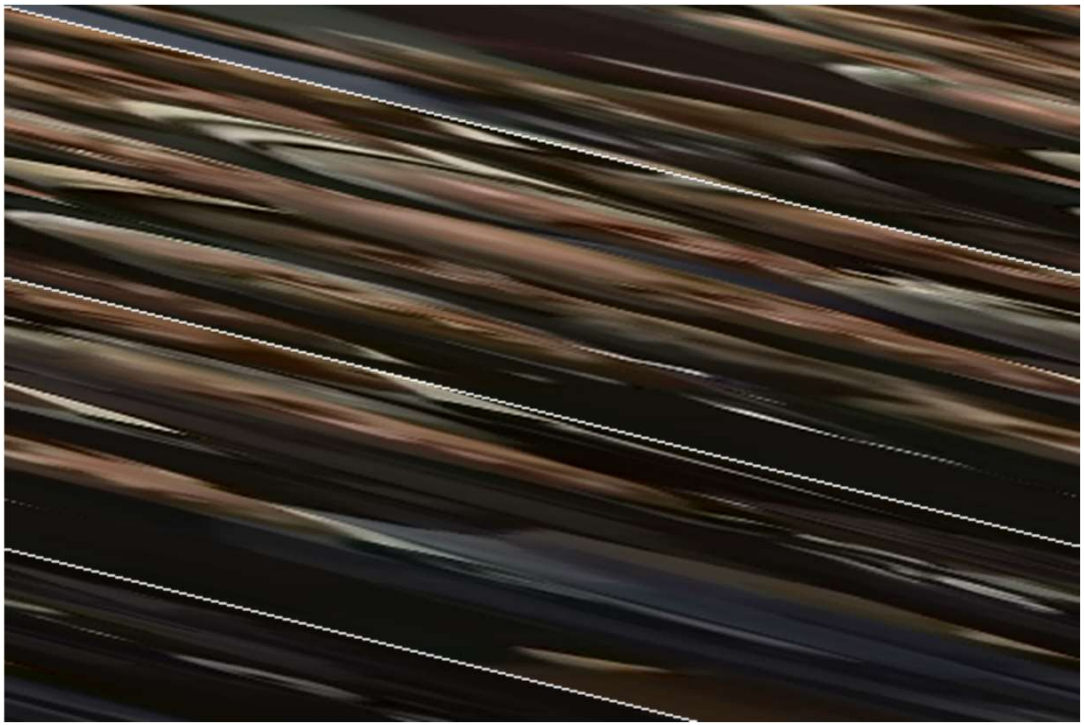
$$f(n) = n^2 \cdot (8{,}1 + 6{,}4 / n + 2{,}5 / n^2)$$
$$g(n) = n^2$$

- Für diese Funktion f(n) gilt also schon für

$$n_0 = 4, \text{ dass } f(n) \leq c \cdot g(n) \text{ mit } c = 10$$
$$und \;\; g(n) = n^2 \text{ ist.}$$



„Laufzeit“: Chariots of Fire



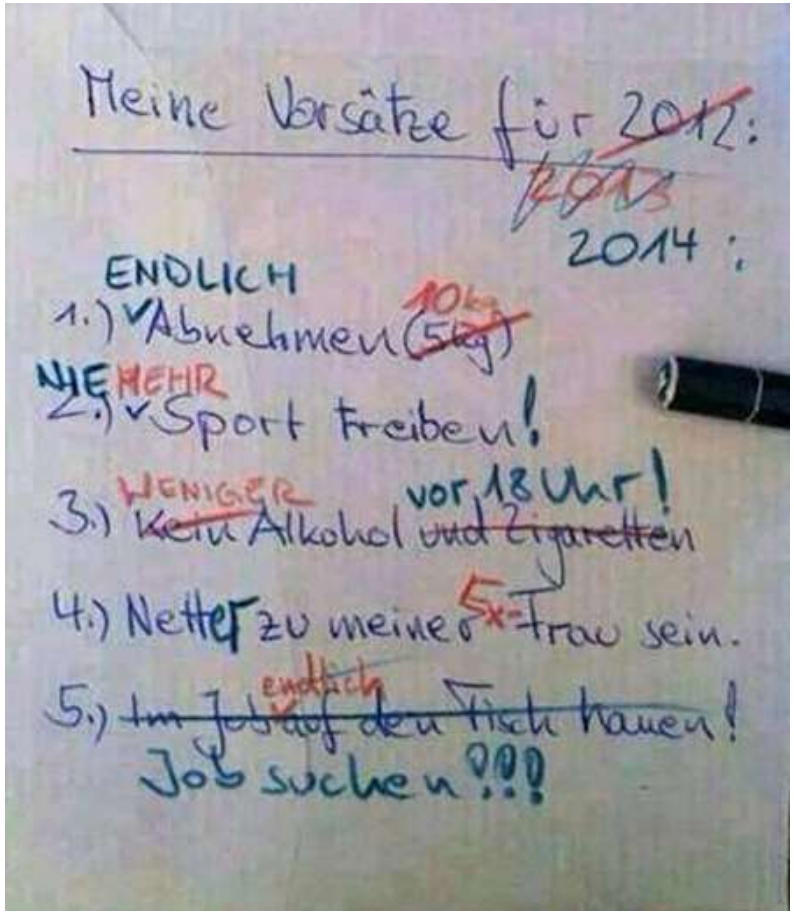
„Laufzeit“: Mr. Bean





Fragen?


Backup
Shift
Insert


Meine Vorsätze für 2012:
2013
2014:
ENDLICH
1.) Abnehmen (5kg) 10kg
2.) Sport treiben!
MEHR
NIE
3.) Kein Alkohol und Zigaretten
WENIGER
vor 18 Uhr!
4.) Netter zu meiner Frau sein.
5x
5.) Im Job auf den Tisch hauen!
endlich
Job suchen?!!

Auswahl eines Algorithmus

- Entwicklung von Alternativen (z.B. drei):
  - Analyse der Algorithmen (mathematisch, empirisch)
  - Begründete Auswahl
  - Implementierung
  - Test

  Regel: nur 1 einfachster Algorithmus o.k., wenn Algorithmus nur wenig genutzt oder wenn erste schnelle Lösung im Gesamtsystem gefordert

  Fehler 1: Missachtung der Leistungsmerkmale, schnellerer Algorithmus ist oft nicht komplizierter

  Fehler 2: Leistungsmerkmale erhalten zu viel Aufmerksamkeit

- Zur iterativen Verbesserung der Lösung (sukzessive Verfeinerung):
  - Finde innere Schleife
  - Prüfe jeden Befehl der inneren Schleife (z.B. keine Prozeduraufrufe)
  - Ersetze rekursiv durch iterativ
  - Letztlich implementiere in Assembler

Effizienz von Algorithmen

Effizienz von Algorithmen hinsichtlich Rechenzeit kann erheblicher sein als durch Hardware und Software bedingte Unterschiede

Beispiel Rechenzeit für Sortieralgorithmus (für n Einträge):

Insertion Sort: $c_1 * n^2$ (c_1 Konstante unabhängig von n)

Merge Sort: $c_2 * n * \log_2 n$ (c_2 andere Konstante unabhängig von n)

Im Allgemeinen gilt $c_1 < c_2$.

Bei kleinem n ist Insertion Sort schneller als Merge Sort, aber ganz gleich um wieviel kleiner c_1 gegenüber c_2 ist, es wird einen Übergangspunkt geben für n, ab dem Sortieren durch Mischen schneller ist.

Bespiel: Rechner A 10^9 Instruktionen pro Sekunde, c_1 sei 2
Rechner B 10^7 Instruktionen pro Sekunden, c_2 sei 50
für n = 10^6 Einträge, Rechenzeit A = 2000 Sekunden, Rechenzeit B ~ 100 Sekunden

Typische Wachstumsgesetze

Vergleich von Rechenzeiten:

- Bestimmen Sie die Rechenzeiten für ein Problem mit $n=10^3$ für die in der Tabelle gelisteten Wachstums-gesetze
- Verwendet wird ein Rechner mit 10^9 Instruktionen pro Sekunde (1.000 MIPS) und ein Algorithmus mit 1.000 Instruktionen.

	Rechenzeit [sec]
$\log_2 n$	
$\sqrt{n}$	
n	
$n \log_2 n$	
n^2	
n^3	
2^n	
$n!$	

Typische Wachstumsgesetze Lsg.

Vergleich von Rechenzeiten:

- Bestimmen Sie die Rechenzeiten für ein Problem mit n=10^3 für die in der Tabelle gelisteten Wachstums-gesetze
- Verwendet wird ein Rechner mit 10^9 Instruktionen pro Sekunde (1.000 MIPS) und ein Algorithmus mit 1.000 Instruktionen.

	Rechenzeit [sec]
$\log_2 n$	9,97 µs
$\sqrt{n}$	31,6 µs
n	1 ms
$n \log_2 n$	9,97 ms
n^2	1 s
n^3	1000 s
2^n	1,07 * 10^{295} s
$n!$	4,02 * 10^{2561} s

Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

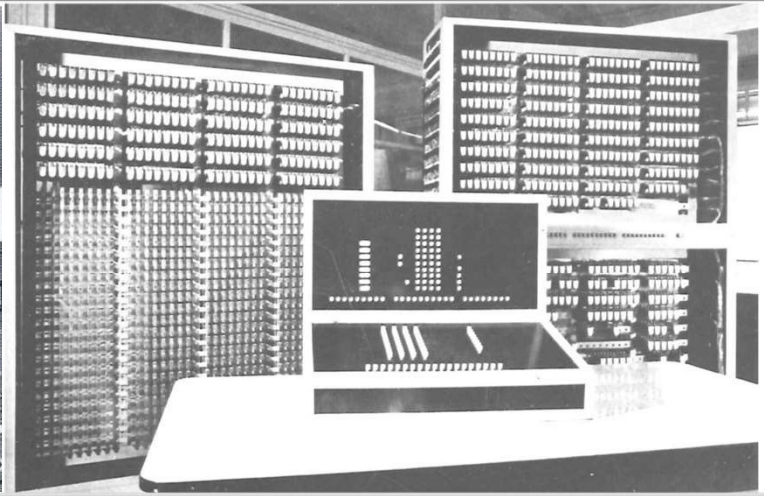
Institutsleitung
Prof. Dr.-Ing. J. Becker
Prof. Dr.-Ing. E. Sax
Prof. Dr. rer. nat. W. Stork

3. Sortieralgorithmen

Informationstechnik II

Prof. Dr.-Ing. Eric Sax







Inhalt IT2

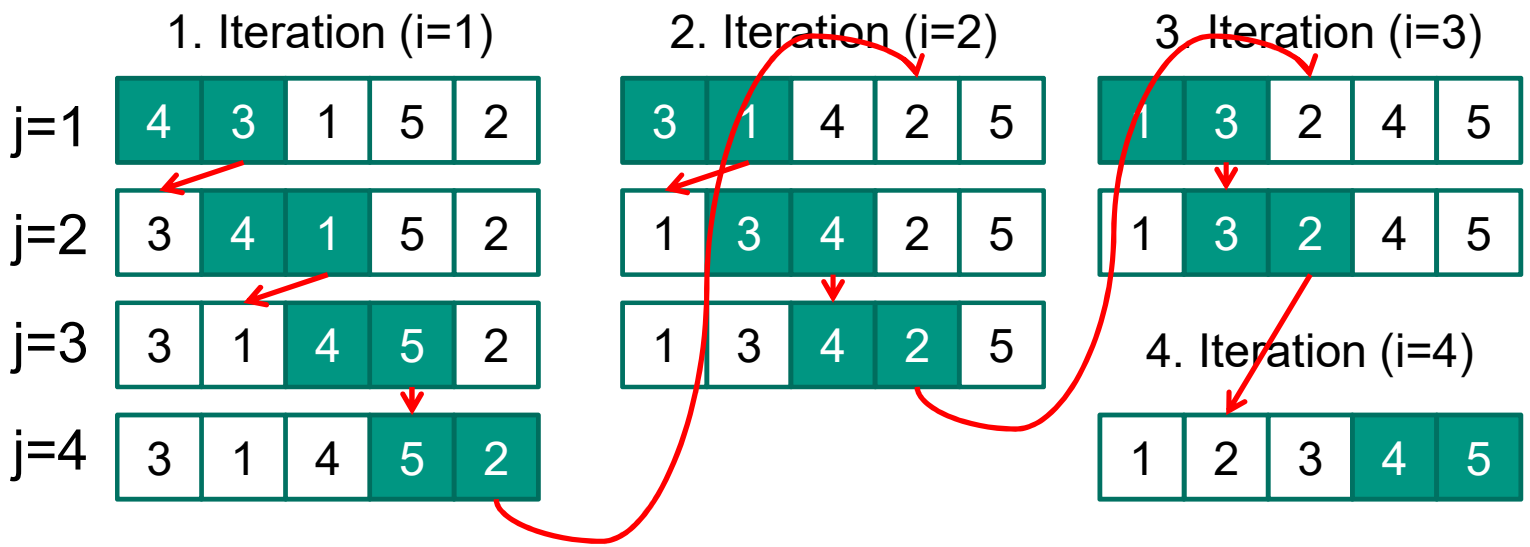
1. **Einführung**
   - Aufbau Modul IT 2 / Verbindung zu IT1 und anderen Lehrveranstaltungen
   - Anwendungsbeispiele und Motivation
   - Definitionen und Begriffe

2. **Algorithmische Grundlagen/Eigenschaften**
   - Definition Algorithmus et al.
   - Klassifikation von Algorithmen (Komplexität, Problemstellung, Anwendung)
   - Merkmale
   - Algorithmen-Analyse (Laufzeit, Effizienz, Effektivität)

Sortieralgorithmen
- Bubble Sort
- Insertion Sort
- Merge Sort
- Quick Sort



Bubble Sort - Verfahren



Bubblesort (A)

```
1    for i = 1 to länge[A]-1
2    do for j = 1 to länge[A]-i
3       do if A[j] > A[j+1]
4          then vertausche A[j] mit A[j+1]
```

Bubble Sort - Laufzeitanalyse

länge[A]

Prüfabfrage in for-Schleife

Zeile	Kostenkoeffizient	Zeit
1	c1	n
2	c2	
3	c3	
4	c4	

n -1 +1

$$\sum_{i=1}^{n-1}(n-i+1)=\sum_{i=1}^{n-1}(n-i)+\sum_{j=1}^{n-1}1$$

$$\sum_{i=1}^{n-1}(n-i)$$

$$\sum_{i=1}^{n-1}(n-i)$$

Bubblesort (A)

```
1    for i = 1 to länge[A]-1
2    do for j = 1 to länge[A]-i
3       do if A[j] > A[j+1]
4          then vertausche A[j] mit A[j+1]
```

n=Wortlänge von A

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$



Bubble Sort - Laufzeitanalyse

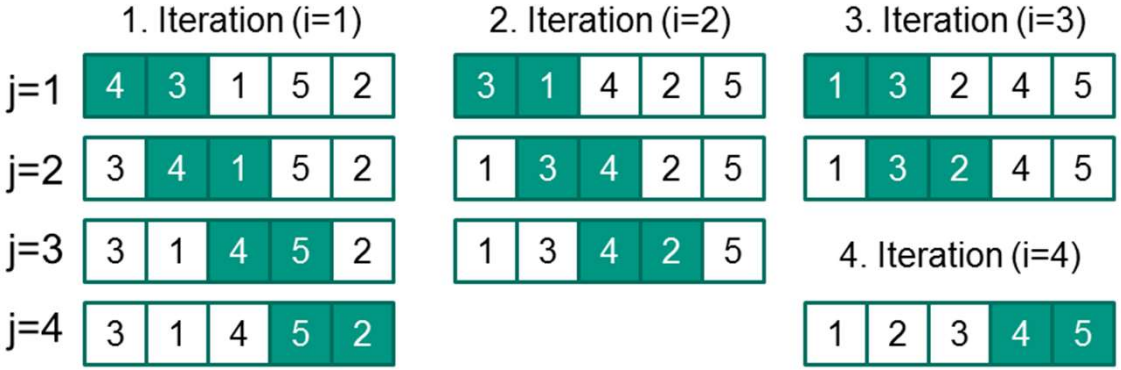
Zeile	Kostenkoeffizient	Zeit
1	c1	n
2	c2	
3	c3	
4	c4	

n -1 +1

$$\sum_{i=1}^{n-1}(n-i+1)=\sum_{i=1}^{n-1}(n-i)+\sum_{i=1}^{n-1}1=$$

$$\sum_{i=1}^{n-1}(n-i)=\sum_{i=1}^{n-1}n-\sum_{i=1}^{n-1}i=$$

$$\sum_{i=1}^{n-1}(n-i)=$$



n=Wortlänge von A

$$\sum_{k=1}^{n}k=\frac{n(n+1)}{2}$$

Bubble Sort - Laufzeitanalyse

Somit gilt

$$f(n) = O(n^2)$$

oder anders ausgedrückt

$$f(n) \text{ ist von der Ordnung } (n^2)$$

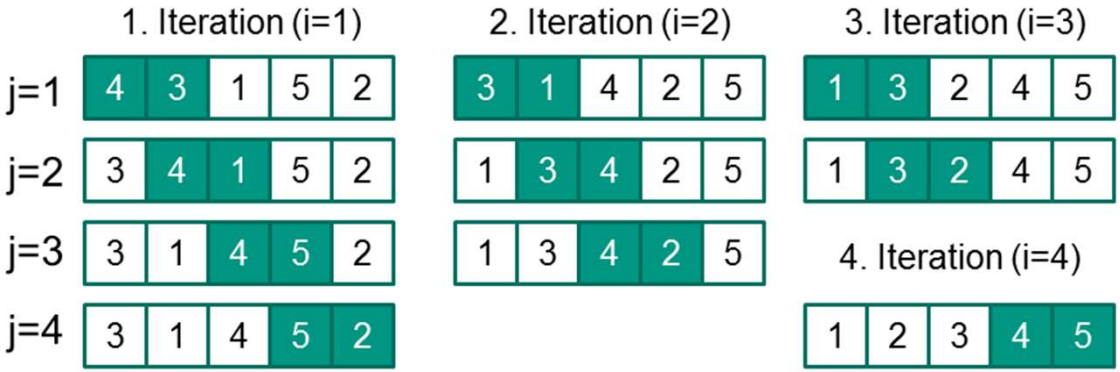
	Zeile	Kostenkoeffizient	Zeit
5	1	c1	n
14	2	c2	((n²+n)/2)-1
10	3	c3	(n²-n)/2
10	4	c4	(n²-n)/2

n -1 +1

$$\sum_{i=1}^{n-1}(n-i+1) = \sum_{i=1}^{n-1}(n-i) + \sum_{i=1}^{n-1}1 = n\cdot\frac{n-1}{2} + \frac{2n-2}{2}$$

$$\sum_{i=1}^{n-1}(n-i) = \sum_{i=1}^{n-1}n - \sum_{i=1}^{n-1}i = (n-1)n - \frac{(n-1)(n)}{2} = n\cdot\frac{n-1}{2}$$

$$\sum_{i=1}^{n-1}(n-i) = n\cdot\frac{n-1}{2}$$

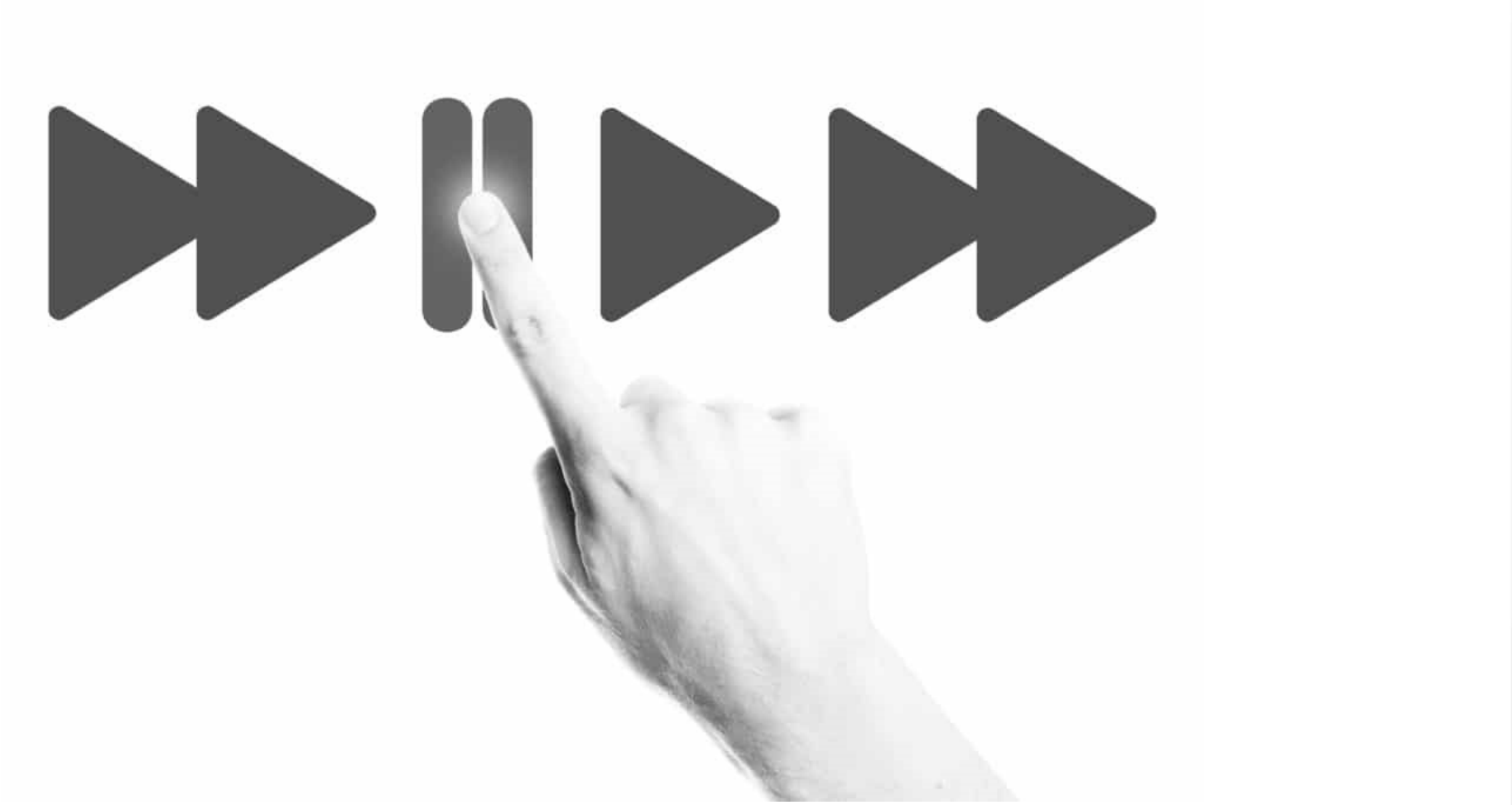


n=Wortlänge von A

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

- Bubble Sort



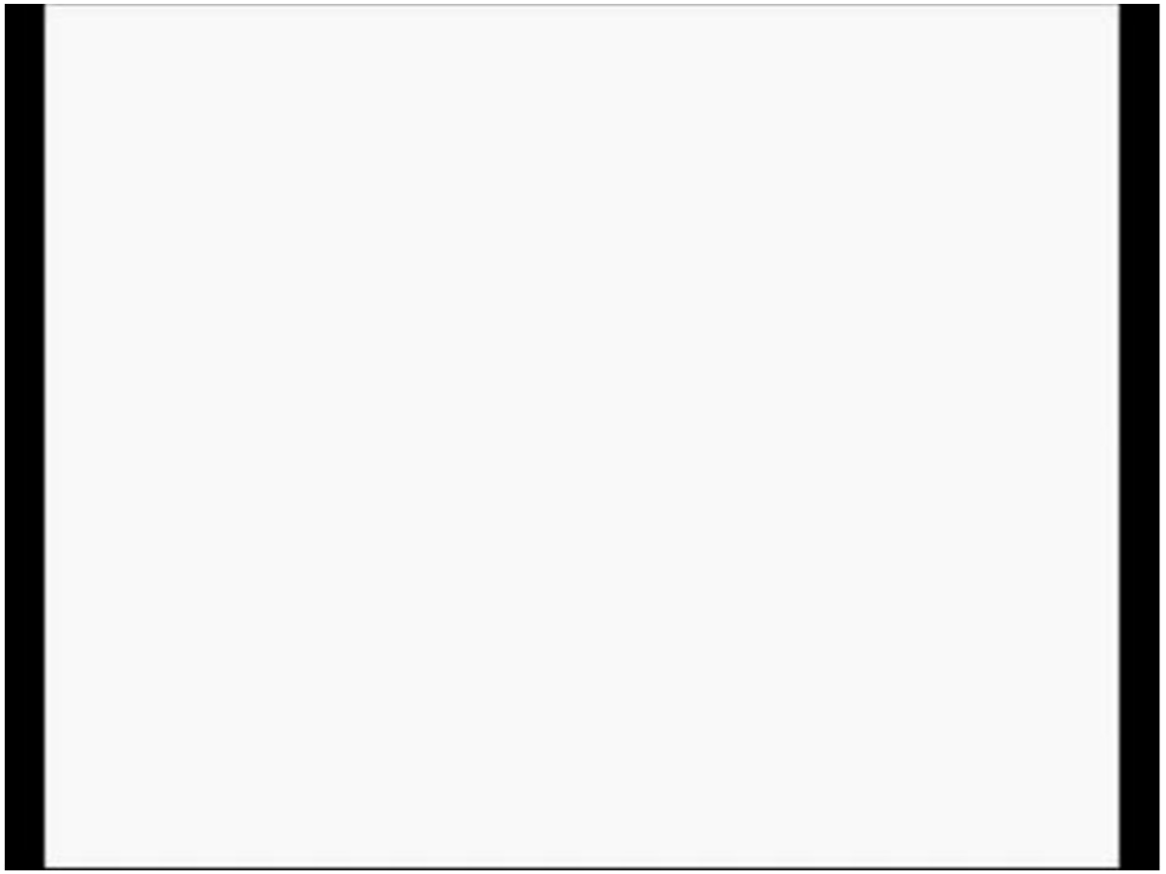


Insertion Sort

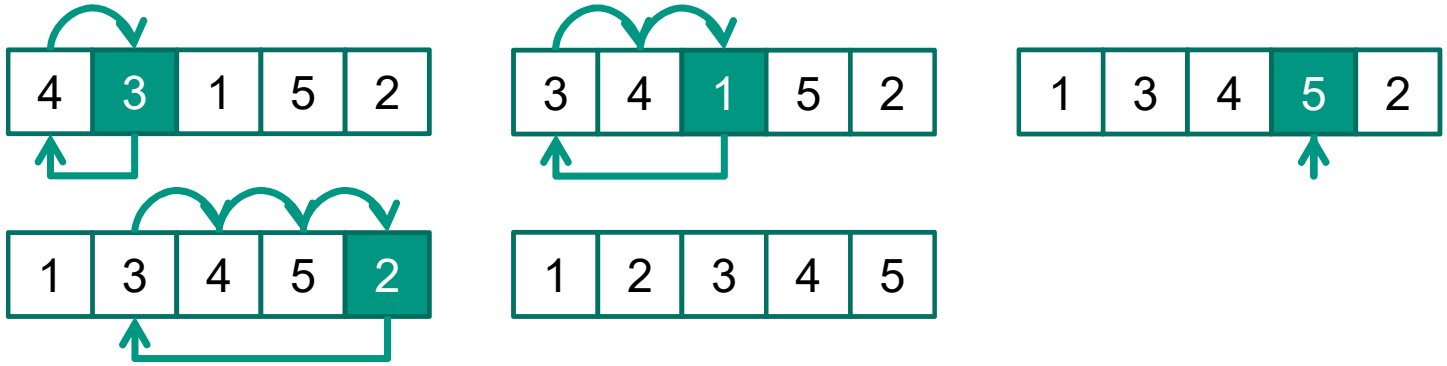
- Sortieren durch Einfügen
  - (engl. Insertion Sort)

- Eigenschaften:
  - einfaches Sortierverfahren
    - weniger effizient bei großen Datenmengen als andere anspruchsvolle Sortierverfahren (Quick Sort, Merge Sort etc.)
  - Vorteil:
    - einfache Implementierung
    - effizient bei kleinen Eingabemengen
    - effizient bei schon vorsortierten Eingabemengen
  - stabil
    - d.h. die Reihenfolge von schon sortierten Elementen ändert sich nicht
  - minimal im Speicherverbrauch, da der Algorithmus ortsfest arbeitet

Insertion Sort

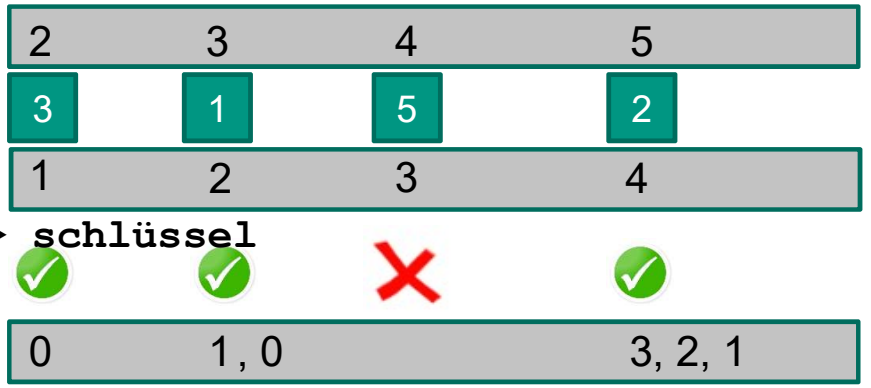


Insertion Sort - Verfahren




```
INSERTIONSORT(A)
1 for j = 2 to länge[A]
2   do schlüssel = A[j]
3   i = j - 1
4     while i > 0 and A[i] > schlüssel
5       do A[i + 1] = A[i]
6          i = i - 1
7   A[i + 1] = schlüssel
```

tauschen



Insertion Sort - Laufzeitanalyse

- Für jedes $j = 2, 3, \ldots, n \ mit \ n = länge[A]$ gibt t_j an, wie viele Male der Test der while-Schleife in Zeile 4 für den Wert j ausgeführt wird
- Ist der Test negativ, so wird die Abfrage, aber nicht die Schleife ausgeführt.

```
INSERTIONSORT(A)
1 for j = 2 to länge[A]
2   do schlüssel = A[j]
3   i = j - 1
4     while i > 0 and A[i] > schlüssel
5       do A[i + 1] = A[i]
6       i = i - 1
7   A[i + 1] = schlüssel
```

Z	Kosten-koeffizient	Zeit
1	c_1	$n-1+1$
2	c_2	$n-1$
3	c_3	$n-1$
4	c_4	$\sum_{j=2}^{n} t_j$
5	c_5	$\sum_{j=2}^{n} (t_j-1)$
6	c_6	$\sum_{j=2}^{n} (t_j-1)$
7	c_7	$n-1$

Insertion Sort - Laufzeitanalyse

- Anzahl der Vergleiche und Verschiebungen ist von der Anordnung der Elemente in der unsortierten Eingangsfolge abhängig
- Average case: nicht einfach schätzbar
- Best case: bereits sortiertes Feld richtige Reihenfolge (kein Tauschen in Zeile 5 und 6)

$$T(n) = (c_1 + c_2 + c_3 + c_4 + c_7)n - (c_2 + c_3 + c_4 + c_7)$$

Z	Kosten-koeffizient	Zeit
1	c_1	n
2	c_2	$n-1$
3	c_3	$n-1$
4	c_4	$n-1$
5	c_5	$\sum_{j=2}^{n} t_j - 1$
6	c_6	$\sum_{j=2}^{n} t_j - 1$
7	c_7	$n-1$

Insertion Sort - Laufzeitanalyse

- Anzahl der Vergleiche und Verschiebungen ist von der Anordnung der Elemente in der unsortierten Eingangsfolge abhängig
- Average case: nicht einfach schätzbar
- Best case: bereits sortiertes Feld richtige Reihenfolge (kein Tauschen in Zeile 5 und 6)

$$T(n) = (c_1 + c_2 + c_3 + c_4 + c_7)n - (c_2 + c_3 + c_4 + c_7)$$ $O(n)$

- Worst case: bereits sortiertes Feld in falscher Reihenfolge (t_j=j) $O(n^2)$

Z	Kosten-koeffizient	Zeit
1	c_1	n
2	c_2	$n-1$
3	c_3	$n-1$
4	c_4	$\sum_{j=2}^{n} t_j$
5	c_5	$\sum_{j=2}^{n} (t_j-1)$
6	c_6	$\sum_{j=2}^{n} (t_j-1)$
7	c_7	$n-1$

$$T(n) = \left(\frac{c_4}{2} + \frac{c_5}{2} + \frac{c_6}{2}\right)n^2 + \left(c_1 + c_2 + c_3 + \frac{c_4}{2} - \frac{c_5}{2} - \frac{c_6}{2} + c_7\right)n - (c_2 + c_3 + c_4 + c_7)$$

-1, da j=2

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

- Insertion Sort

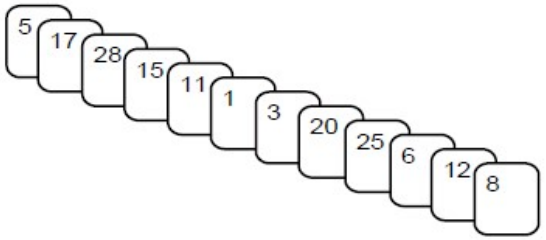


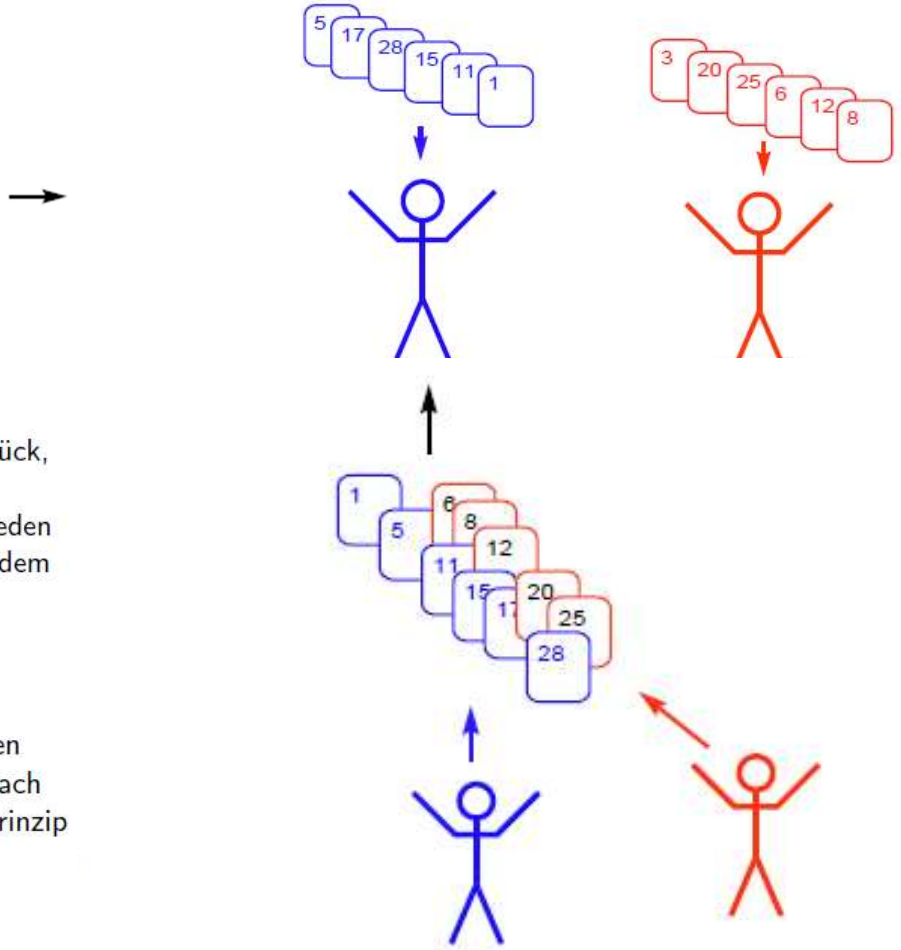
Schnelle Sortieralgorithmen - Merge Sort

- Sortieren durch Zusammenfügen
  - (engl. Merge Sort)
- Eigenschaften:
  - Rekursiver, stabiler Sortieralgorithmus
  - Prinzip: Teile und Herrsche (Divide and Conquer)
  - Im allgemeinen nicht In-place (Implementierungsabhängig)
  - Implementierung durch verkettete Listen besonders geeignet

Schnelle Sortieralgorithmen - Merge Sort

Gegeben: Stapel nummerierter Karten,
Aufgabe: sortiere aufsteigend von unten nach oben





Schritt 1: Aufteilen

Falls der Stapel nur aus einer Karte besteht, gib ihn sofort zurück, andernfalls:
Teile den Stapel in zwei möglichst gleich große Teile und gib jeden Teil je einem Gehilfen mit der Bitte, ihn ebenfalls genau nach dem hier beschriebenen Verfahren zu sortieren.

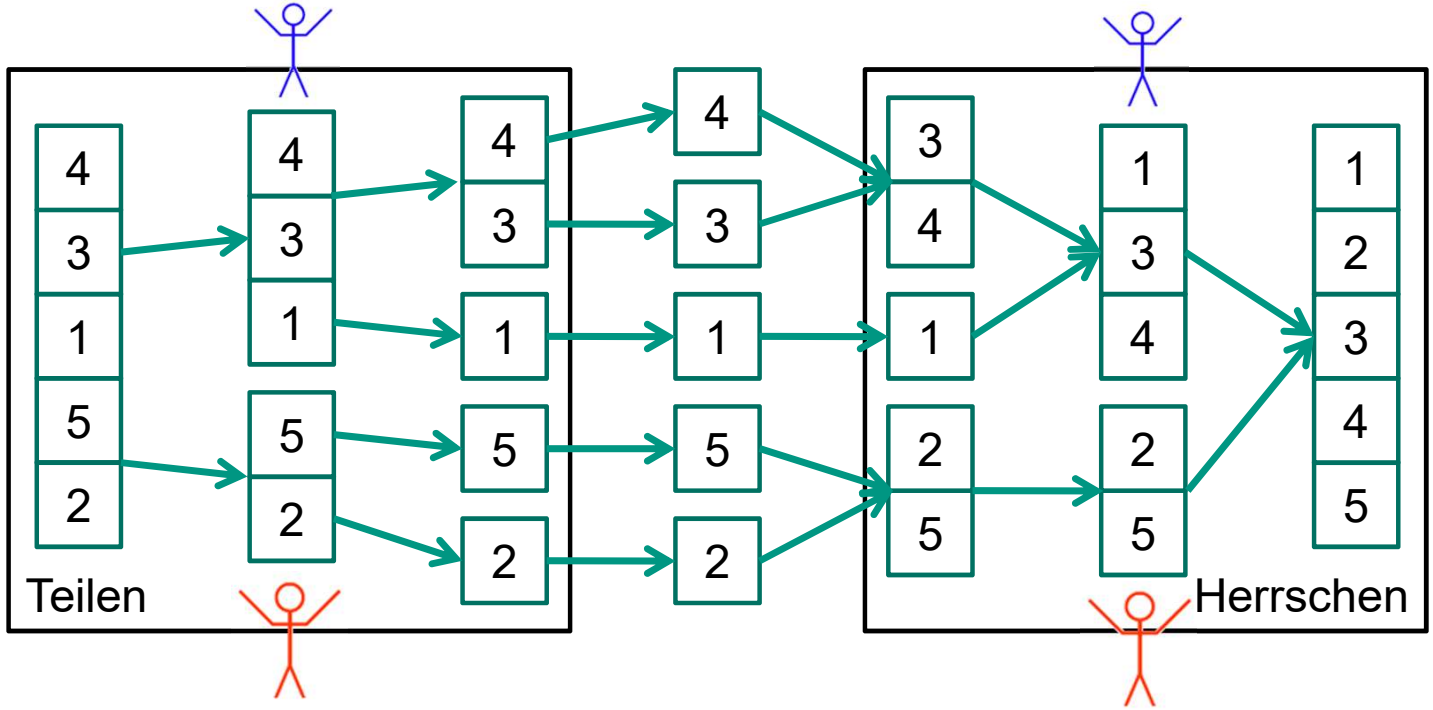
Schritt 2: Mischen

Warte bis dir beide Gehilfen die sortierten Teile zurückgegeben haben. Dann durchlaufe beide Stapel gleichzeitig von oben nach unten, und mische die Karten nach einer Art Reißverschlussprinzip zu einem sortierten Gesamtstapel zusammen.
Gib den sortierten Gesamtstapel zurück.

Merge Sort



Merge Sort - Verfahren

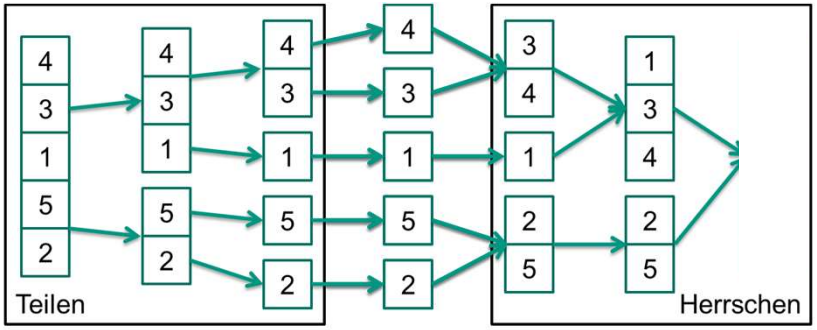


- Teile …
  - … das Problem in eine Menge von Teilproblemen
- Beherrsche …
  - … die Teilprobleme durch rekursives Lösen
- Verbinde …
  - … die Teillösungen zur Lösung des Gesamtproblems

Merge Sort - Pseudocode

```
Mergesort( A )
1   B = elements[A] 1 upto A/2   -- untere Hälfte
2   C = elements[A] A/2+1 upto length[A] -- obere Hälfte
3   return Merge( Mergesort( B ), Mergesort( C ) ) -- Rekursion
```

```
Merge( B, C )
1   create new list D
2   while B and C are not empty
3   do if B[1] > C[1]
4      then append C[1] to D and remove C[1]
5      else append B[1] to D and remove B[1]
6   return D
```

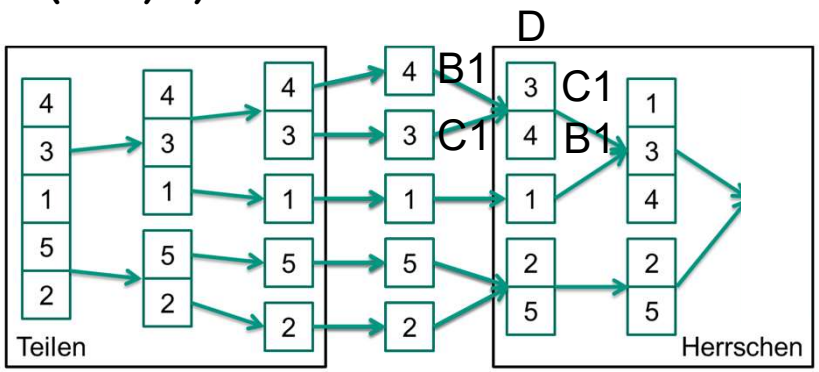


„remove“ setzt den Index eine Stelle weiter und am Ende des Vectors auf 0

Bei jedem Durchlauf wird nur nach dem ersten Element der Liste geschaut

Merge Sort - Pseudocode

```
Mergesort( A )
1   B = elements[A] 1 upto A/2
2   C = elements[A] A/2+1 upto length[A]
3   return Merge( Mergesort( B ), Mergesort( C ) )
```

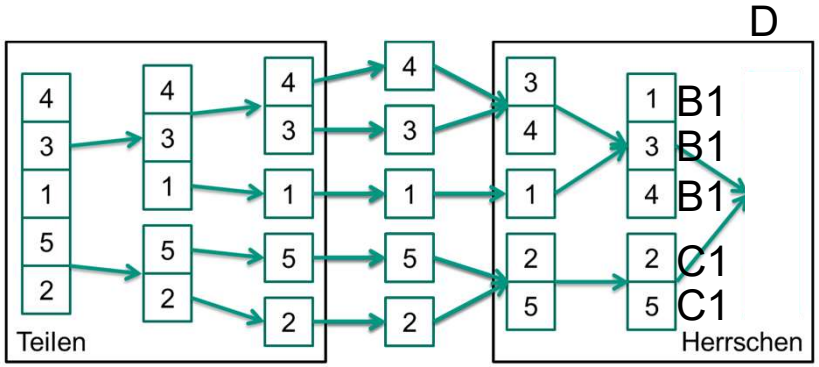



```
Merge( B, C )
1   create new list D
2   while B and C are not empty
3   do if B[1] > C[1]
4      then append C[1] to D and remove C[1]
5      else append B[1] to D and remove B[1]
6   return D
```

Merge Sort - Pseudocode

```
Mergesort( A )
1   B = elements[A] 1 upto A/2
2   C = elements[A] A/2+1 upto length[A]
3   return Merge( Mergesort( B ), Mergesort( C ) )
```

```
Merge( B, C )
1   create new list D
2   while B and C are not empty
3   do if B[1] > C[1]
4      then append C[1] to D and remove C[1]
5      else append B[1] to D and remove B[1]
6   return D
```



<Für Interessierte>

```
funktion mergesort(liste);
  falls (Größe von liste <= 1) dann antworte liste
  sonst
     halbiere die liste in linkeListe, rechteListe
     linkeListe = mergesort(linkeListe)
     rechteListe = mergesort(rechteListe)
     antworte merge(linkeListe, rechteListe)
```

```
funktion merge(linkeListe, rechteListe);
  neueListe
  solange (linkeListe und rechteListe nicht leer)
  |    falls (erstes Element der linkeListe <= erstes Element der rechteListe)
  |    dann füge erstes Element linkeListe in die neueListe hinten ein und entferne es aus linkeListe
  |    sonst füge erstes Element rechteListe in die neueListe hinten ein und entferne es aus rechteListe
  solange_ende
  solange (linkeListe nicht leer)
  |    füge erstes Element linkeListe in die neueListe hinten ein und entferne es aus linkeListe
  solange_ende
  solange (rechteListe nicht leer)
  |    füge erstes Element rechteListe in die neueListe hinten ein und entferne es aus rechteListe
  solange_ende
  antworte neueListe
```

<für Interessierte>

https://blog.digilentinc.com/recursive-sorting-algorithms/

```
void mergeSort(int list[], int n)
{
        int i;
        int lower[N], higher[N];
        int left, right;

        if (n < 2)
                return;   //base case

        // divide list into two array lower and higher
        left = n / 2;   // the number of elements in lower
        right = n - left;  // the number of elements in higher

        // move the first n/2 elements to lower
        for (i = 0; i < left; i++) {
                lower[i] = list[i];
        }
        // move the rest to higher
        for (i = 0; i < right; i++) {
                higher[i] = list[i + left];
        }
        // recursive calls
        mergeSort(lower, left);
        mergeSort(higher, right);

        // conquer
        merge(list, lower, higher, left, right);

}
```

```
/*
merge will combine two arrays into the sorted value
*/
void merge(int list[], int lower[], int higher[], int left, int right)
{
        //values to increment
        int i = 0, j = 0, k = 0;

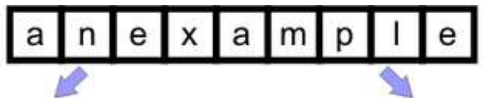
        //while there are still variables in lower and higher
        while (i < left && j < right)
        {
                //if the value of lower[i] <= higher[j]
                if (lower[i] <= higher[j])
                {
                        //assign value to list[k]
                        list[k] = lower[i];
                        i++;    //move to next value of lower
                }
                //value of higher < lower
                else
                {
                        //assign value to list[k]
                        list[k] = higher[j];
                        j++;    //move to next value of higher
                }

                //increment to next location in list
                k++;
        }

        //move leftover elements of lower into list
        while (i < left)
        {
                list[k] = lower[i];
                i++;
                k++;
        }
        //move leftover elements of higher into list
        while (j < right)
        {
                list[k] = higher[j];
                j++;
                k++;
        }
}
```

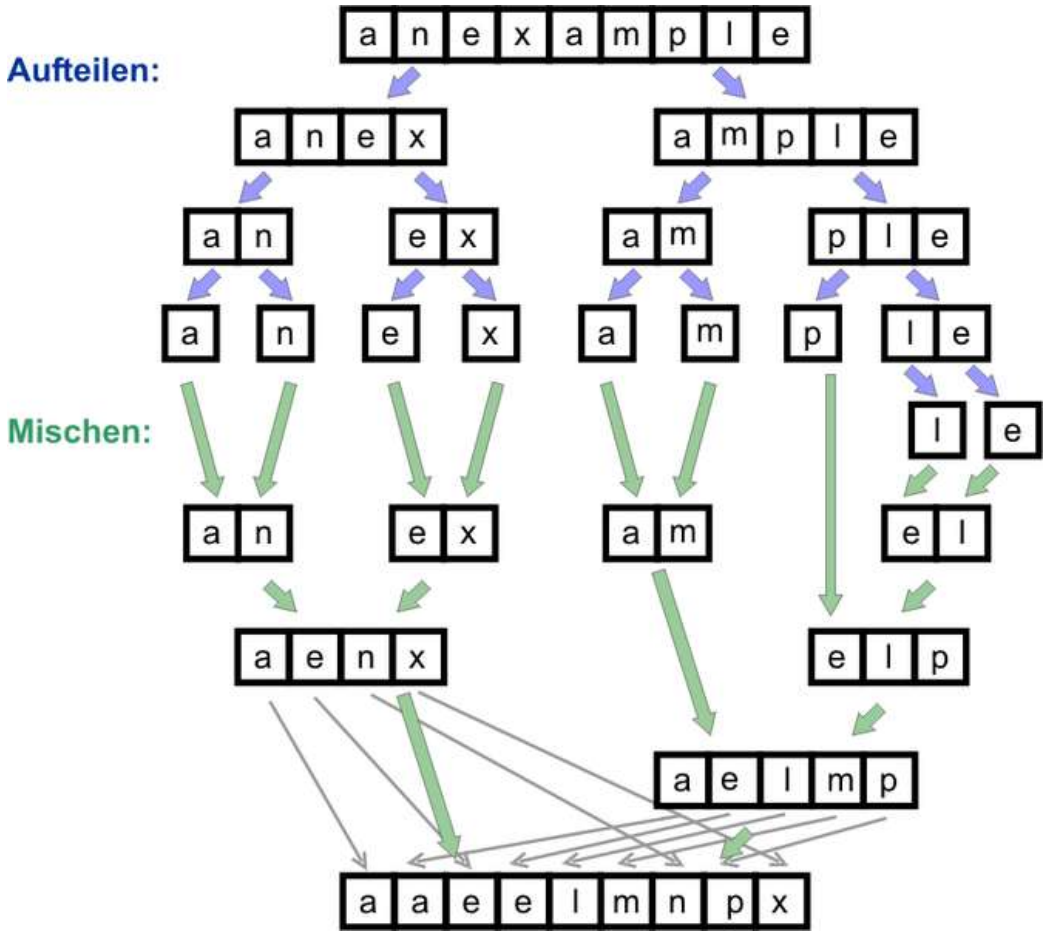
Merge Sort (Beispiel)

Aufteilen:



Im letzten Merge-Schritt ist das Reißverschlussverfahren beim Mischen angedeutet. Blaue Pfeile verdeutlichen den Aufteilungs-Schritt, grüne Pfeile die Misch-Schritte.

Merge Sort (Beispiel)



Im letzten Merge-Schritt ist das Reißverschlussverfahren beim Mischen angedeutet. Blaue Pfeile verdeutlichen den Aufteilungs-Schritt, grüne Pfeile die Misch-Schritte.

Merge Sort - Laufzeitanalyse

- Laufzeit $T(n)$ für Merge Sort der Länge n
  - Annahme:
    - konstante Zeit c für $n = 1$,
    - konstante Zeit auch pro Feldelement für Schritte Teilen und Mischen
- „Teile“:
  - Mitte des Felds berechnen: konstante Zeit $O(1)$
- „Beherrsche“:
  - löse zwei Teilprobleme der Größe $\frac{n}{2}$ rekursiv,
  - Laufzeitbeitrag: $2 \cdot T(\frac{n}{2})$
- „Verbinde“:
  - Laufzeitbeitrag von Prozedur Merge, um 2 Teile zu verschmelzen: $O(n)$

$$T(n) = \underbrace{T(\lfloor \tfrac{n}{2} \rfloor)}_{\text{Aufwand, den einen Teil zu sortieren}} + \underbrace{T(\lceil \tfrac{n}{2} \rceil)}_{\text{Aufwand, den anderen Teil zu sortieren}} + \underbrace{\mathcal{O}(n)}_{\text{Aufwand, die beiden Teile zu verschmelzen}} \Rightarrow 2 \cdot T(\tfrac{n}{2}) + cn$$

Merge Sort - Laufzeitanalyse

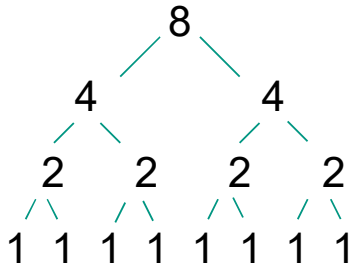
Veranschaulichung

- Binärbaum

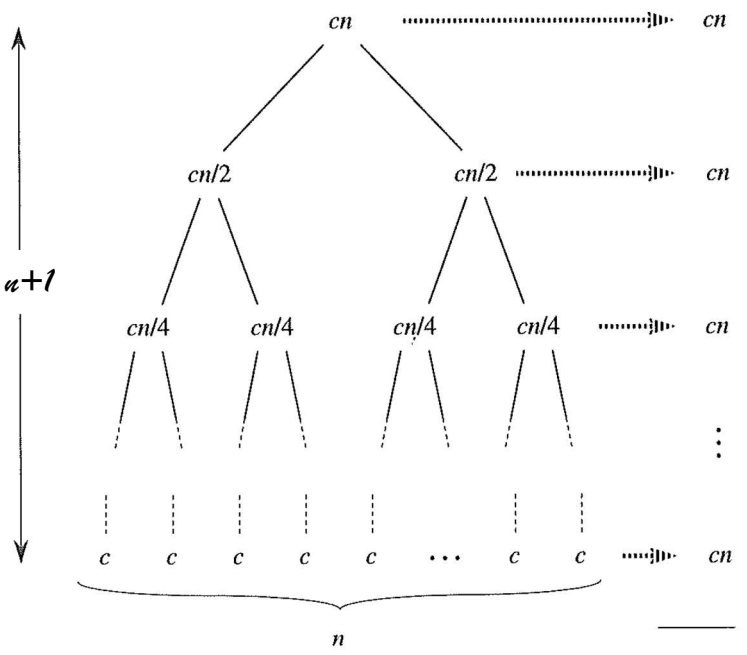
$$T(n) = \begin{cases} c & falls\ n = 1 \\ 2 \cdot T(\frac{n}{2}) + cn & falls\ n > 1 \end{cases}$$

Anzahl Ebenen $ld\, n + 1$

Laufzeit $cn \cdot (ld\, n + 1)$ → **O(n ld n+n)**



- Für Teilen und Mischen



Remark (s. Folgefolien): Hinsichtlich der Komplexität im worst case dem nachfolgend dargestellten Sortierverfahren Quick Sort überlegen, denn Quick Sort hat im worst case O(n^2).

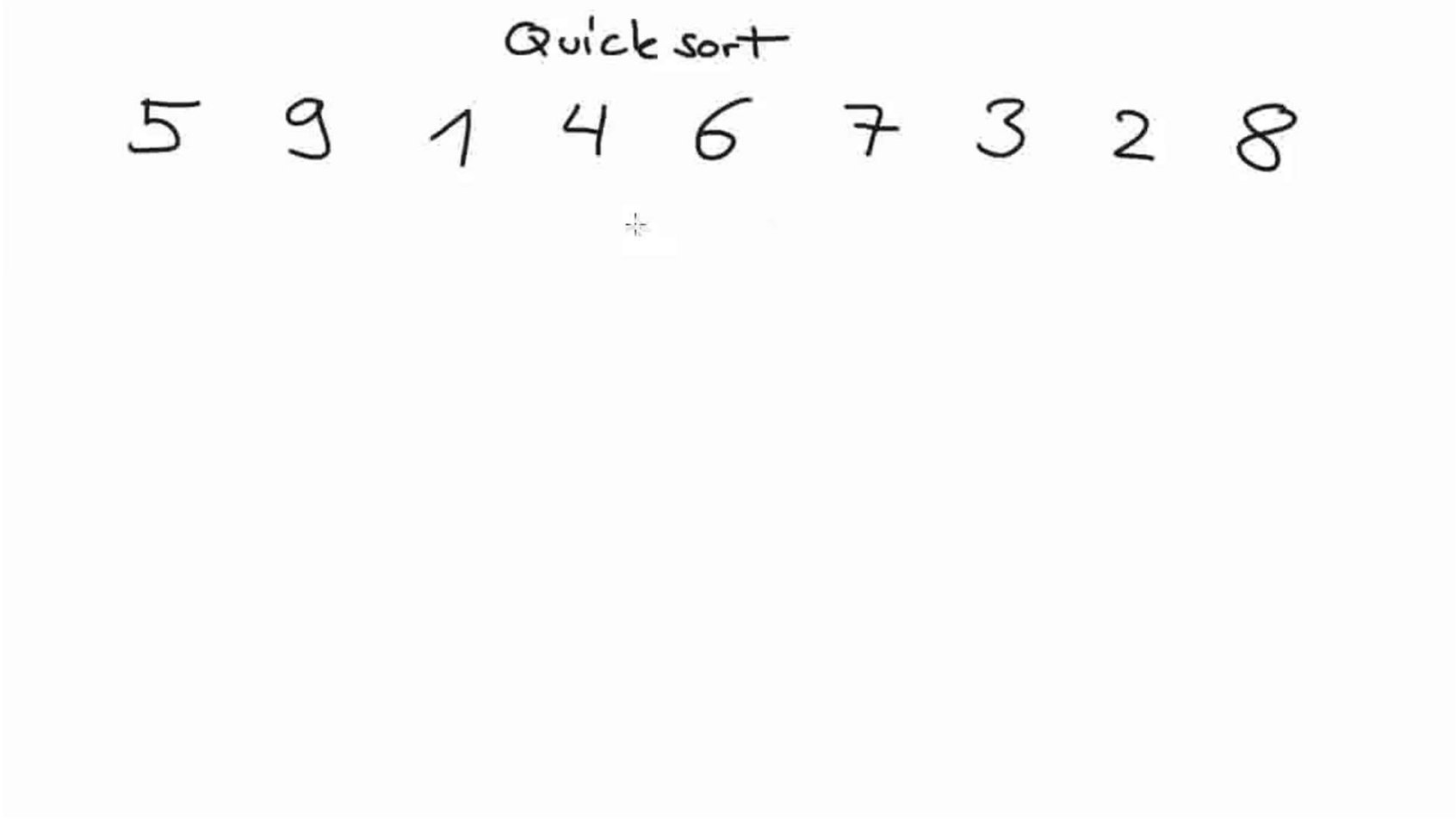
- Merge Sort



Quick Sort

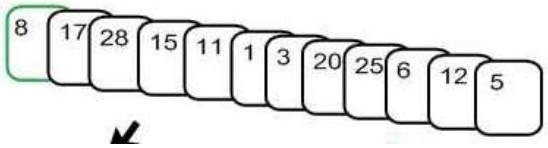
- Quick Sort
  - (C. Antony, R. Hoare: ca. 1960)

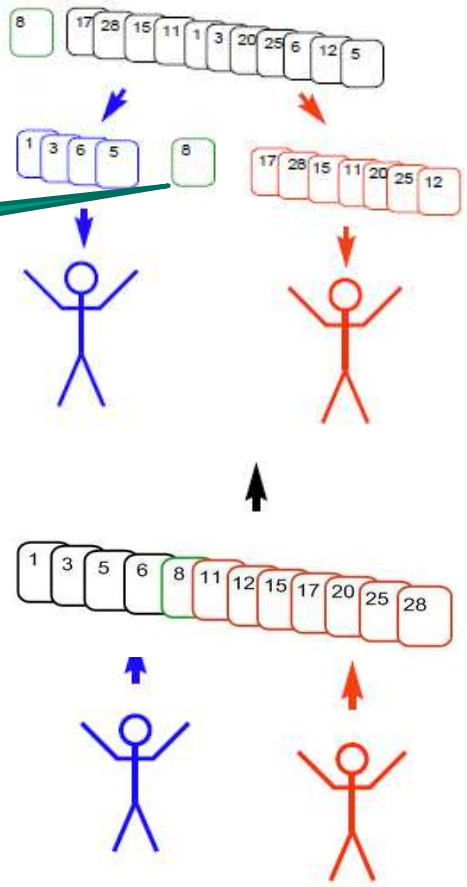
- Eigenschaften
  - nicht stabiler Sortieralgorithmus
  - Prinzip: Teile und Herrsche
  - sehr kurze innere Schleife erhöht Ausführungsgeschwindigkeit
  - kommt ohne zusätzlichen Speicherplatz aus
  - lässt sich einfach durch Rekursionen realisieren
  - im Mittel schnellster Suchalgorithmus: $O(n*\log_2 n)$


Quicksort
5 9 1 4 6 7 3 2 8

Schnelle Sortieralgorithmen - Quick Sort

Gegeben: Stapel nummerierter Karten,
Aufgabe: sortiere aufsteigend von unten nach oben





Schritt 1: Aufteilen

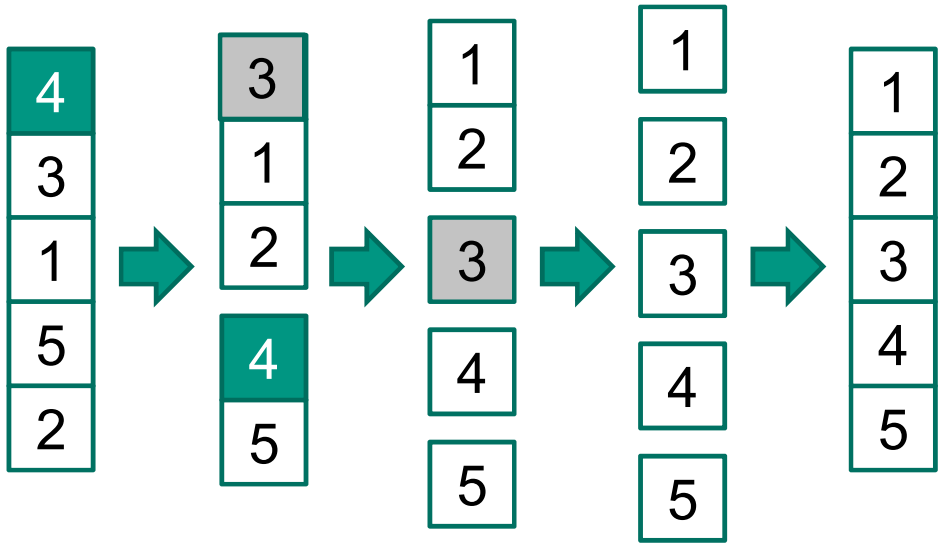
Falls der Stapel nur aus einer Karte besteht, gib ihn sofort zurück, andernfalls: Nimm die erste Karte aus dem Stapel. Durchlaufe die restlichen Karten und teile sie auf:
Stapel 1: alle mit Wert kleiner oder gleich dem der ersten Karte
Stapel 2: alle mit Wert größer als dem der ersten Karte.
Teilstapel, wenn sie überhaupt Karten enthalten, an je einen Gehilfen geben, der sie ebenfalls genau nach dem hier beschriebenen Verfahren sortiert.

Schritt 2: Zusammensetzen

Warte bis dir beide Gehilfen die sortierten Teile zurückgegeben haben, dann lege zuunterst den sortierten Stapel 1, darauf die anfangs gezogene Karte, darauf den sortierten Stapel 2 und gib das Ganze als sortiert zurück.

[1]Dreh- und Angelpunkt

Quick Sort - Verfahren



- In jedem Schritt wird ein Pivotelement gewählt
- Elemente, die größer sind als das Pivotelement werden in die eine Hälfte, Elemente die kleiner oder gleich sind in die andere Hälfte sortiert

Quick Sort - Pseudocode

Zum Start: `Quicksort( A, 1, länge[A] )`

```
Quicksort( A, p, r )
1   if p < r
2   then q = Partition( A, p, r )
3        Quicksort( A, p, q - 1 )
4        Quicksort( A, q + 1, r )
```

```
Partition( A, p, r )
1   x = A[r]
2   i = p - 1
3   for j = p to r - 1
4   do if A[j] <= x
5       then i = i + 1
6       vertausche A[i] mit A[j]
7   vertausche A[i+1] mit A[r]
8   return i+1
```

Quick Sort – Pseudocode

in place Arbeitsweise

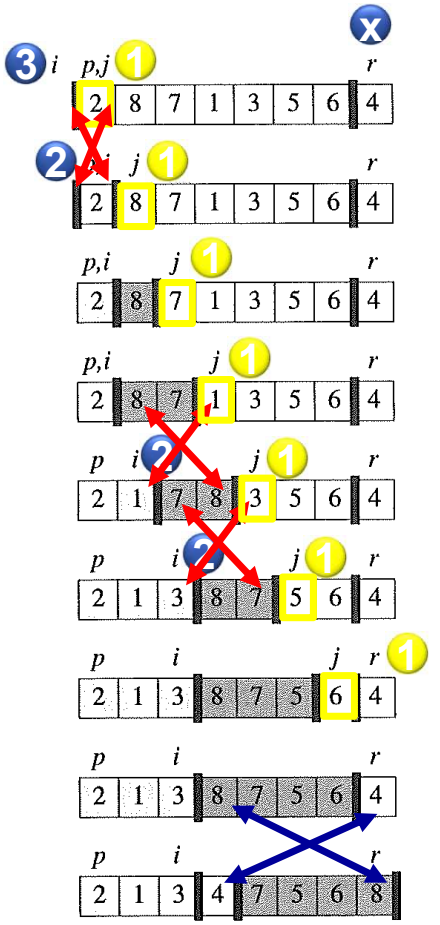
```
Quicksort( A, p, r )
1   if p < r
2   then q = Partition( A, p, r )
3        Quicksort( A, p, q - 1 )
4        Quicksort( A, q + 1, r )
```

Nächste Schritte!

```
Partition( A, p, r )
1   x = A[r]
2   i = p - 1
3   for j = p to r - 1
4   do if A[j] <= x
5        then i = i + 1
6        vertausche A[i] mit A[j]
7   vertausche A[i+1] mit A[r]
8   return i+1
```

Pivotelement

Teile nach Größe



1. Schritt!

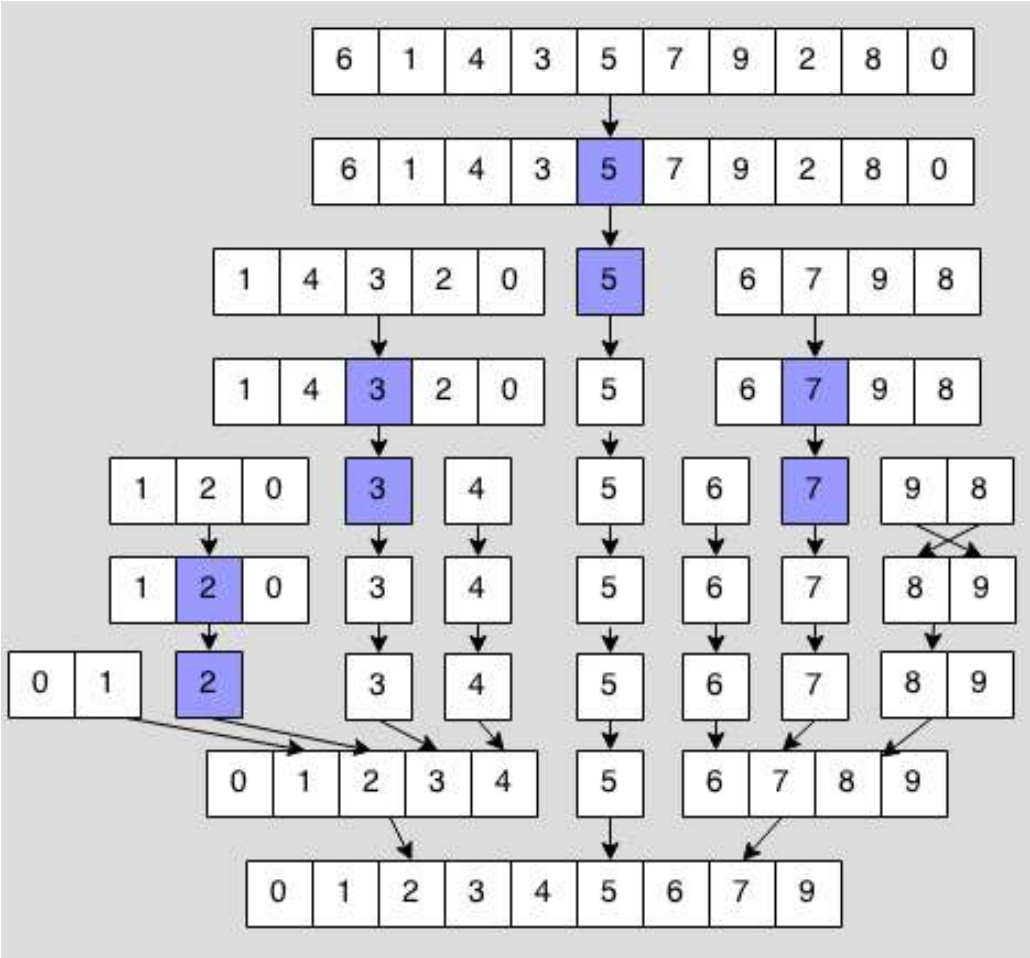
Beispiel Quicksort

Pivot-Element aus der Mitte!

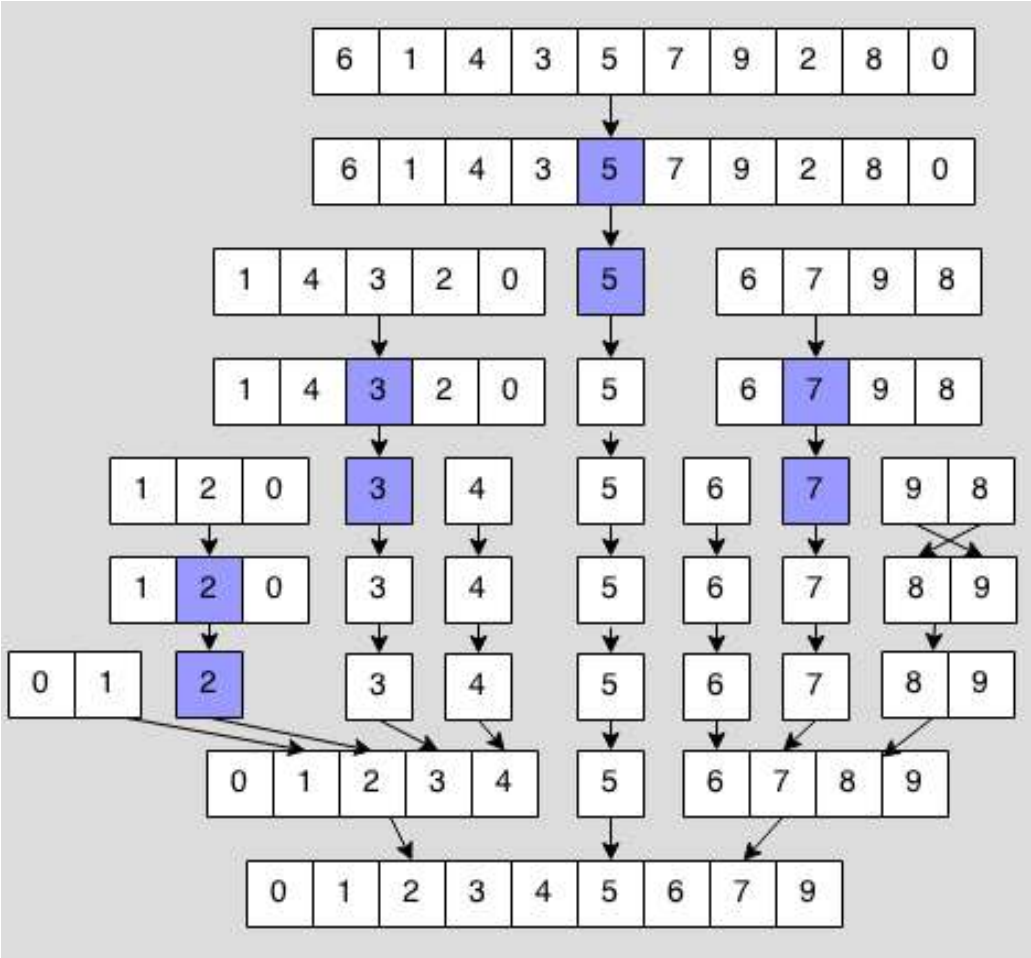


Beispiel Quicksort

Pivot-Element aus der Mitte!

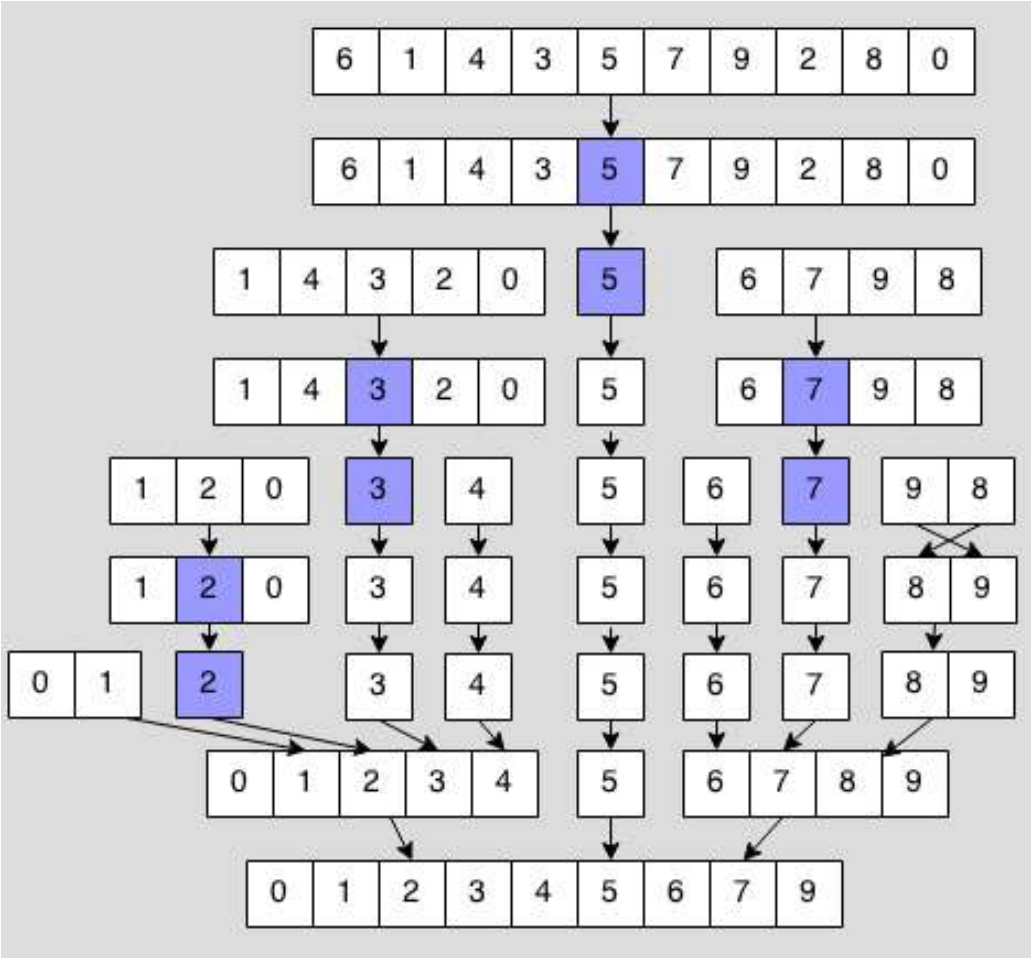


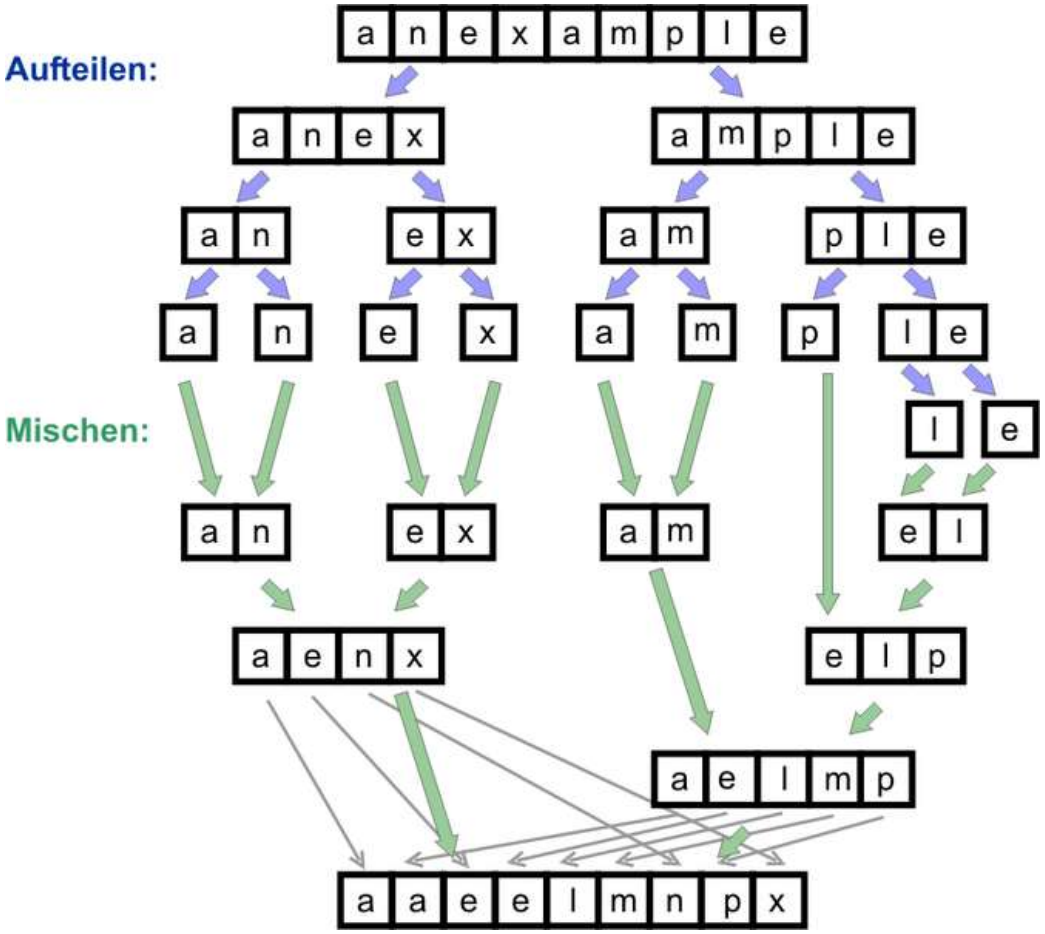
Quick Sort - Laufzeitanalyse

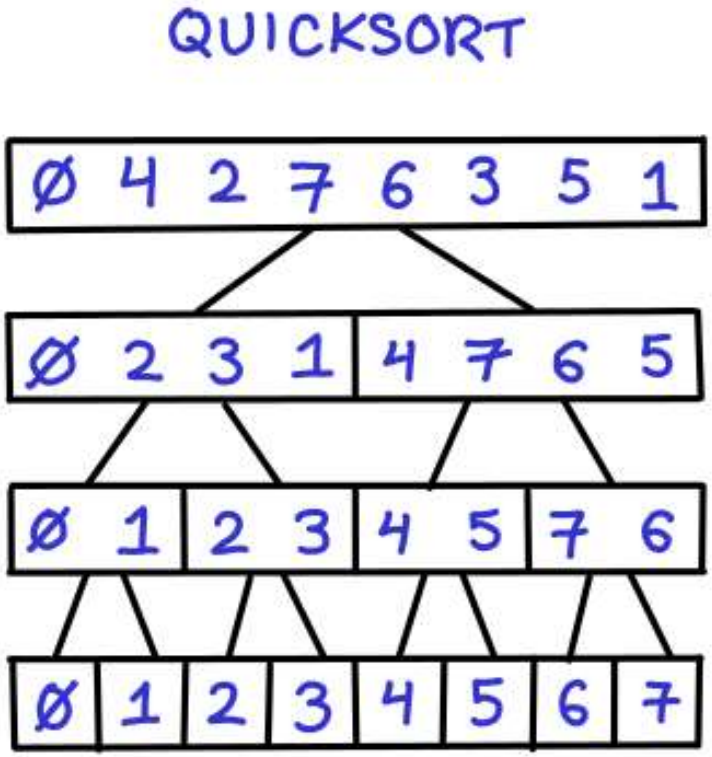


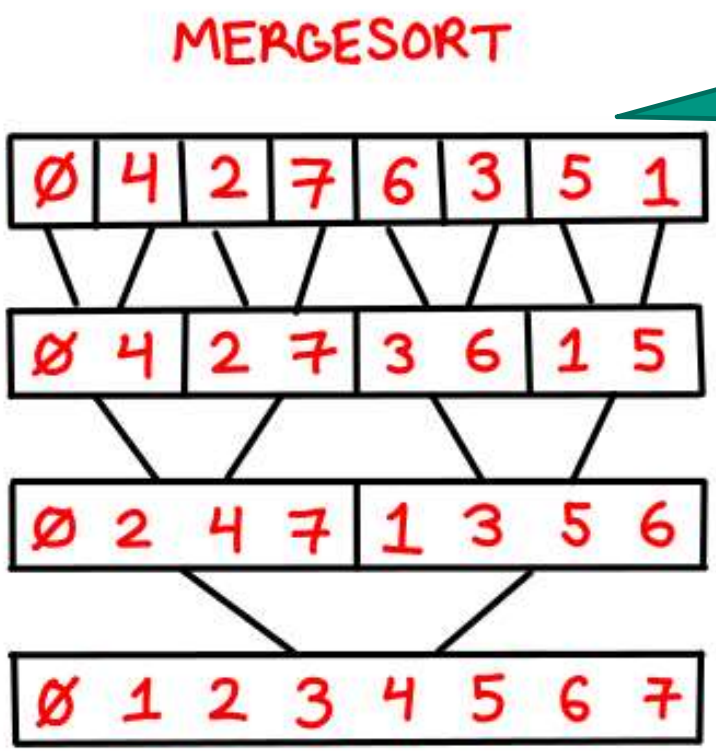
- Laufzeit hängt davon ab, ob die Zerlegung balanciert oder unbalanciert ist:
  - schlechteste Zerlegung:
    - ein Teilfeld Länge 0, zweites Teilfeld Länge n-1 ergibt $O(n^2)$.
    - $O(n^2)$ auch wenn Eingabefeld bereits vollständig geordnet.
  - beste Zerlegung:
    - ein Teilfeld Länge n/2, zweites Teilfeld Länge n/2-1 ergibt O(n ld n)
  - balancierte Zerlegung:
    - die mittlere Laufzeit von Quicksort ist viel näher an der des besten Falles als an der des schlechtesten Falles z.B. bei einer Zerlegung im Verhältnis 1:9 auch O(n ld n),
    - dies gilt auch noch z.B. bei Verhältnis 1:99
      - Untersuchungen zeigen: Partition liefert in >80% der Fälle besser balanciert als 1:9.
- Verbesserungspotential durch geeignete Wahl des Pivot-Elementes.
- Fazit: Im Mittel schnellster Sortieralgorithmus, in den Grenzfällen jedoch langsamer (gleiches Laufzeitverhalten wie Merge Sort)

Quick Sort - Laufzeitanalyse

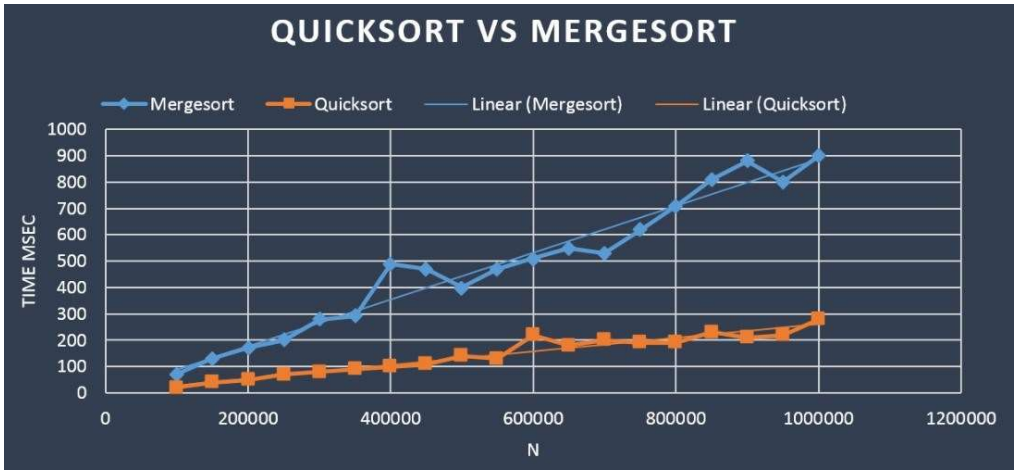


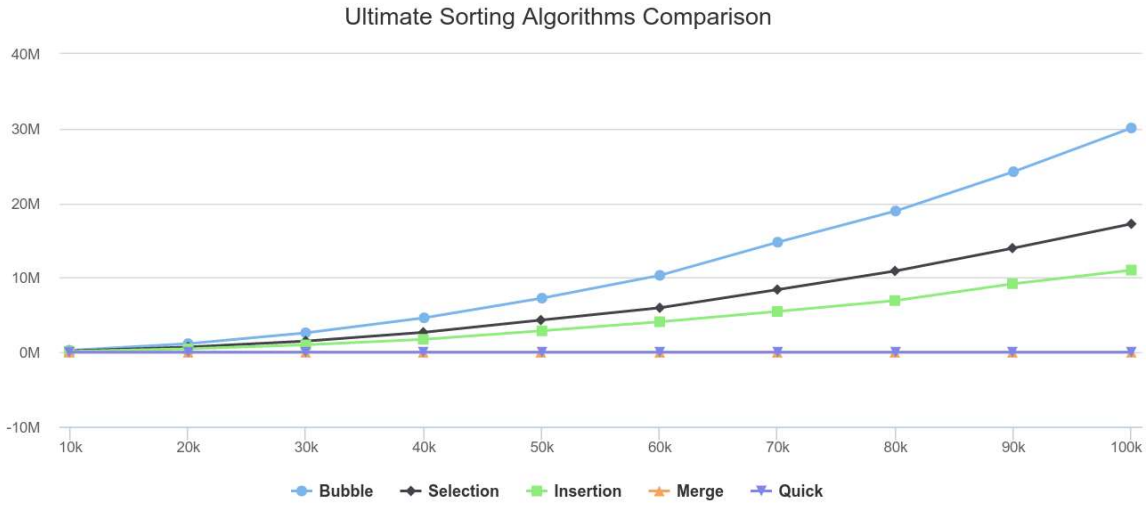






Hier hat der Teilungsprozess schon stattgefunden

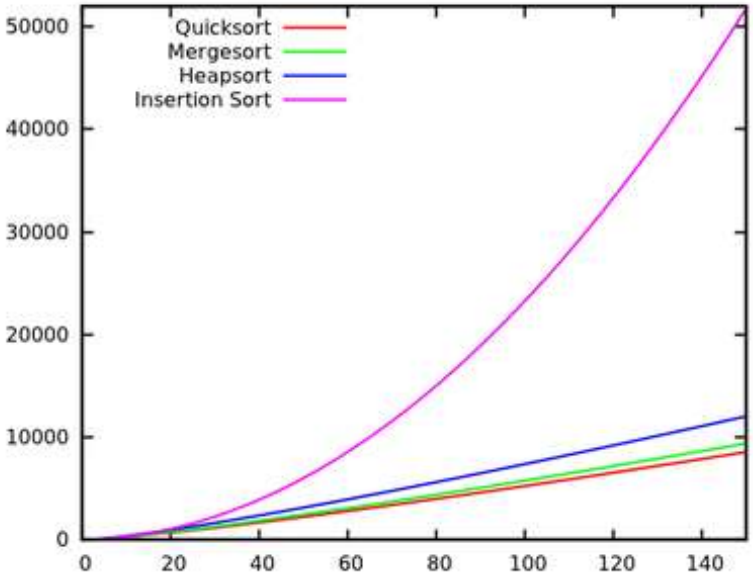

QUICKSORT VS MERGESORT
Mergesort
Quicksort
Linear (Mergesort)
Linear (Quicksort)
TIME MSEC
1000
900
800
700
600
500
400
300
200
100
0
0
200000
400000
600000
800000
1000000
1200000
N


Ultimate Sorting Algorithms Comparison
40M
30M
20M
10M
0M
-10M
10k
20k
30k
40k
50k
60k
70k
80k
90k
100k
Bubble
Selection
Insertion
Merge
Quick

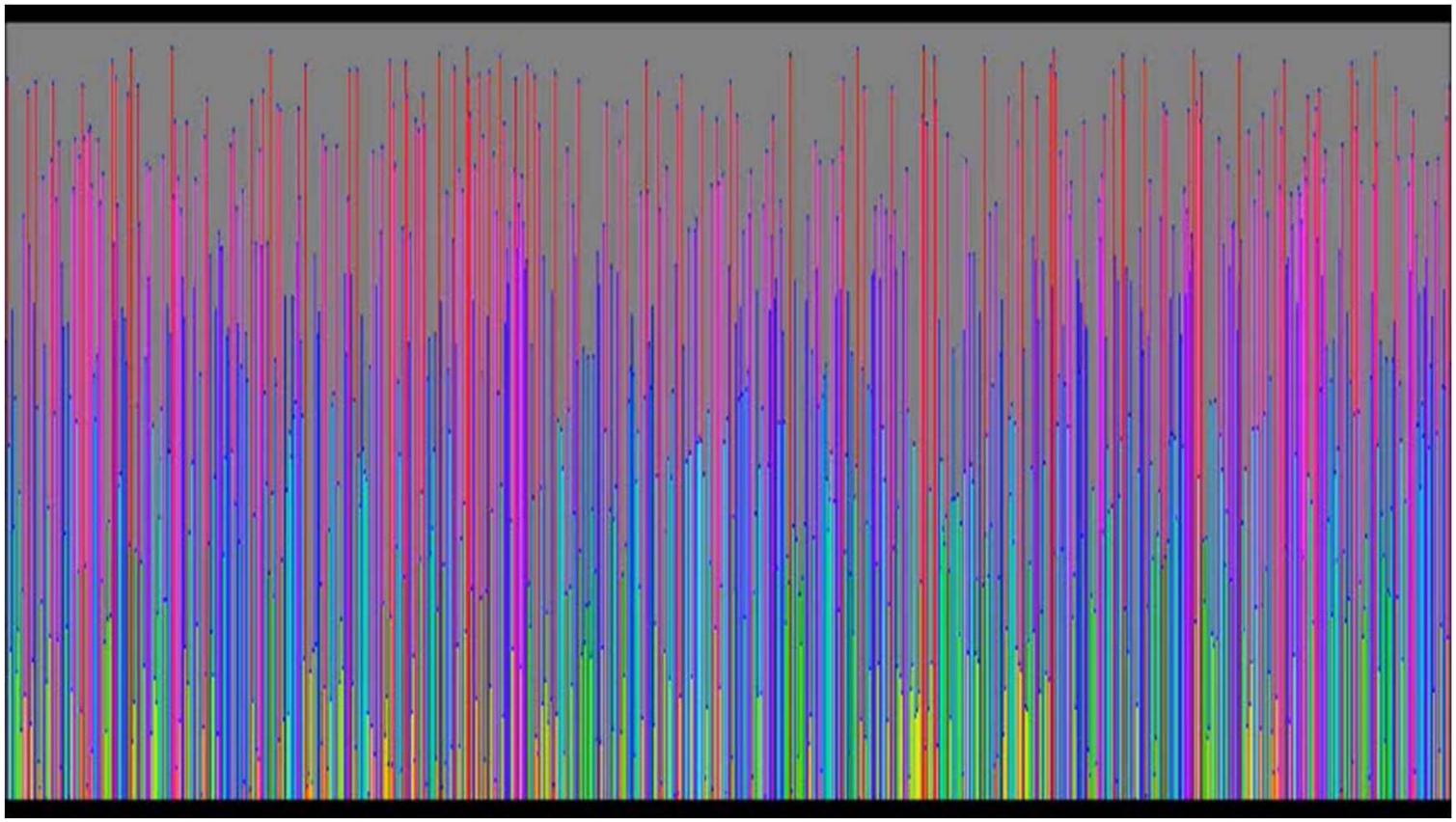
Warum ist der Quick Sort „im Mittel“ schneller?

This is done in Don **Knuth**'s book series "The Art of Computer Programming" for an artificial "typical" computer invented by the author. In volume 3 you find exact **average case results** for many sorting algorithms, e.g.

- Quicksort: $11.667(n+1)\ln(n) - 1.74n - 18.74$
- Mergesort: $12.5n\ln(n)$
- Heapsort: $16n\ln(n) + 0.01n$
- Insertionsort: $2.25n^2 + 7.75n - 3ln(n)$



http://www.sorting-algorithms.com/



- Quick Sort




Backup
Shift
Insert

Reihen

Spezielle Reihen

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2} \qquad \sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{k=0}^{n} x^k = \frac{1-x^{n+1}}{1-x} \qquad \sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \quad (|x| < 1)$$

$$\sum_{k=0}^{\infty} \frac{x^k}{k!} = e^x \qquad \sum_{k=1}^{\infty} \frac{x^k}{k} = -\ln(1-x) \quad (|x| < 1)$$

$$\sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{(2k+1)!} = \sin x \qquad \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k)!} x^{2k} = \cos x$$

Harmonische Reihe

$\sum_{k=0}^{\infty} \frac{1}{k^\alpha}$, konvergent für $\alpha > 1$, divergent für $\alpha \leq 1$



Karlsruher Institut für Technologie

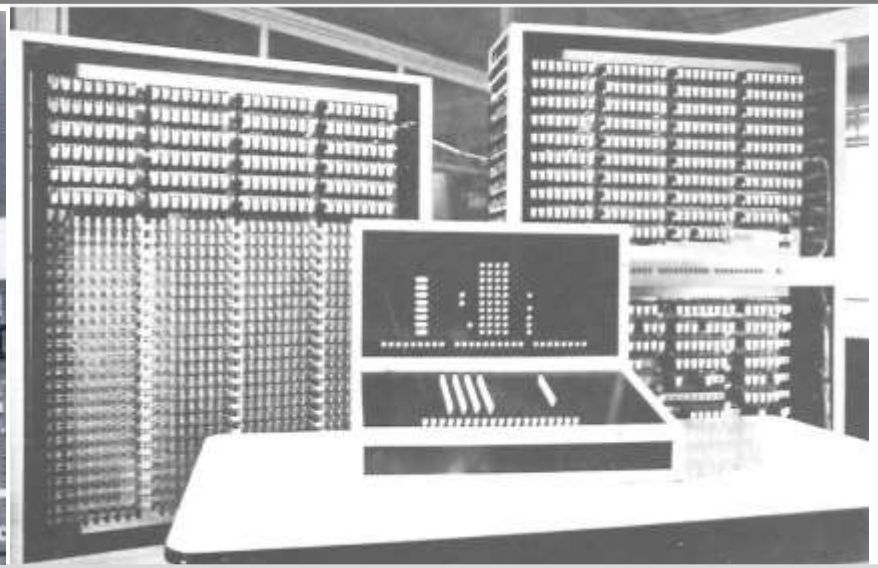
Institut für Technik der Informationsverarbeitung

Institutsleitung
Prof. Dr.-Ing. J. Becker
Prof. Dr.-Ing. E. Sax
Prof. Dr. rer. nat. W. Stork

4. Suchalgorithmen

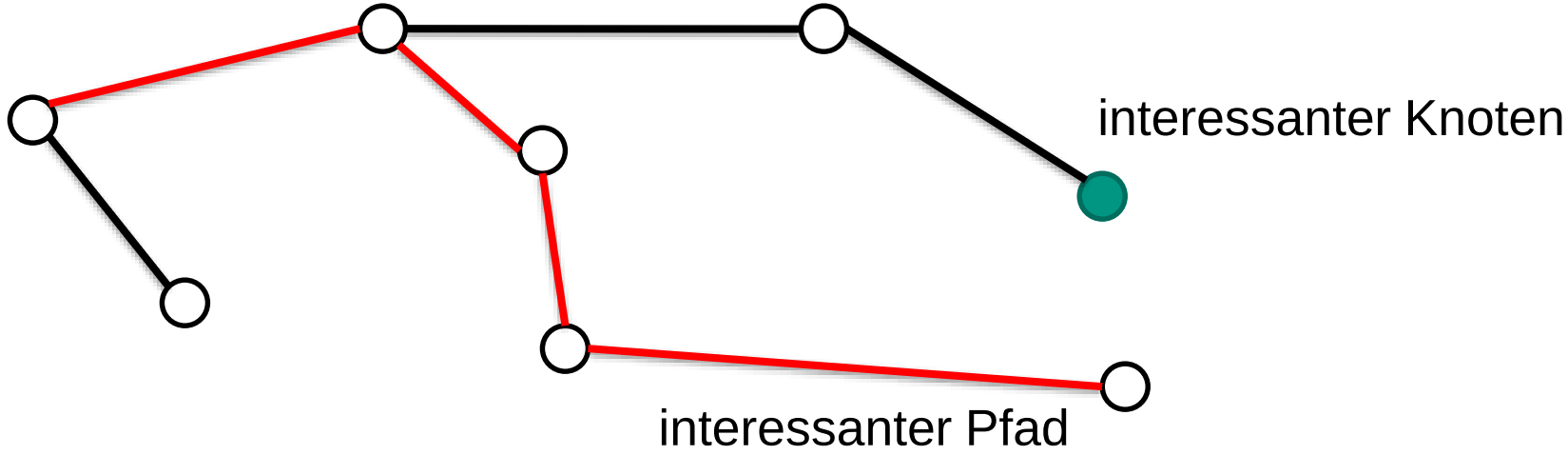
Informationstechnik II

Prof. Dr.-Ing. Eric Sax



Suchaufgabe in Graphen



- Zwei typische Problemstellungen in der Praxis:
  - Finden eines bestimmten Knotens
  - Finden eines bestimmten Pfades im Graph

Das Suchproblem

- Viele Probleme der Graphentheorie können mit Suchalgorithmen gelöst werden.
- Suchen ist ein klassisches Problem der Informatik.

- Definition:
  „Ein Suchalgorithmus sucht in einer Gesamtmenge von Daten (Suchraum) nach Mustern oder Objekten mit bestimmten Eigenschaften“

- Auch außerhalb der Graphentheorie stellt sich das Suchproblem für viele Datenstrukturen
- Vielfältige Lösungen sind vorhanden mit unterschiedlichen Anforderungen an Ressourcen und Geschwindigkeit

Suchen und Sortieren

- Geordnete oder sortierte Strukturen reduzieren die Laufzeit eines Suchalgorithmus.
- Der Sortieralgorithmus benötigt ebenfalls Zeit.
- Besonders nützlich bei mehreren Suchvorgängen und nur einmaligem Sortieren.
- Problem:
  - Der Suchalgorithmus benötigt Kenntnis über die Struktur der Daten

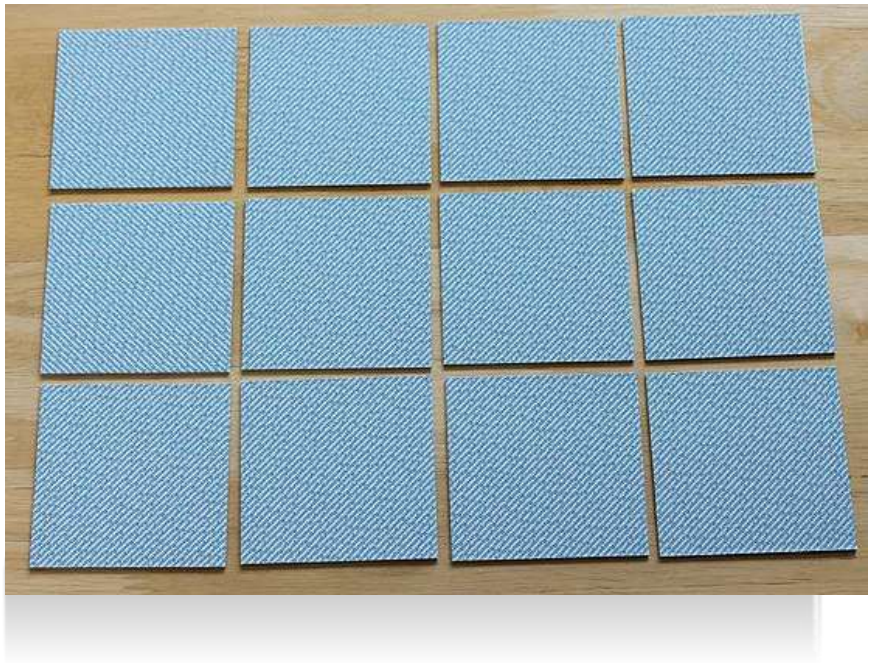


Lineare Suche

„Memory-Ansatz“ ohne Vorwissen

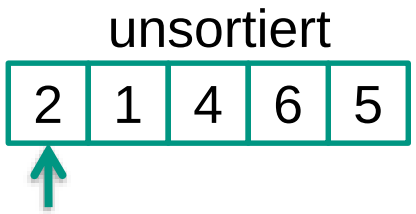
```
linearSearch( list, key )
   for each element in list
   do if element == key
      then return success

   return failure
```



- Naive Lösung:
  - Jedes Element der Datenmenge wird untersucht
- Elemente der Datenmenge unterliegen keiner Struktur oder Sortierung
- Oft eingesetzt zur Suche in Listen

Lineare Suche – Aufwand (unsortiert)



Stelle, an der die Suche nach „3“ abgebrochen wird?

- Durchschnittlicher Aufwand: O(n)
  - n/2, falls das Element in der Liste vorhanden ist (Gleichverteilung auf [1..n])
  - n, falls das Element nicht in der Liste steht
- Verbesserungsversuch:
  - Sortierung der Liste
- Eventuell kann entschieden werden ob ein Element in der Liste ist, ohne diese komplett zu durchlaufen

Lineare Suche – Aufwand (sortiert)



Stelle, an der die Suche nach „3“ abgebrochen wird

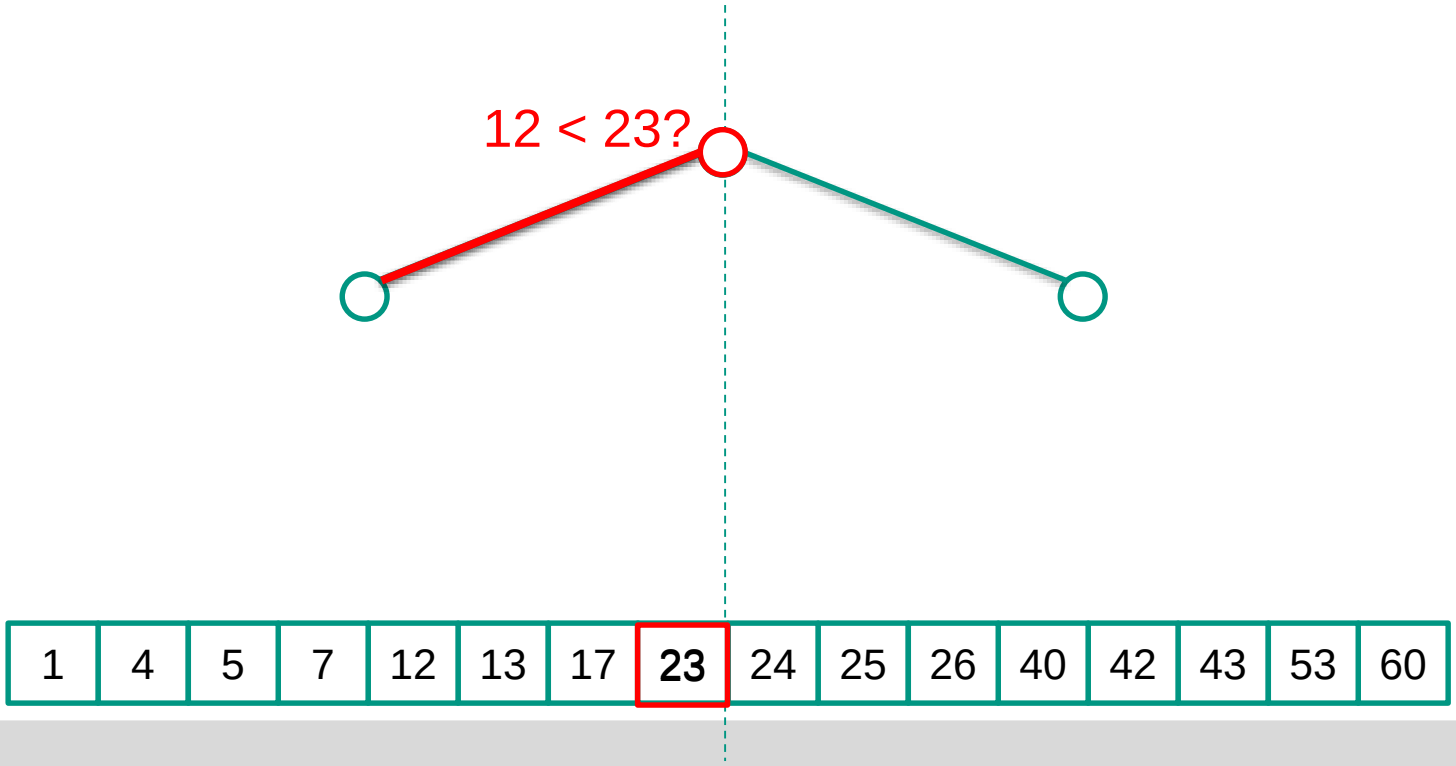
- Durchschnittlicher Aufwand: O(n)
  - n/2, unabhängig davon, ob das Element in der Liste vorhanden ist oder nicht (Gleichverteilung auf [1..n])
- Suche mit sublinearem Aufwand möglich?
- Idee: Verwendung einer geeigneten Kombination aus Datenstruktur und Suchalgorithmus

Binäre Suche

- Vorgehen:
  - Die Binärsuche springt rekursiv immer in die Mitte des noch zu durchsuchenden Intervalls und überprüft, ob der gesuchte Wert größer oder kleiner als das so bestimmte Pivotelement ist.
  - Der Suchraum wird bei jedem Schritt halbiert.
- Vorraussetzungen:
  - Sortierte Datenstruktur mit freiem Zugriff auf Elemente (Array)
- Komplexität (worst case):
  - Aufwand der Binärsuche ist O(ld(n) (→ Suche im Binärbaum der Tiefe ld(n)) und damit sublinear
  - S. Merge Sort und Quick Sort

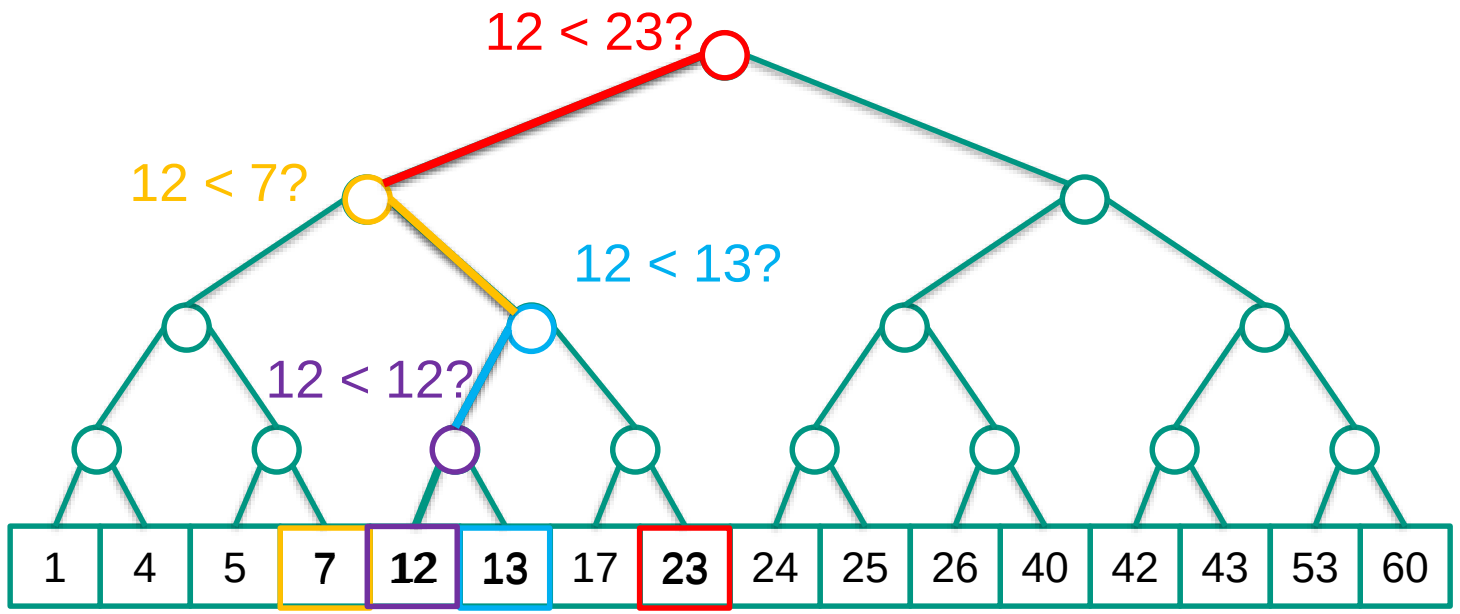
Beispiel: Binäre Suche

- Vorgehen:
  - Die Binärsuche springt rekursiv immer in die Mitte des noch zu durchsuchenden Intervalls und überprüft, ob der gesuchte Wert größer oder kleiner als das so bestimmte Pivotelement ist
  - Der Suchraum wird bei jedem Schritt halbiert
  - Beispiel: *Suche nach 12*



Beispiel: Binäre Suche

- Vorgehen:
  - Die Binärsuche springt rekursiv immer in die Mitte des noch zu durchsuchenden Intervalls und überprüft, ob der gesuchte Wert größer oder kleiner als das so bestimmte Pivotelement ist
  - Der Suchraum wird bei jedem Schritt halbiert
  - Beispiel: Suche nach 12



Beispiel: Binäre Suche (rekursive Variante)

```
binarySearch( list, key, bottom, top )
  center = ( bottom + top ) DIV 2
  if list[center] == key
  then return center
  elseif top - bottom > 0
  then if key < list[center]
    then return binarySearch( list, key, bottom, center )
    else return binarySearch( list, key, center + 1, top )
```

→ (list, 12, 1, 16)

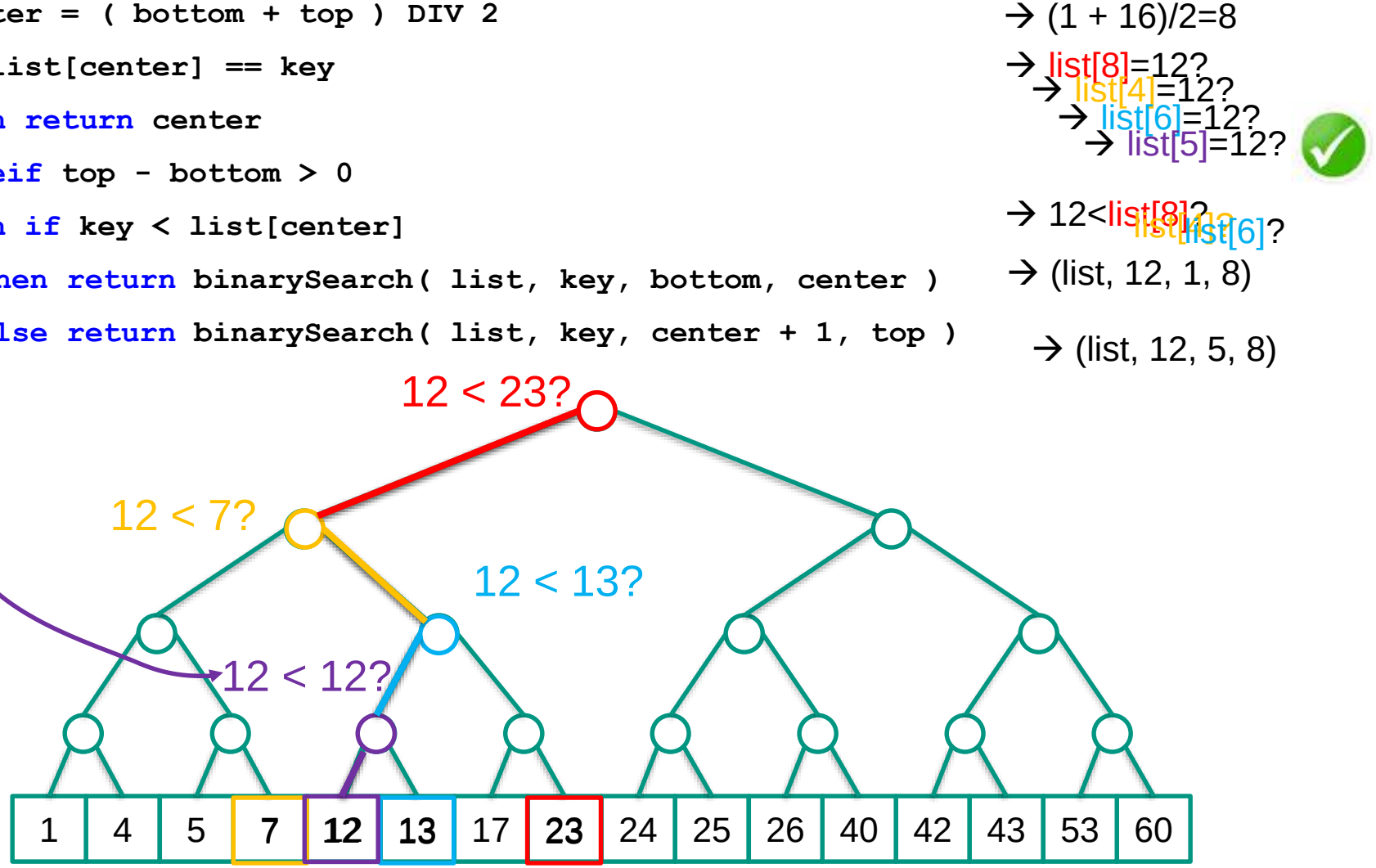
→ (1 + 16)/2=8

→ list[8]=12?

→ list[4]=12?

→ list[6]=12?

→ list[5]=12?



→ 12<list[8]?

→ list[4]?

→ list[6]?

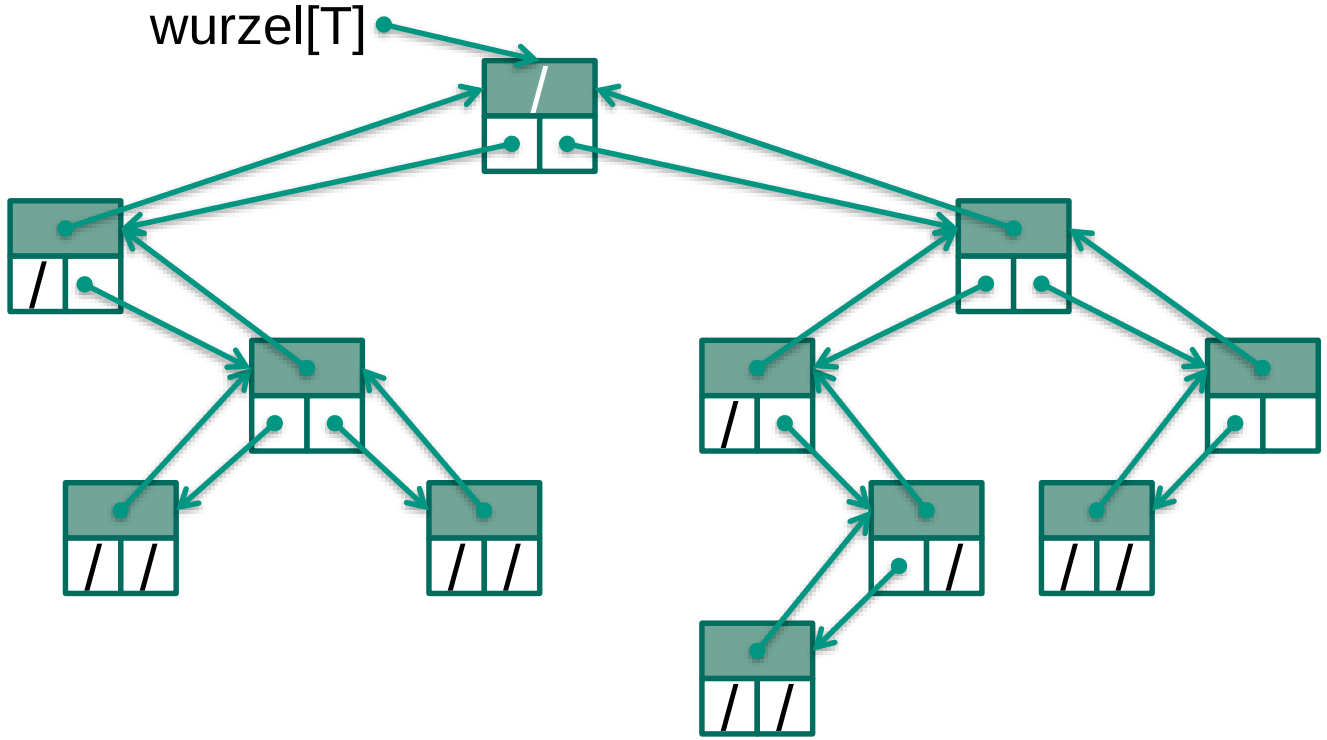
→ (list, 12, 1, 8)

→ (list, 12, 5, 8)

Prof. Dr.-Ing. Eric Sax, © 2019

Binärbaum: Repräsentation

- Statt eines Arrays kann zur Speicherung auch eine Baumstruktur als speziell verkettete Liste verwendet werden
- Dabei besteht jeder Knoten aus dem Wert und drei Zeigern:
  Vorgänger, linker Nachfolger und rechter Nachfolger



- Binäre Suche



Pause



Planänderung

- 22.05. Ersatztermin Vorlesung
- 24.05. Vorlesung fällt aus
- 29.05. Ersatztermin Übung

Veranstaltungsplan Informationstechnik 2 SS2019

Vorlesung/Übung:	Mi, 8:00 - 9:30 Uhr	Gebäude 30.22, Gaede-Hörsaal
Vorlesung/Übung:	Fr, 9:45 - 11:15 Uhr	Gebäude 30.22, Gaede-Hörsaal

Woche	Datum Mi.	Vorlesung	Datum Fr.	Übung	Inhalt Mi.	Inhalt Fr.
1.	24. Apr		26. Apr	VL		VL0: Einleitung und Motivation
2.	01. Mai	frei	03. Mai	VL	Tag der Arbeit	VL1: Algorithmische Grundlagen
3.	08. Mai	ÜB	10. Mai	VL	ÜB1: Organisatorisches; Besprechung Aufg. 1;	VL2: Sortieralgorithmen
4.	15. Mai		17. Mai	VL		VL3: Algorithmen auf Graphen
5.	22. Mai	VL	24. Mai	frei	VL4: Optimierungsalgorithmen	frei
6.	29. Mai	ÜB	31. Mai	VL	ÜB2: Besprechung Aufg. 2;	VL5: Big Data Einführung
7.	05. Jun	ÜB	07. Jun	VL	ÜB3: Besprechung Aufg. 3;	VL6: Big Data - Prozesse
8.	12. Jun	frei	14. Jun	frei	Pfingsten	Pfingsten
9.	19. Jun	ÜB	21. Jun	VL	ÜB4: Besprechung Aufg. 4;	VL7: Big Data - Modelling 1
10.	26. Jun	ÜB	28. Jun	VL	ÜB5: Besprechung Aufg. 5;	VL8: Big Data - Modellin 2
11.	03. Jul		05. Jul	VL		VL9: Big Data - Evaluation
12.	10. Jul	ÜB	12. Jul	VL	ÜB6: Besprechung Aufg. 6;	VL10: Big Data - Infrastruktur
13.	17. Jul	ÜB	19. Jul	VL	ÜB7: Besprechung Aufg. 7;	VL11: Anwendungen Institut
14.	24. Jul	ÜB	26. Jul	VL	ÜB8: Besprechung Aufg. 8;	VL12: Cyber Security

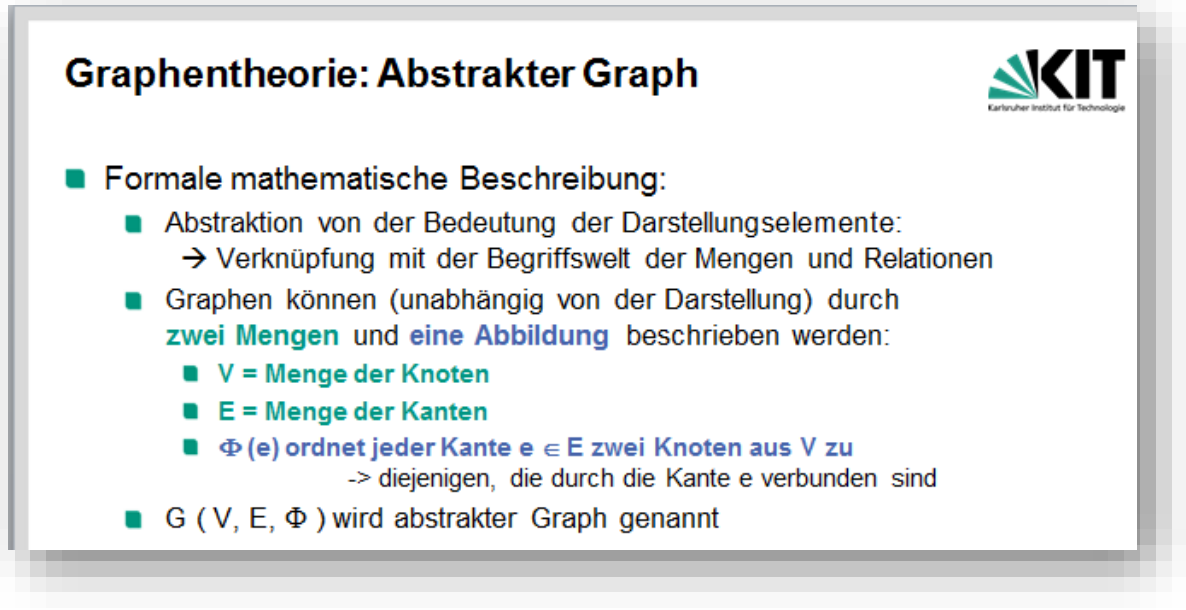
Alternative Suchmethoden

München: Kürzester Weg vom Marienplatz zur Frauenkirche



Breitensuche

- Breitensuche (Breadth First Search – BFS) ist ein grundlegender Algorithmus der Graphentheorie
- Resultate des Algorithmus:
  - Erzeugt einen Breitensuchbaum mit Wurzel s, der alle erreichbaren Knoten auf dem kürzesten Weg mit s verbindet
  - Findet alle von einem vordefinierten Startknoten s erreichbaren Knoten
- Eigenschaften:
  - Hat einen Laufzeitaufwand von O(V+E)
  - Funktioniert für gerichtete und ungerichtete Graphen



s. IT1

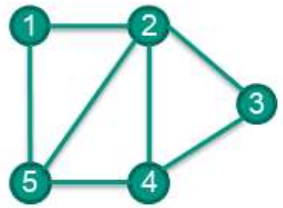
Graphentheorie: Adjazent
(Knoten orientiert)

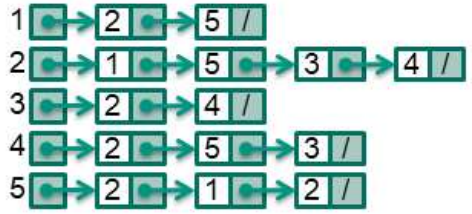
- Zwei durch eine Kante e verbundene Knoten g und h heißen adjazent zur Kante e
- **Adjazenzmatrix (Nachbarschaftsmatrix)**
  - speichert, welche Knoten des Graphen durch eine Kante verbunden sind.
  - besitzt für jeden Knoten eine Zeile und eine Spalte, woraus sich für *n* Knoten eine n x n-Matrix ergibt.
- Gerichteter Graph
  - Gibt es eine Kante von Knoten g zu Knoten h, wird in der Matrix in der g-ten Zeile an der h-ten Stelle eine 1 eingetragen.
- Ungerichteter Graph
  - Es muss eigentlich nur die Hälfte gespeichert werden, da sich die andere Hälfte durch Spiegelung ergibt.

s. IT1

Graphen: Darstellung Adjazenzmatrix
(Knoten orientiert)

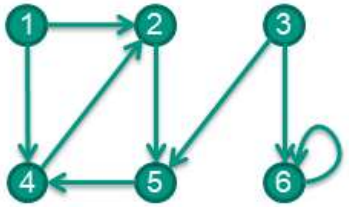
- Adjazenzdarstellung für einen ungerichteten Graphen:

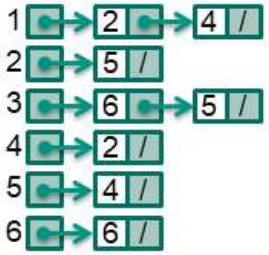




	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

- Adjazenzdarstellung für einen gerichteten Graphen:



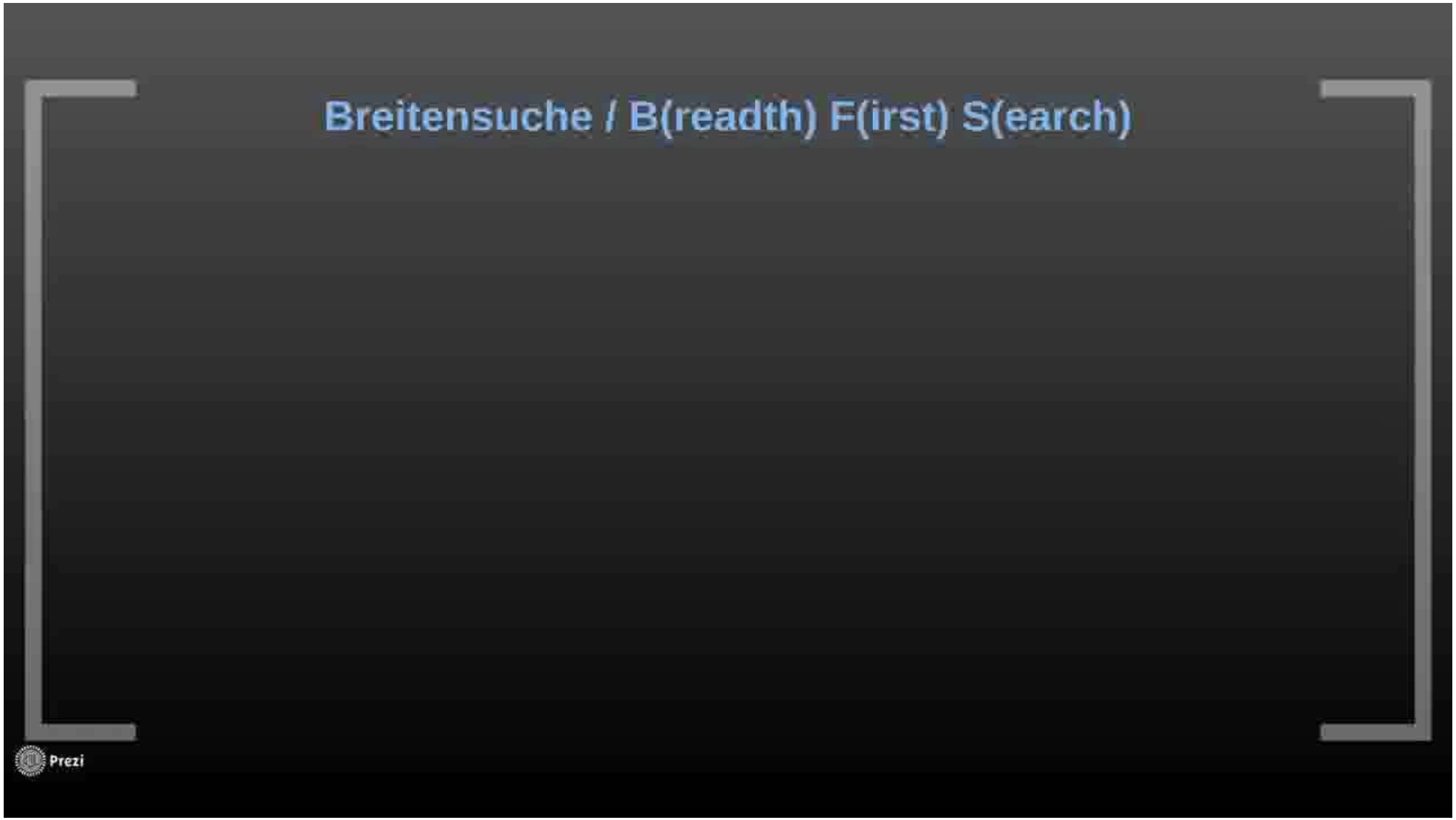


	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

9 | Informationstechnik
Kapitel 7: Datenstrukturen

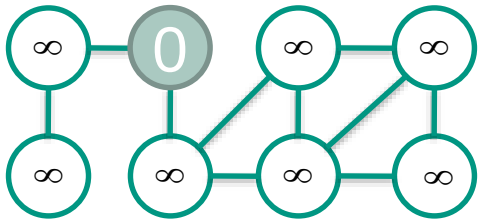
Institut für Technik der Informationsverarbeitung (ITIV)
Prof. Dr.-Ing. Eric Sax, © 2017

Breitensuche



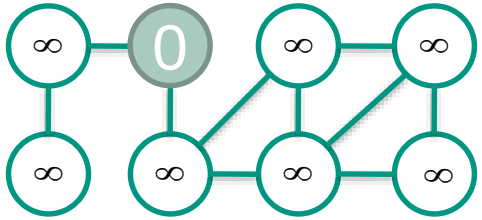
Ablauf: Breitensuche

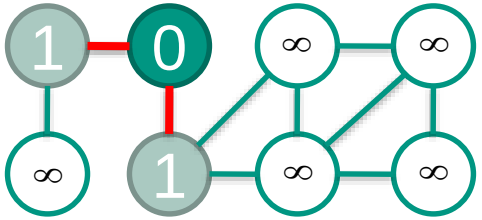
1.

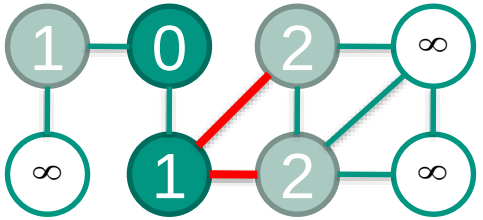


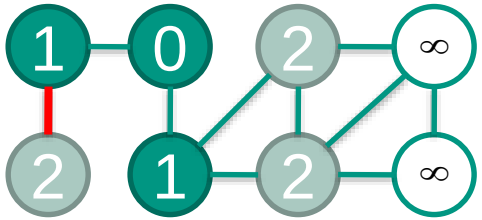
Am Anfang sind alle Knoten weiß, entdeckte Knoten werden grau, schon abgearbeitete Knoten dunkelgrün.

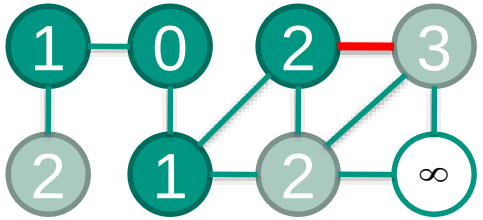
Ablauf: Breitensuche

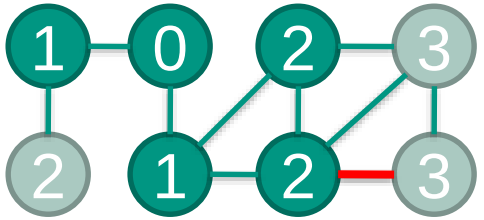
1.


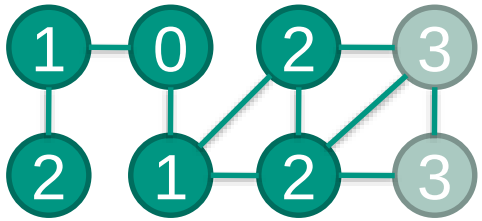
2.


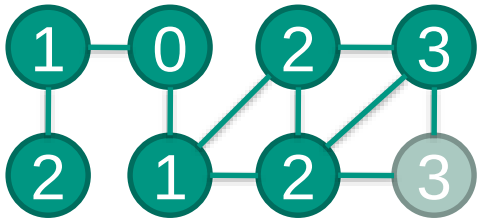
3.


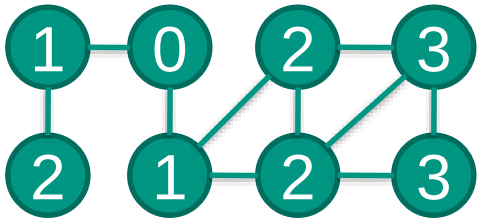
4.


5.


6.


7.


8.


9.


Am Anfang sind alle Knoten weiß, entdeckte Knoten werden grau, schon abgearbeitete Knoten dunkelgrün.

Breitensuche: Pseudocode

```
BreadthFirstSearch( G, s )
  for alle Knoten u in G                  //
  do farbe[u] = weiss                     //
     d[u] = ∞                             //
     vater[u] = NIL                       //

  create queue Q                          //
  farbe[s] = grau                         //
  d[s] = 0                                //
  enqueue( Q, s )                         //

  while Q not empty                       //
  do u = dequeue( Q )                     //
    for alle Knoten v aus Adj[u]          //
    do if farbe[v] == weiss               //
       then farbe[v] = grau               //
            d[v] = d[u] + 1               //
            vater[v] = u                  //
            enqueue( Q, v )               //
    farbe[u] = dunkelgrün                 //
```

Am Anfang sind alle Knoten weiß, entdeckte Knoten werden grau, schon abgearbeitete Knoten dunkelgrün.

Breitensuche: Pseudocode

Zum Selbststudium

```
BreadthFirstSearch( G, s )
  for alle Knoten u in G              // Initialisiere alle Knoten im Graph
  do farbe[u] = weiss                 // Alle Knoten wurden noch nicht gefunden
     d[u] = ∞                         // Alle Knoten haben noch keinen Abstand
     vater[u] = NIL                   // Alle Knoten haben noch keinen Vater

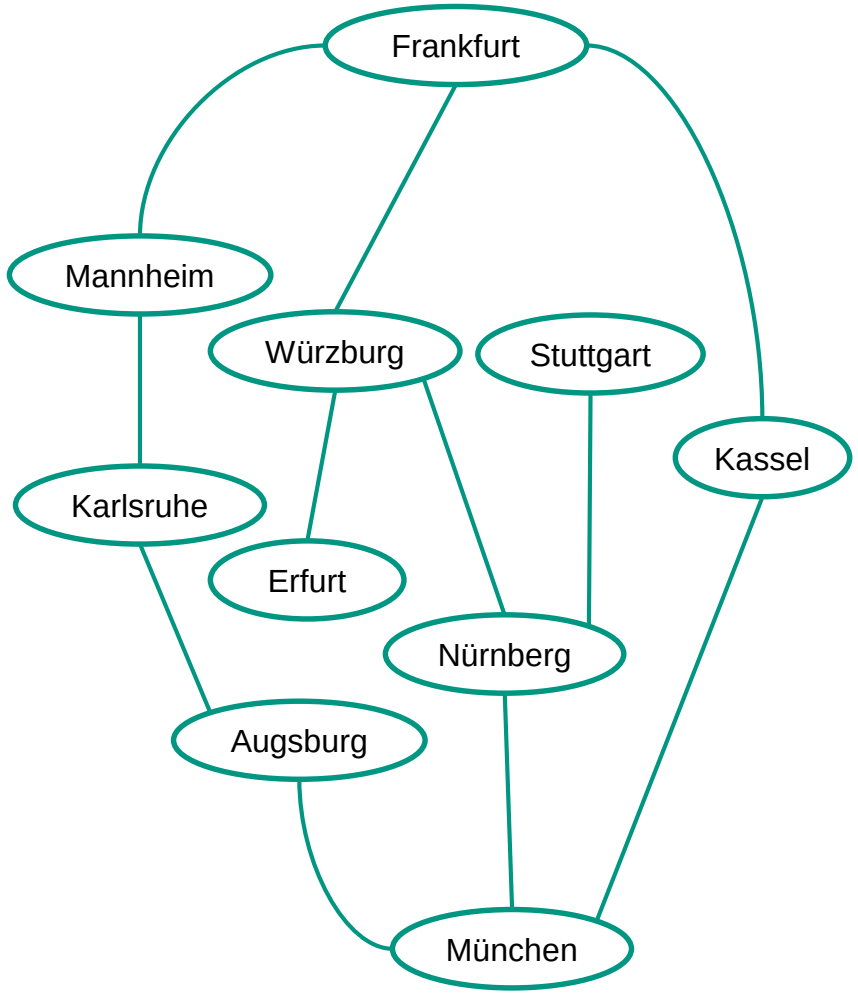
  create queue Q                      // Erzeuge eine Warteschlange Q
  farbe[s] = grau                     // Starte mit Startknoten s
  d[s] = 0                            // Abstand zum Startknoten d
  enqueue( Q, s )                     // Hänge s an die Warteschlange

  while Q not empty                   // Solange noch Knoten abgearbeitet werden
  do u = dequeue( Q )                 // hole den nächsten Knoten u
    for alle Knoten v aus Adj[u]      // Für alle zu u adjazente Knoten v
    do if farbe[v] == weiss           // prüfe ob v schon gefunden wurde
       then farbe[v] = grau           // wenn nicht, dann setze v auf gefunden
            d[v] = d[u] + 1           // Weise Knoten v seinen Abstand zu
            vater[v] = u              // Der Vater von Knoten v ist der Knoten u
            enqueue( Q, v )           // Merke v für die weitere Verarbeitung
    farbe[u] = dunkelgrün             // Markiere Knoten u als abgearbeitet
```

Am Anfang sind alle Knoten weiß, entdeckte Knoten werden grau, schon abgearbeitete Knoten dunkelgrün.

Beispiel: Breitensuche

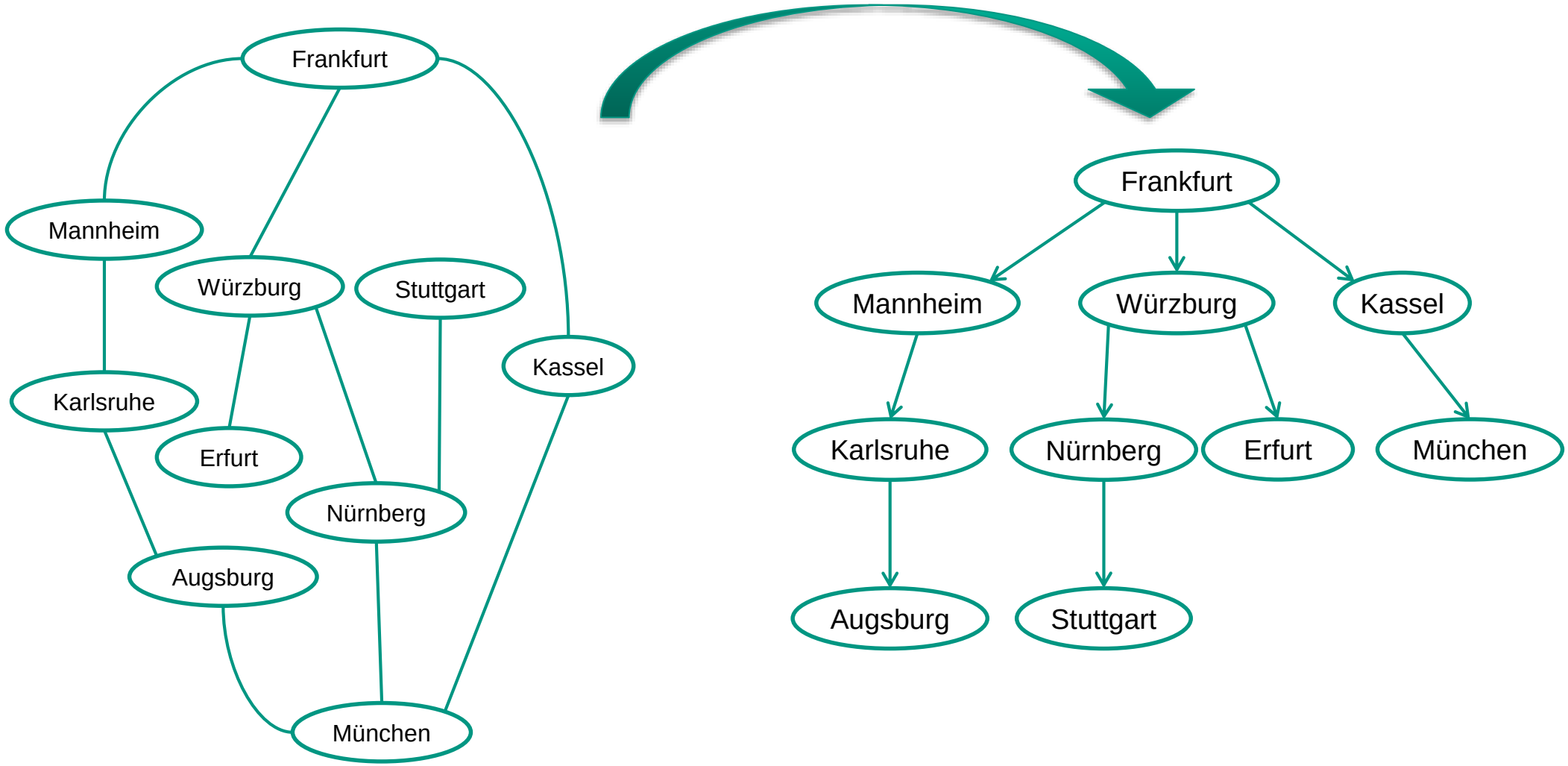
Start in Frankfurt



- Von Frankfurt aus werden die Nachbarn betrachtet:
  - Mannheim, Würzburg und Kassel.
- Von diesen aus werden auch jeweils die Nachbarn betrachtet:
  - Karlsruhe, Nürnberg, Erfurt, München.
- Gehen wir nun eine Stufe weiter, finden wir
  - als Nachbar von Karlsruhe Augsburg
  - als Nachbar von Nürnberg Stuttgart und München.
- Da München jedoch auf anderem Wege schon besucht wurde, wird es nicht mehr in den Graphen aufgenommen.
- Augsburg hat nun als einzigen Nachbarn noch das bereits besuchte München,
  - Stuttgart hat keinen Nachbarn außer Nürnberg, wo wir aber hergekommen sind.
- Die Breitensuche ist also komplett.
- Es entsteht ein Breitensuchbaum.

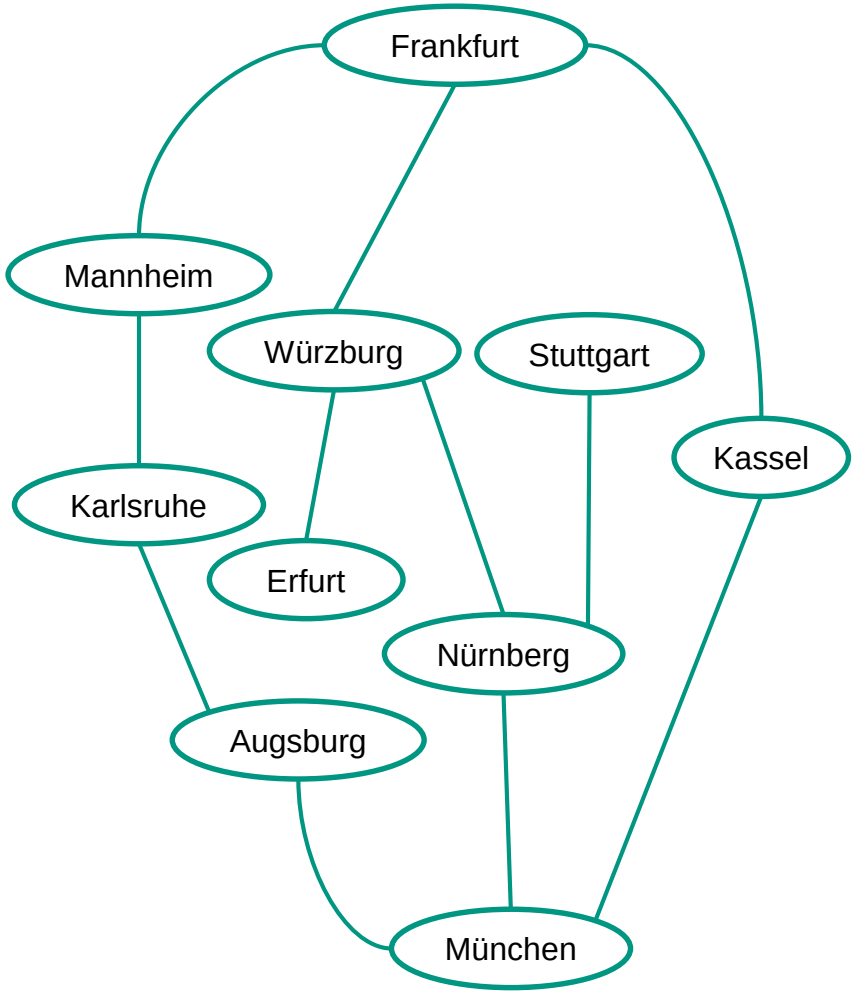
Beispiel: Breitensuche

Start in Frankfurt



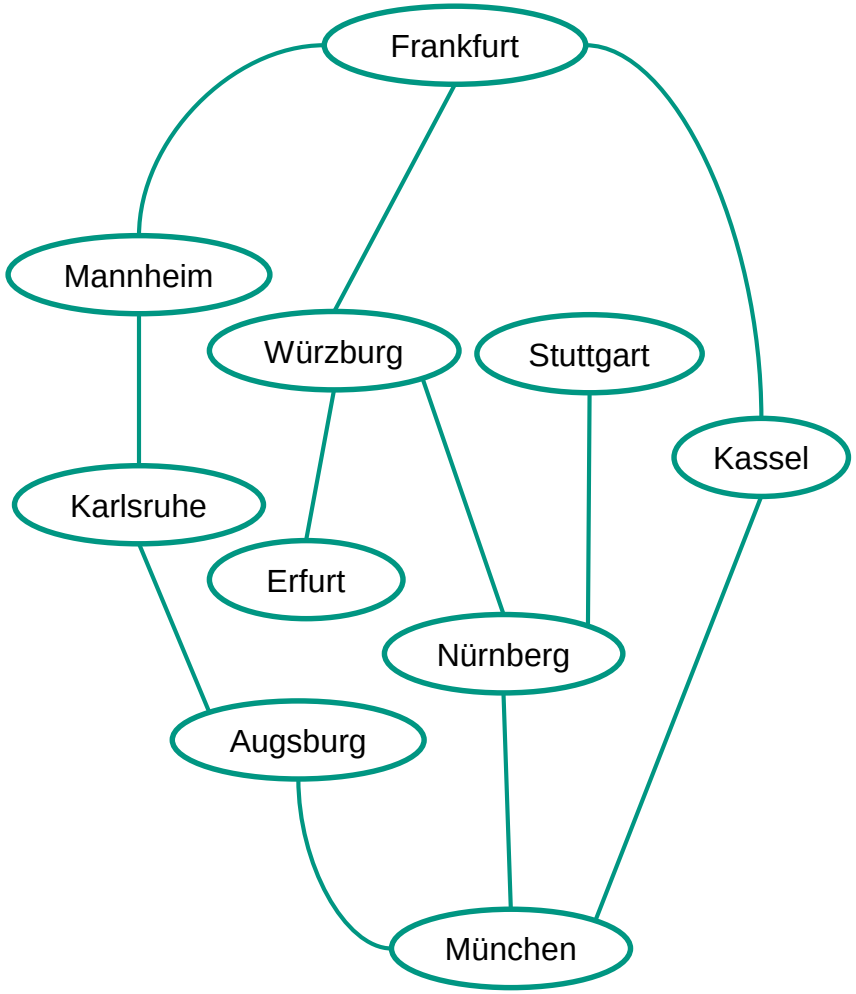
Übung: Breitensuche

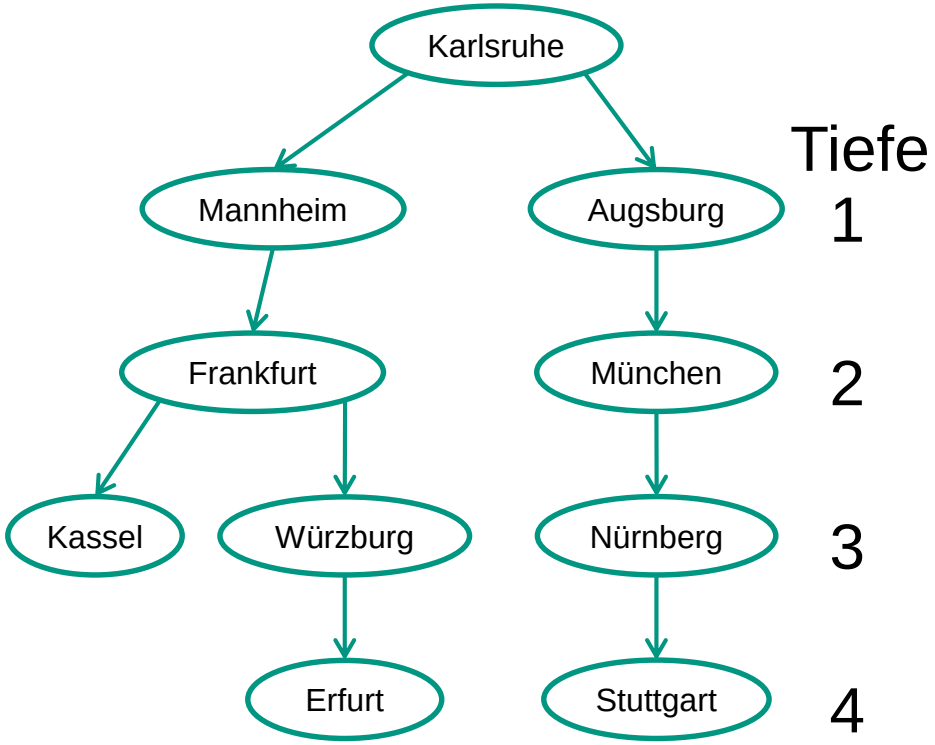
- Erzeugen Sie den Breitensuchbaum, wenn Sie in Karlsruhe starten.



Übung: Breitensuche

- Erzeugen Sie den Breitensuchbaum, wenn Sie in Karlsruhe starten.

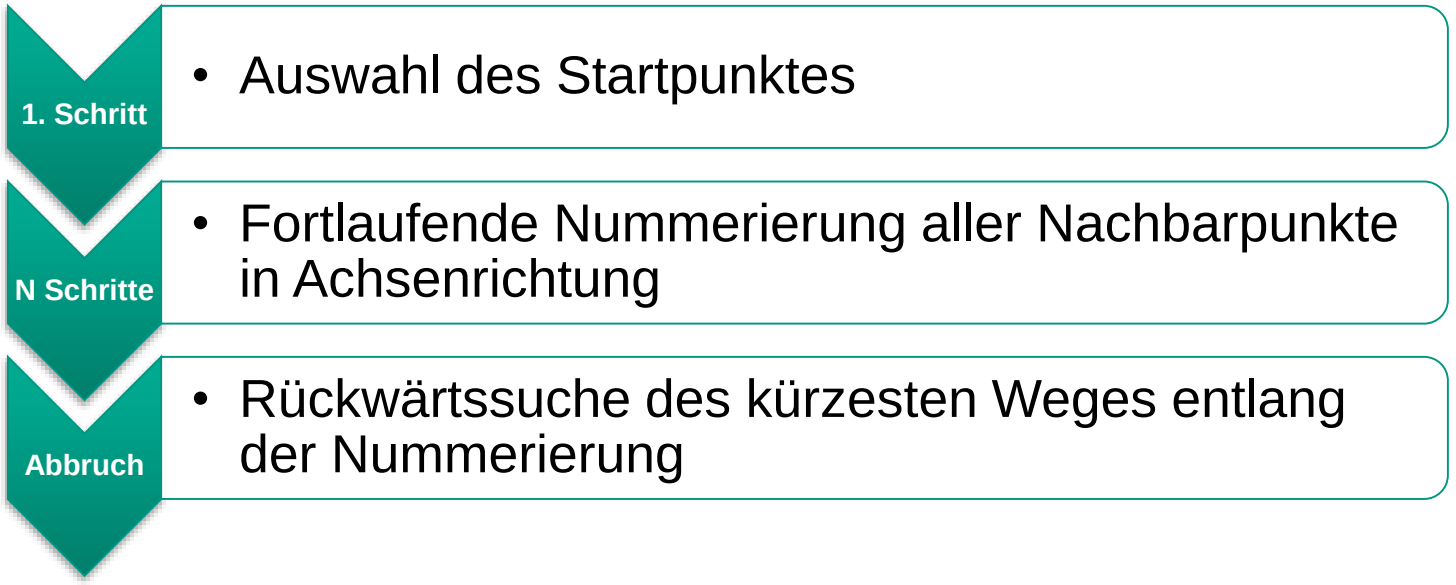




Beispiel Anwendung Breitensuche

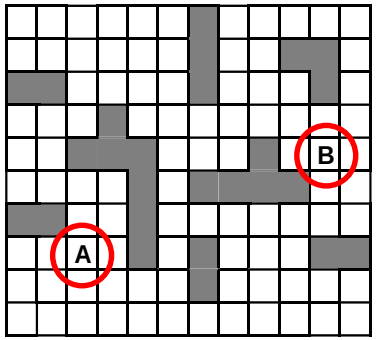
LEE-Algorithmus (maze routing)

- Der Lee-Algorithmus ist eine von mehreren Lösungen zur Breitensuche/Pathfinding
  - zum Finden eines Weges von einem Ausgangspunkt zu einem Zielpunkt.
- Zuerst wurde er bei der automatischen Erstellung von Platinen angewandt, kann aber auch bei Computerspielen zum Einsatz kommen, um die Figuren innerhalb der Spielwelt zu bewegen.

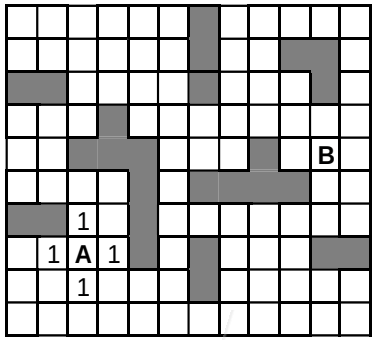


Beispiel für LEE-Algorithmus

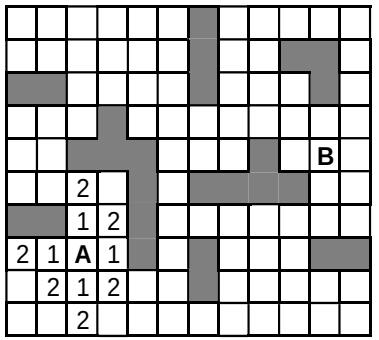
- Start in A → Gesucht wird B



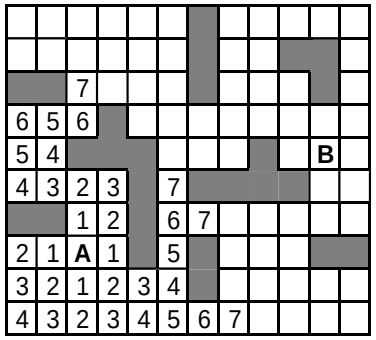
Verbindungspunkte



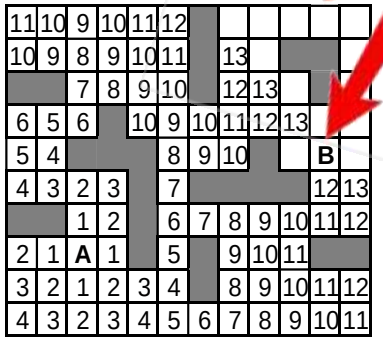
1. Welle



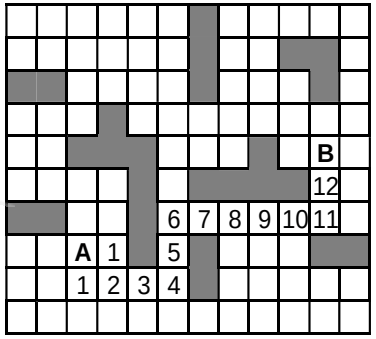
2. Welle



3. bis 7. Welle

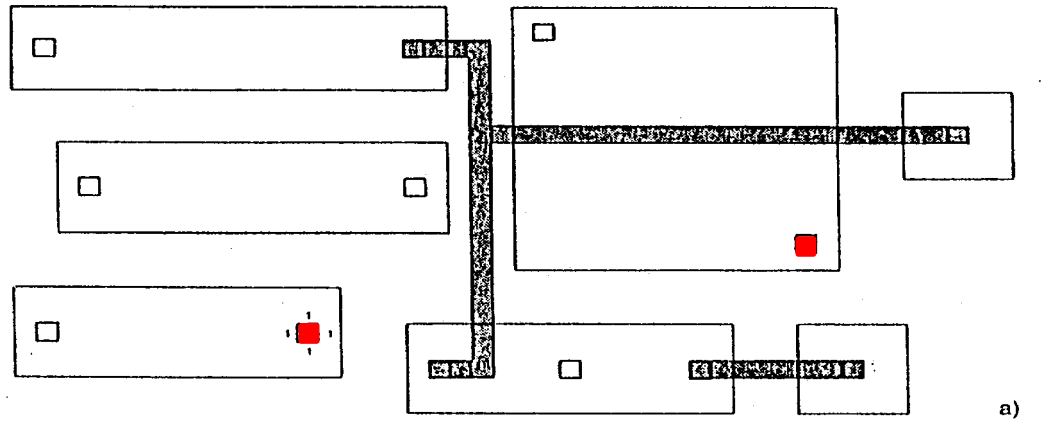


8. bis 13. Welle

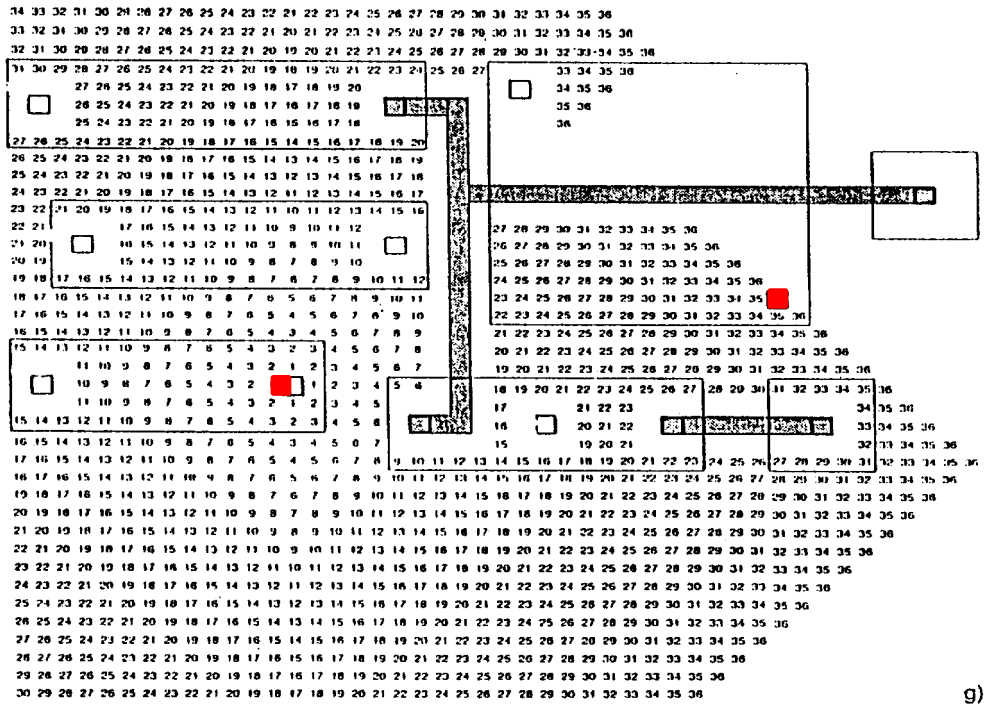


Rückverfolgung

Beispiel: LEE-Algorithmus

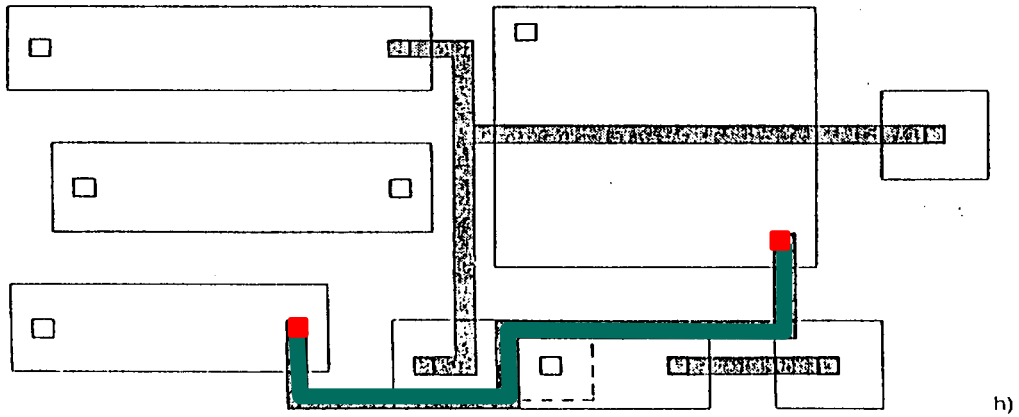


Beispiel: LEE-Algorithmus



g)

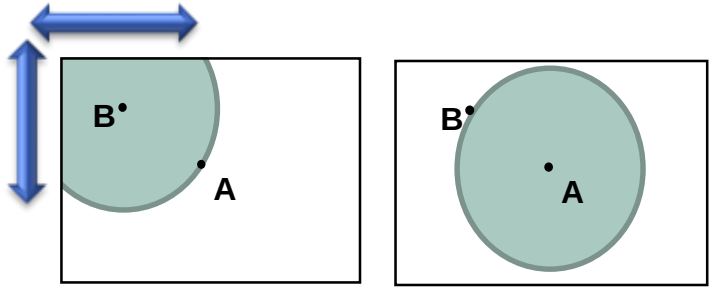
Beispiel: LEE-Algorithmus



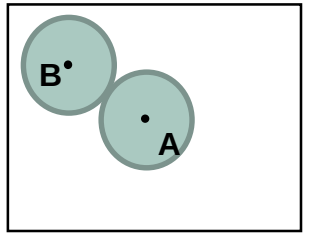
35. Welle → resultierende Verbindung (hier kürzeste Leiterbahn)

Modifikation des LEE-Algorithmus

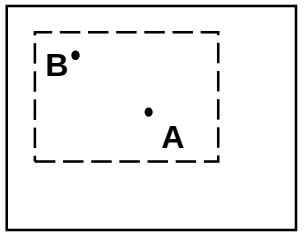
- Günstige Wahl des Startpunktes:



- Gleichzeitige Wegsuche von Start- und Zielpunkt:



- Einführung eines Suchrahmens:



- Übergang von Zweipunkt- zu Mehrpunktverbindungen
- Dreidimensionale Wegesuche möglich (Mehrlagenverbindungen)

- Breitensuche



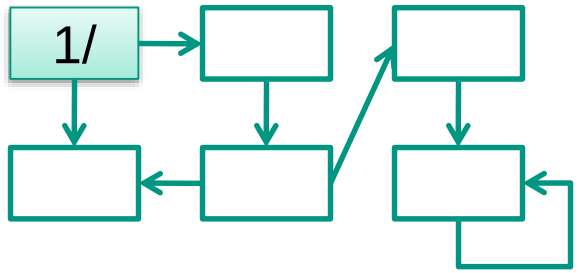
Tiefensuche

- Tiefensuche (*Depth First Search – DFS*) ist der nächste grundlegende Algorithmus der Graphentheorie.
- Im Gegensatz zur Breitensuche wird bei der Tiefensuche zunächst ein Pfad vollständig in die Tiefe beschritten, bevor abzweigende Pfade untersucht werden.
- Resultate des Algorithmus:
  - Findet alle von Startknoten s erreichbaren Knoten
  - Bearbeitet immer den zuletzt gefundenen Knoten
  - Erst nach Abarbeitung aller Nachbar-Knoten kehrt die Suche zum Vaterknoten zurück, so dass vor allem der Entdeckungszeitpunkt interessant ist.
  - Tiefensuchwald:
    - Zusammengesetzt aus mehreren Bäumen, je nach Zusammenhang des Graphen.
- Eigenschaften:
  - Hat einen Laufzeitaufwand von O(V+E)
  - Funktioniert für gerichtete und ungerichtete Graphen


Tiefensuche / D(epth) F(irst) S(earch)
Prezi

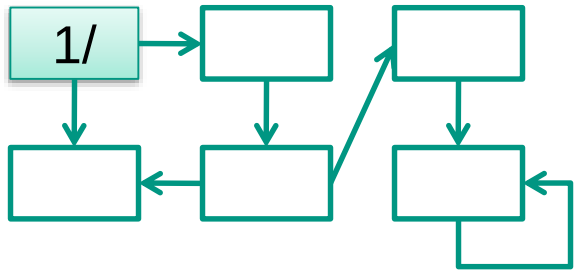
Ablauf: Tiefensuche

1.

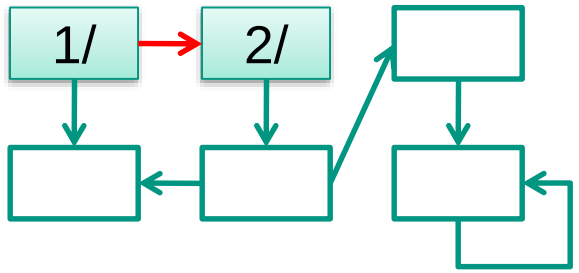


Ablauf: Tiefensuche

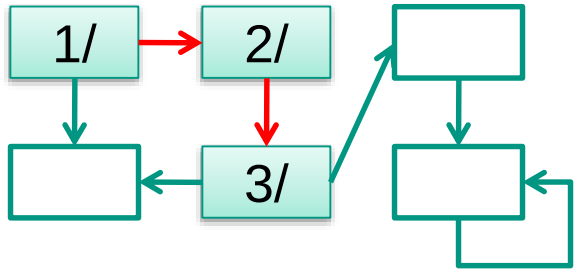
1.



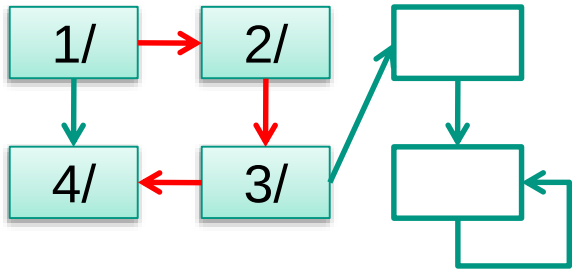
2.



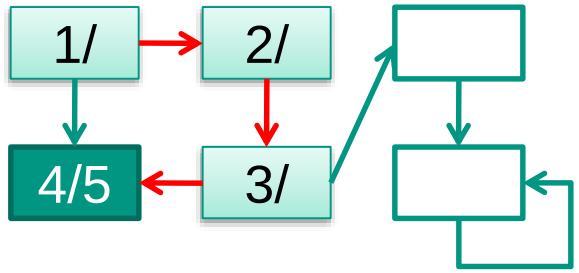
3.



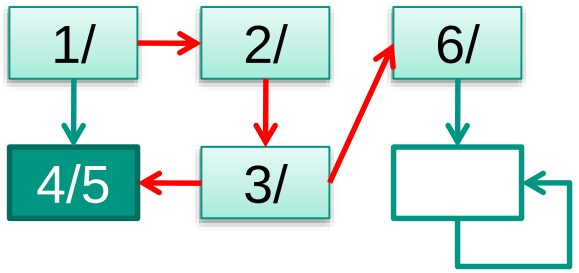
4.



5.

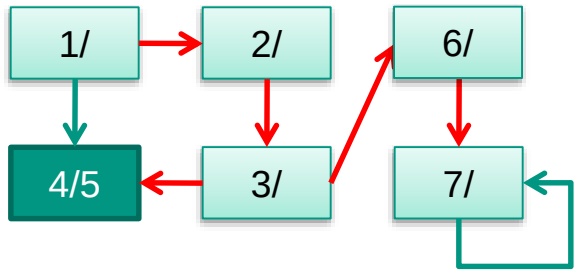


6.



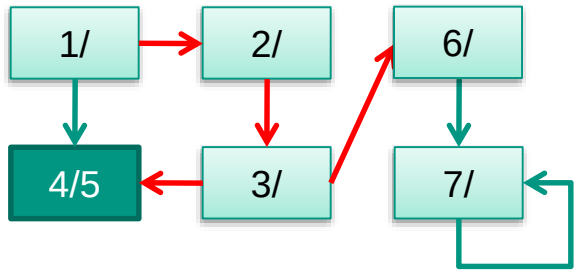
Ablauf: Tiefensuche

7.

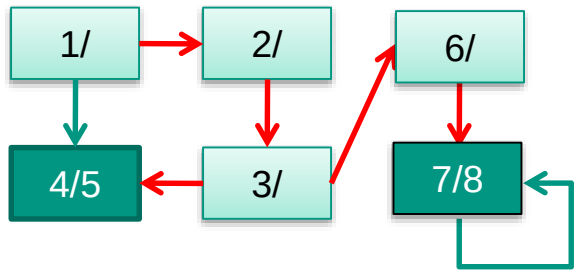


Ablauf: Tiefensuche

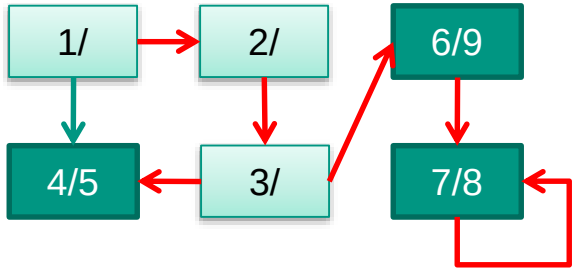
7.



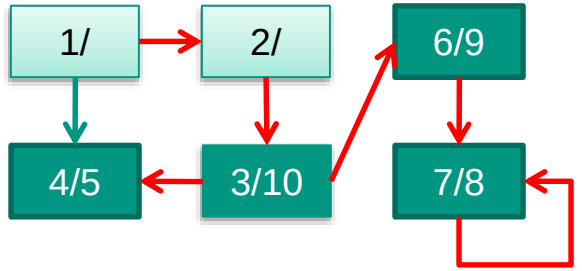
8.



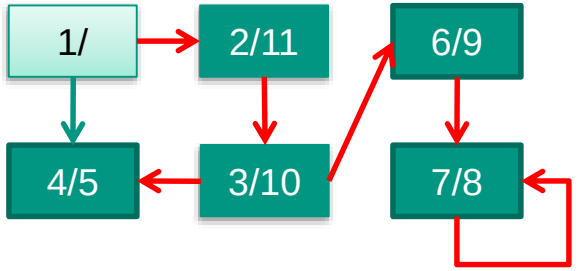
9.



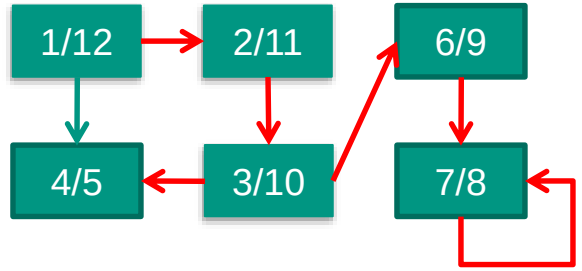
10.



11.



12.



Tiefensuche: Pseudocode

```
DepthFirstSearch( G )
  for alle Knoten u in G                //
  do farbe[u] = weiss                   //
     vater[u] = NIL                     //

  zeit = 0                              //
  for alle Knoten u in G                //
  do if farbe[u] == weiss               //
     then DFS-VISIT( u )                //

DFS-VISIT( u )
  farbe[u] = grau; zeit = zeit + 1      //
  startTime[u] = zeit                   //
  for alle Knoten v aus Adj[u]          //
  do if farbe[v] == weiss               //
     then vater[v] = u                  //
          DFS-VISIT( v )                //

  farbe[u] = schwarz; zeit = zeit + 1//
  endTime[u] = zeit                     //
```

Tiefensuche: Pseudocode

Zum Selbststudium

```
DepthFirstSearch( G )
  for alle Knoten u in G              // Initialisiere alle Knoten im Graph
  do farbe[u] = weiss                 // Alle Knoten wurden noch nicht gefunden
     vater[u] = NIL                   // Alle Knoten haben noch keinen Vater

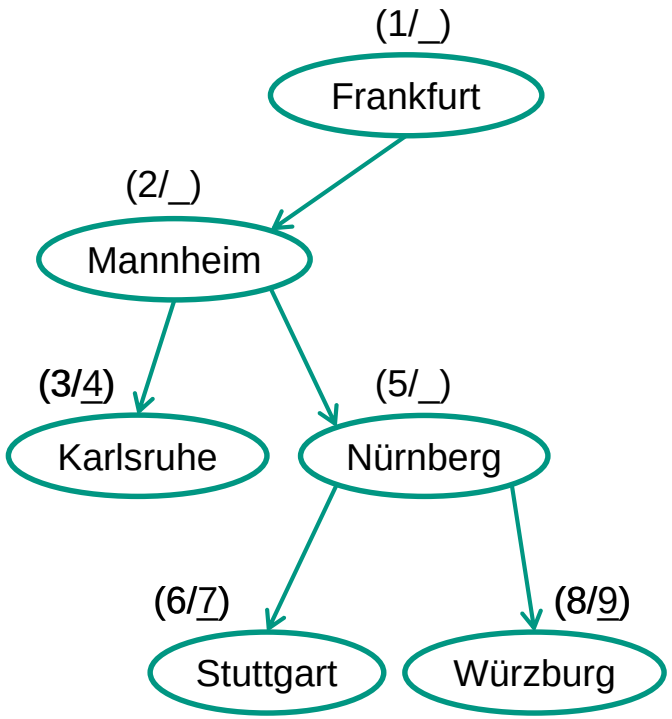
  zeit = 0                            // Zum Messen der Zeitschritte
  for alle Knoten u in G              // Für alle Knoten im Graph,
  do if farbe[u] == weiss             // welche noch nicht bearbeitet wurden
     then DFS-VISIT( u )              // Besuch registrieren

DFS-VISIT( u )
  farbe[u] = grau; zeit = zeit + 1    // Als besucht setzen; Zeitschritt erhöhen
  startTime[u] = zeit                 // Startzeitpunkt des Knotens u abspeichern
  for alle Knoten v aus Adj[u]        // Für alle zu u adjunken Knoten v
  do if farbe[v] == weiss             // prüfe ob v schon gefunden wurde
     then vater[v] = u                // Der Vater vom Knoten v ist der Knoten u
          DFS-VISIT( v )              // Weiter in die Tiefe vom Knoten v gehen

  farbe[u] = schwarz; zeit = zeit + 1// Knoten u ist abgearbeitet; Zeitschritt erhöhen
  endTime[u] = zeit                   // Endzeitpunkt des Knotens u abspeichern
```

Übung: Tiefensuche

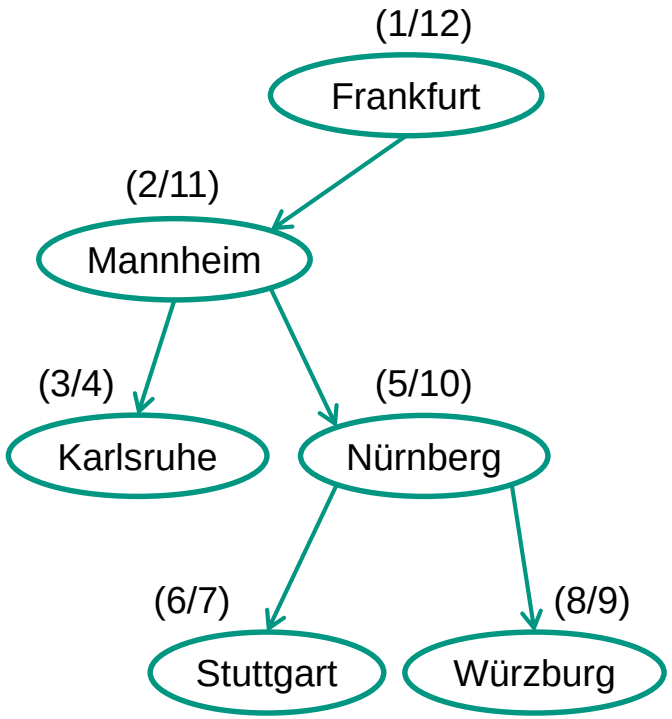
- Finden Sie mit Hilfe der Tiefensuche einen Weg von Frankfurt nach Würzburg.



1
2
3
4
5
6
7
8
9
10
11
12

Übung: Tiefensuche

- Finden Sie mit Hilfe der Tiefensuche einen Weg von Frankfurt nach Würzburg.



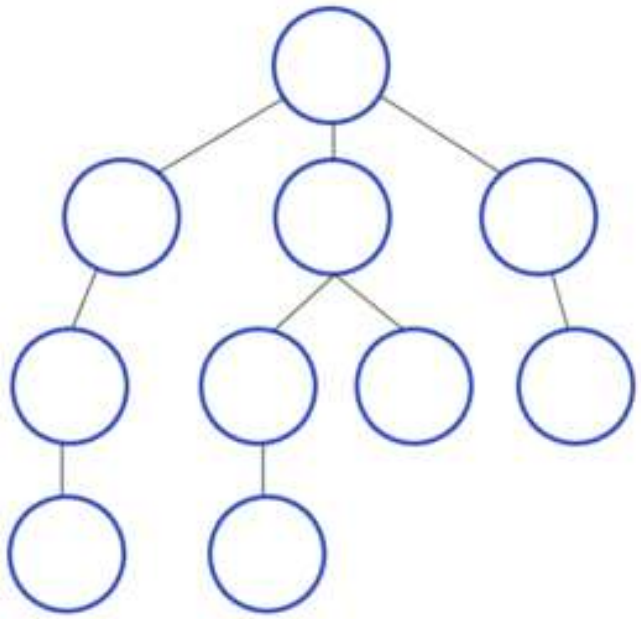

```
1   DFS(Frankfurt)
      // Frankfurt in Bearbeitung
2     DFS(Mannheim)
        // Mannheim in Bearbeitung
3       DFS(Karlsruhe)
          // Karlsruhe in Bearbeitung
4         Karlsruhe abgeschlossen
5       DFS(Nürnberg)
          // Nürnberg in Bearbeitung
6         DFS(Stuttgart)
            // Stuttgart in Bearbeitung
7           Stuttgart abgeschlossen
8         DFS(Würzburg)
            // Würzburg in Bearbeitung
                // Pfad gefunden
9           Würzburg abgeschlossen
10      Nürnberg abgeschlossen
11    Mannheim abgeschlossen
12  Frankfurt abgeschlossen
```

- Tiefensuche

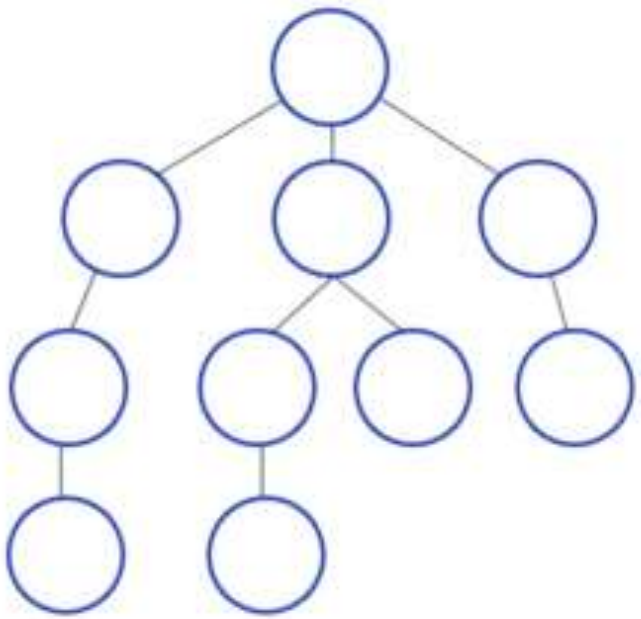


Vergleich

- Breitensuche
  - Bevorzugt bei Knotensuche
  - Bei mehreren möglichen Lösungen, schneller alternative, kostengünstige Lösung

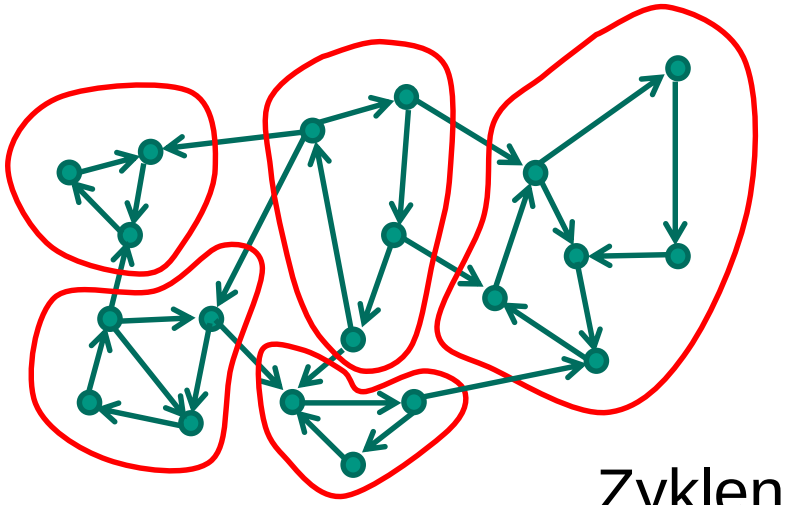


- Tiefensuche
  - Bevorzugt bei Blattsuche
  - Tiefenbeschränkung möglich
  - Kosten eskalieren bei monoton steigenden Pfad-Suchkosten



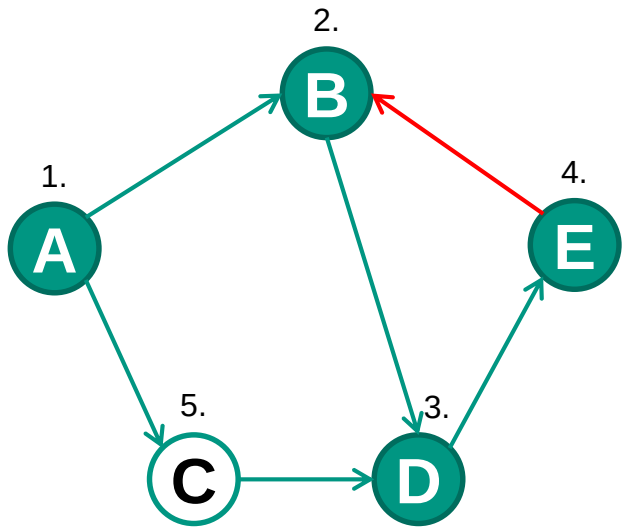
Zyklensuche

- Anwendung der Tiefensuche zum Finden von Zyklen in Graphen



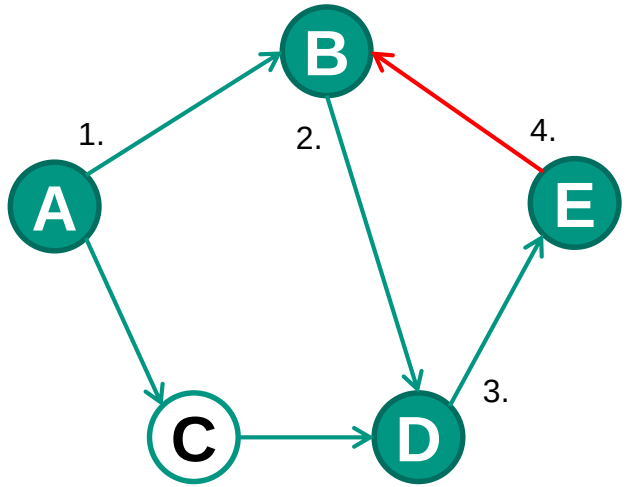
Zyklensuche

- Anwendung der Tiefensuche zum Finden von Zyklen in Graphen
- Nutzung der verschiedenen Kantenarten bei der DFS:
  - Vorwärtskanten, z.B.: von A nach C
  - Querkanten, z.B.: von C nach D
  - Rückwärtskanten, z.B.: von E nach B



Zyklensuche

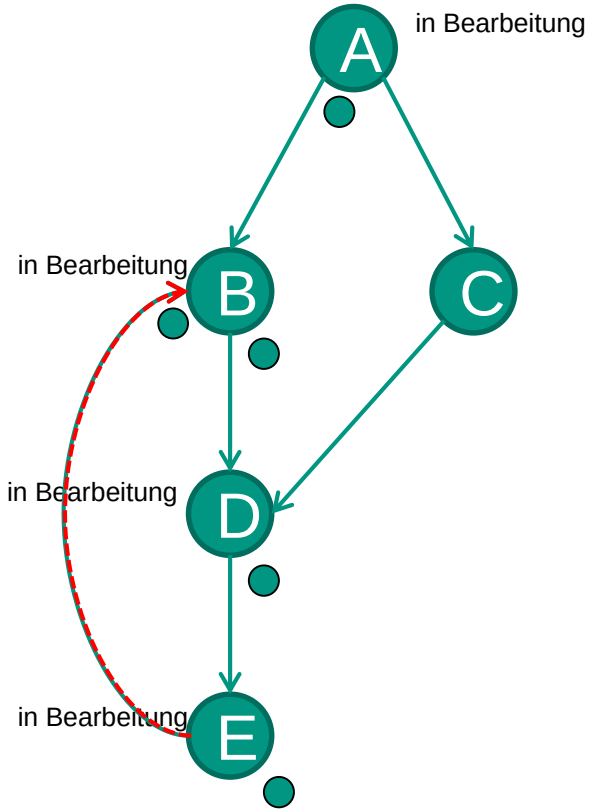
- Anwendung der Tiefensuche zum Finden von Zyklen in Graphen
- Nutzung der verschiedenen Kantenarten bei der DFS:
  - Vorwärtskanten, z.B.: von A nach C
  - Querkanten, z.B.: von C nach D
  - Rückwärtskanten, z.B.: von E nach B
- Nur Rückwärtskanten können Zyklen verursachen
- Vorgehen zur Zyklensuche:
  - Führe Tiefensuche durch
  - Sobald die Tiefensuche auf eine Rückwärtskante stößt, die auf einen Knoten mit Status „in Bearbeitung“ verweist, ist ein Zyklus gefunden



Im 4. Schritt stößt die DFS von E auf B
Ein Zyklus ist gefunden!

Beispiel: Zyklensuche

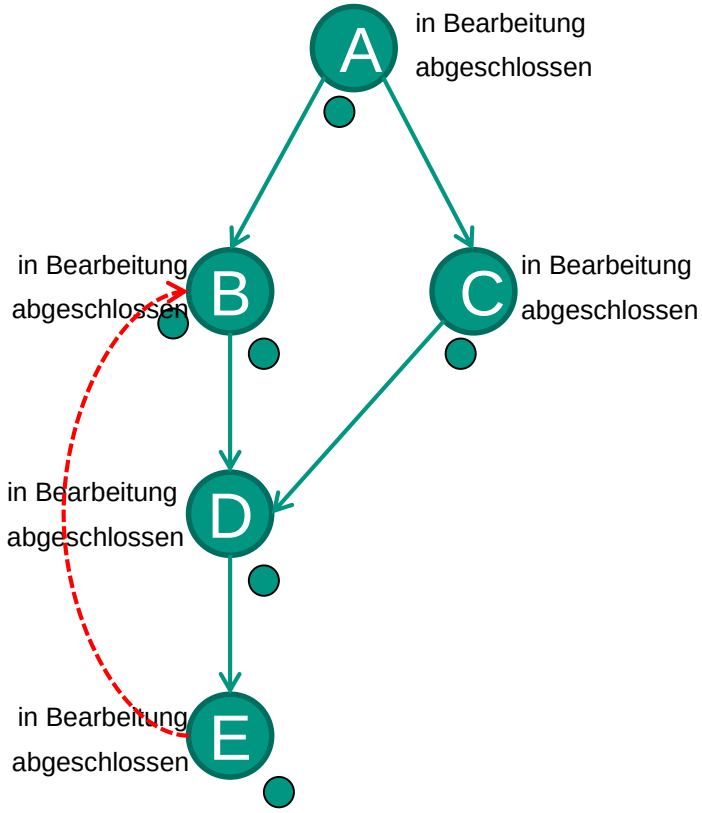
```
Zyklensuche (A) // A noch nicht begonnen
  A in Bearbeitung
  Zyklensuche (B) // B noch nicht begonnen
    B in Bearbeitung
    Zyklensuche (D) // D noch nicht begonnen
      D in Bearbeitung
      Zyklensuche (E) // E noch nicht begonnen
        E in Bearbeitung
        Zyklensuche (B) // B in Bearbeitung
                Zyklus gefunden!
```



Beispiel: Zyklensuche

```
Zyklensuche (A) // A noch nicht begonnen
  A in Bearbeitung
  Zyklensuche (B) // B noch nicht begonnen
    B in Bearbeitung
    Zyklensuche (D) // D noch nicht begonnen
      D in Bearbeitung
      Zyklensuche (E) // E noch nicht begonnen
        E in Bearbeitung
        Zyklensuche (B) // B in Bearbeitung
          Zyklus gefunden!
        E abgeschlossen
      D abgeschlossen
    B abgeschlossen
  Zyklensuche (C) // C noch nicht begonnen
    C in Bearbeitung
    Zyklensuche (D) // D abgeschlossen
    C abgeschlossen
  A abgeschlossen
```

Fertig!



- Zyklensuche

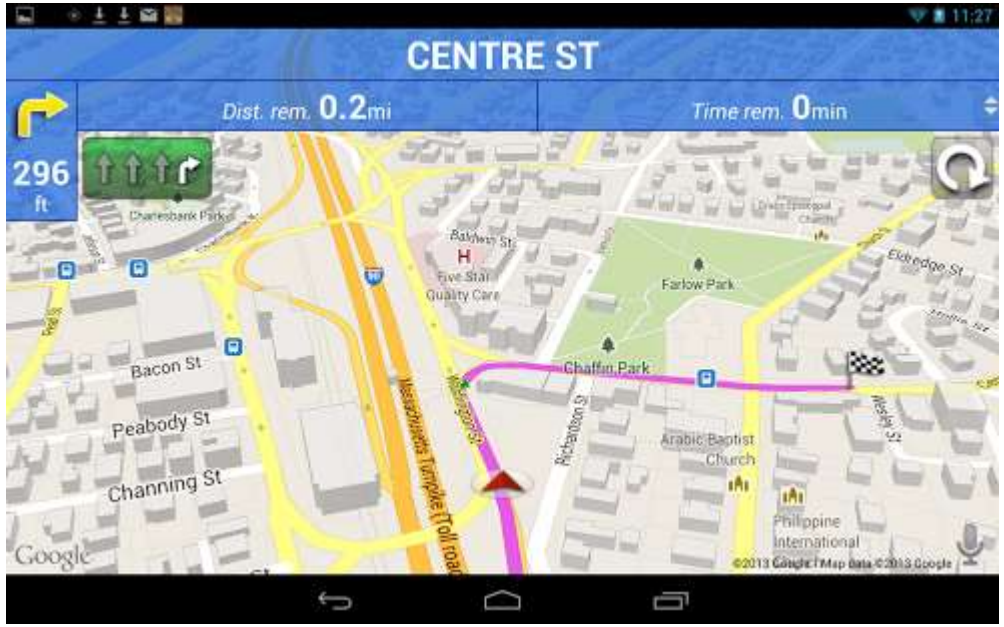


Zyklen





Kürzeste Pfade



- Finden eines kürzesten Pfades in einem Graphen ist ein sehr altes, sehr bekanntes Problem → Distanzberechnung
- Praktische Anwendung in heutigen Navigationssystemen
- Löst das Problem des kürzesten Pfades bei einem Startknoten in einem positiv gewichteten, gerichteten Graphen G(V, E)

Dijkstra-Algorithmus

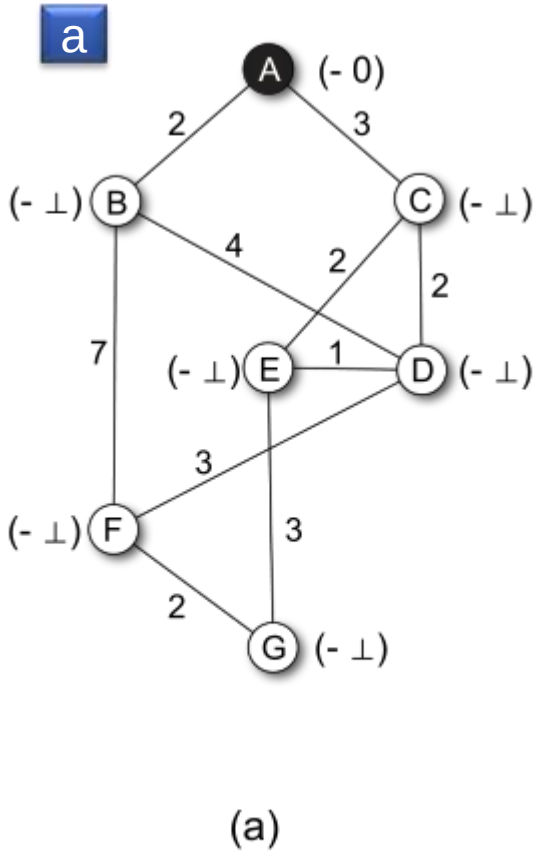
- Die Grundidee des Algorithmus ist es, immer derjenigen Kante zu folgen, die den kürzesten Streckenabschnitt vom Startknoten darstellt.
- Es wird die Distanz eines Knotens zu einem Zielknoten berechnet (*single-pair, shortest path*)
  oder die Distanzen aller Knoten zum Startknoten (*single-source, shortest path*).
- Es werden sowohl die kürzesten Wegstrecken als auch deren Knotenfolgen berechnet.



- **Edsger Wybe Dijkstra** (* 11. Mai 1930 in Rotterdam; † 6. August 2002 in Nuenen, Niederlande) war ein niederländischer Informatiker.
- Er war der Wegbereiter der strukturierten Programmierung

Beispiel zu Dijkstra-Routingverfahren

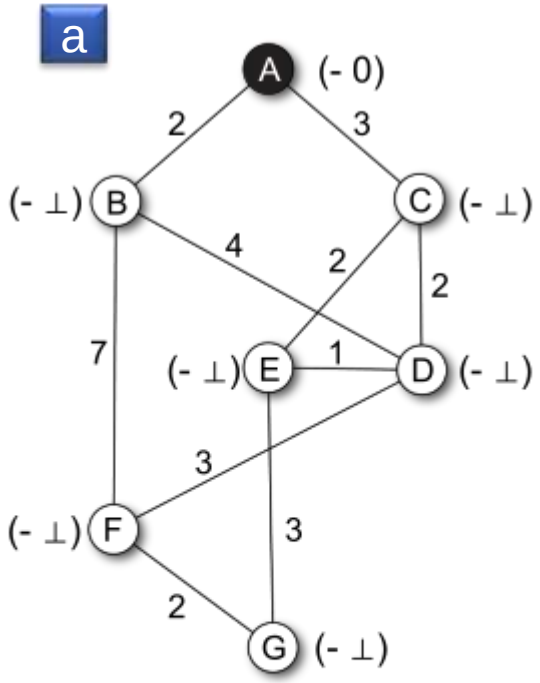
Vorgehen



(a)

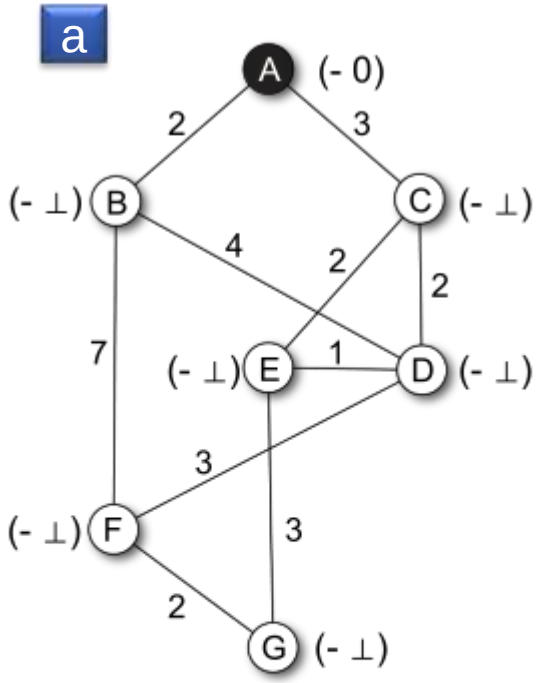
- Initialisiere die Distanz im Startknoten mit 0 und in allen anderen Knoten mit ∞ (oder hier ⊥).
- Solange es noch unbesuchte Knoten gibt, wähle darunter denjenigen mit minimaler Distanz aus und speichere, dass dieser Knoten schon besucht wurde.
- Berechne für alle noch unbesuchten Nachbarknoten die Summe aus dem jeweiligen Kantengewicht und der Distanz vom Startknoten zum aktuellen Knoten.
- Ist dieser Wert für einen Knoten kleiner als die dort gespeicherte Distanz, aktualisiere sie und setze den aktuellen Knoten als Vorgänger.
  - Dieser Schritt wird auch als Update oder Relaxieren bezeichnet.
- Hier: Netzwerk mit Knoten A, B, C, D, E, F, G und den jeweiligen Verbindungsgewichten

Beispiel zu Dijkstra-Routingverfahren



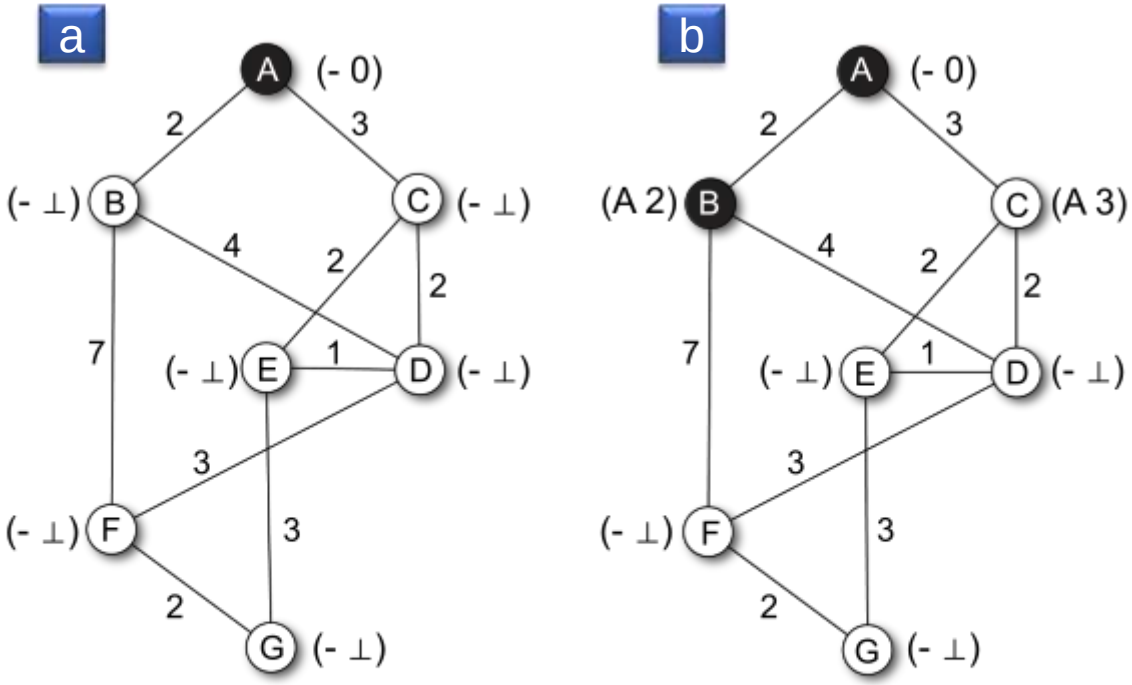
- Es wird ein **kürzester Weg** von A nach G gesucht (single pair, shortest path)
- Zu jedem Knoten wird dazu der jeweilige Vorgängerknoten entlang des kürzesten Weges und die aktuell ermittelte Entfernung dieses Knotens vom Startknoten bereitgestellt.
- Zu Beginn ist für jeden Knoten der Vorgängerknoten unbesetzt und die aktuelle Entfernung wird mit Unendlich angegeben.

Beispiel zu Dijkstra-Routingverfahren



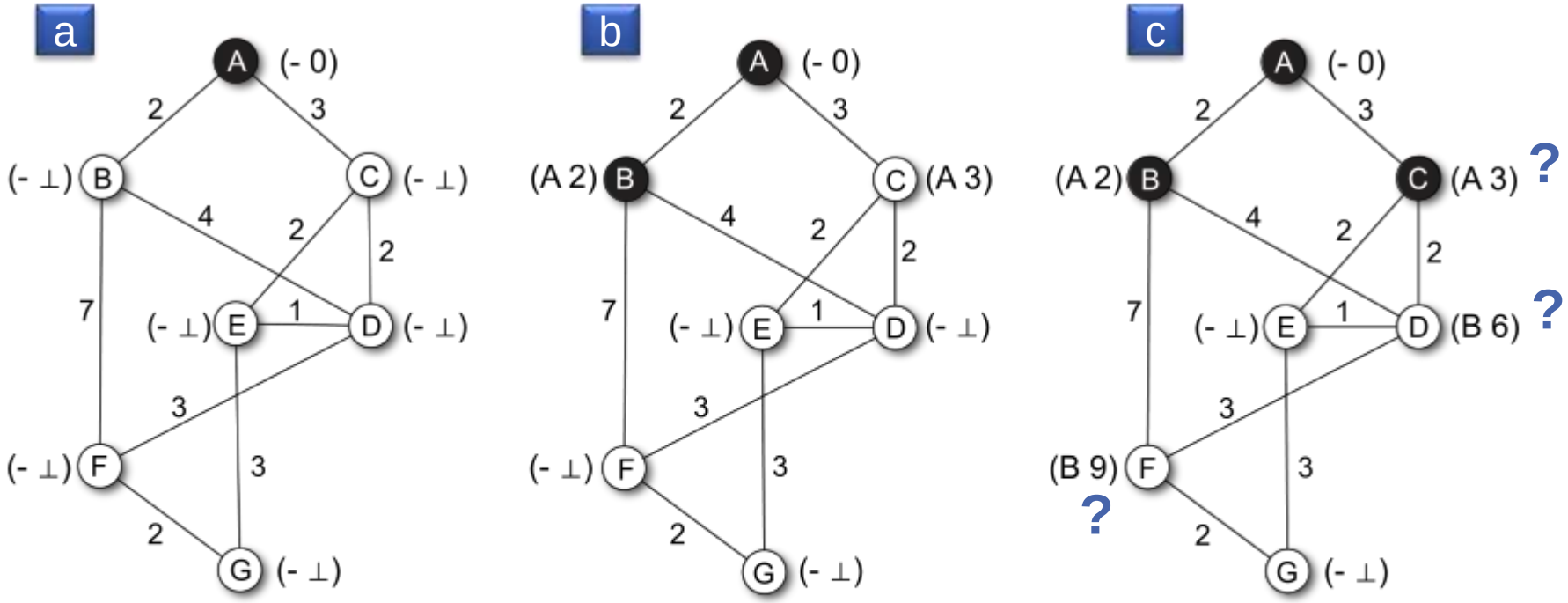
- Wir unterscheiden „**permanente**“ bzw. „besuchte“ Knoten von „noch in der Berechnung“ befindlichen Knoten.
  - Def: Für **permanente** Knoten wurde bereits die kürzeste Entfernung zum Startknoten bestimmt (unveränderlich)
  - Sie können neuer Ausgangspunkt für Folgeberechnungen werden.
  - **Permanente** Knoten sind durch ausgefüllte Kreise kenntlich gemacht.

Beispiel zu Dijkstra-Routingverfahren (2)



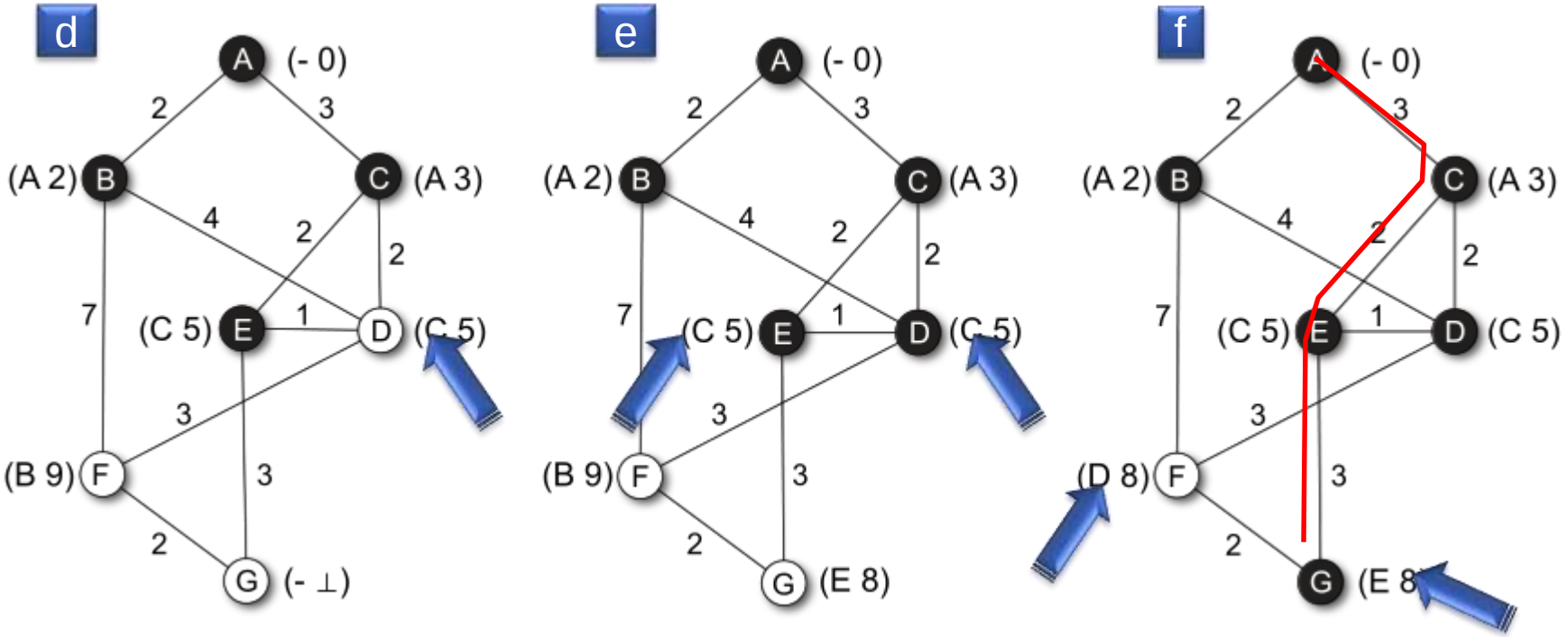
- Zu Beginn der Berechnung wird der Startknoten A als permanent markiert (a).
- Es werden dann die zu A benachbarten Knoten B und C betrachtet,
- Da diese noch nicht als permanent markiert sind, erhalten sie die Belegung (A,2) für B und (A,3) für C entsprechend der (gewichteten) Entfernung von A.
- Jetzt werden alle nicht permanenten Knoten des Graphs untersucht und derjenige Knoten mit der kleinsten Belegung, hier B, wird zum neuen Ausgangsknoten gewählt und als permanent markiert (b).

Beispiel zu Dijkstra-Routingverfahren (2)



- Dann werden die zu B benachbarten, nicht permanenten Knoten D und F betrachtet. Entsprechend ihrer Distanz von A, die sich ergibt aus der Distanz vom Ausgangsknoten B plus dessen Belegung, erhalten die Knoten die Belegungen (B,9) für F und (B,6) für D.
- Zur Ermittlung des nächsten Ausgangsknotens für den kürzesten Weg werden wieder alle nicht permanenten Knoten betrachtet und derjenige mit der kürzesten Distanz zu A, jetzt C, herausgenommen und als permanent markiert (c).
- Dann werden die nicht permanenten Nachbarn von C betrachtet (D und E) und deren Distanz von A entlang der Route über C ermittelt.

Beispiel zu Dijkstra-Routingverfahren (2)

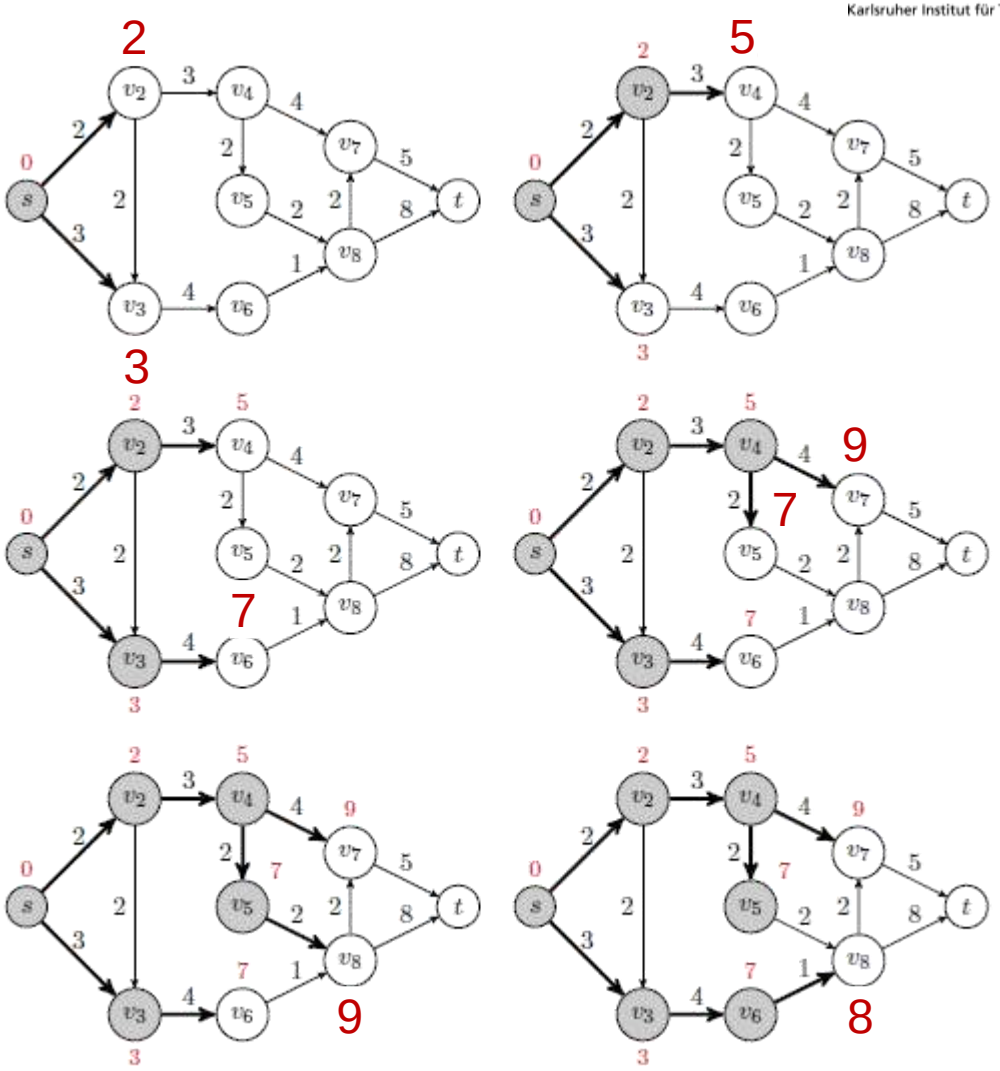


- Da die Route von A nach D über C kürzer ist, als die bisherige, wird die Belegung von D auf (C,5) gesetzt (d) → Relaxation!
- Haben, wie jetzt E und D, zwei oder mehrere nicht permanenten Knoten dieselbe Distanz zur Quelle, so wird **zufällig** einer ausgewählt, als permanent markiert und zum Ausgangspunkt für die nächste Runde gewählt.
- Auf diese Weise wird solange fortgefahren, bis der eigentliche Zielpunkt (G) als permanent markiert worden ist.
- Die Entfernung vom Startpunkt A ergibt sich dann aus der Belegung von G (8), und der kürzeste Weg kann in entgegengesetzter Richtung von G nach A, der Belegung der jeweiligen Knoten folgend, rekonstruiert werden → G,E,C,A

Beispiel

Zum Selbststudium

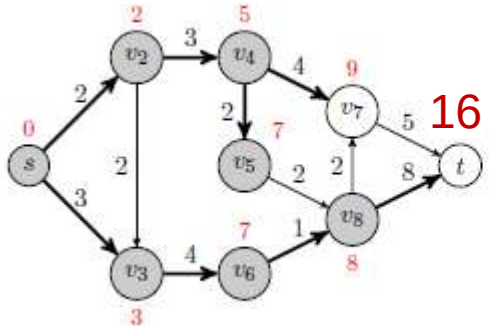
- Gesucht: kürzester s-t-Weg
- Wir schreiben hier die Distanz(v) jeweils außen an den Knoten und kennzeichnen die Menge aller bereits bearbeiteten Knoten durch Einfärben der Knoten.
- Außerdem kennzeichnen wir jeweils die Kante, von der aus ein Knoten sein aktuelles Label erhalten hat, um am Ende den kürzesten Weg rekonstruieren zu können.
- Es ergeben sich dann die folgenden Berechnungsschritte

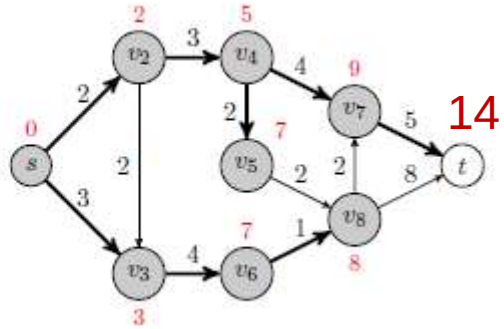


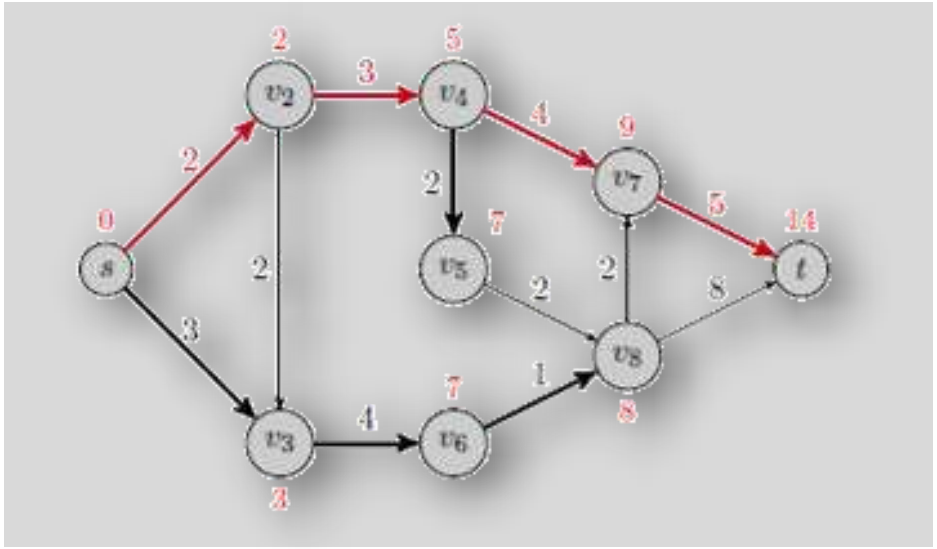
Beispiel

Zum Selbststudium

- Gesucht: kürzester s-t-Weg
- Wir schreiben hier die Distanz(v) jeweils außen an den Knoten und kennzeichnen die Menge aller bereits bearbeiteten Knoten durch Einfärben der Knoten.
- Außerdem kennzeichnen wir jeweils die Kante, von der aus ein Knoten sein aktuelles Label erhalten hat, um am Ende den kürzesten Weg rekonstruieren zu können.
- Es ergeben sich dann die folgenden Berechnungsschritte

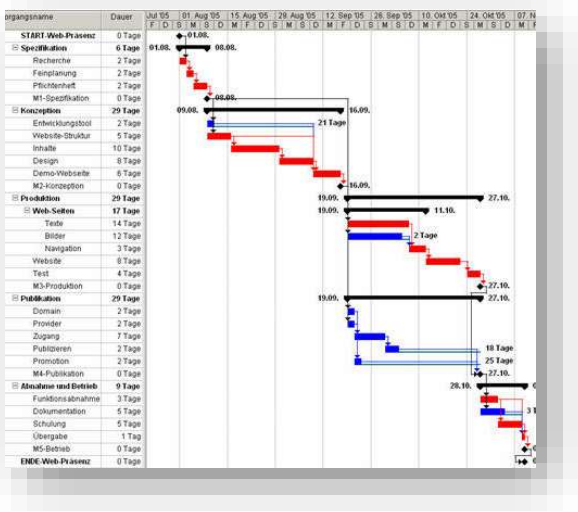






Kritische Pfad Methode (CPM)

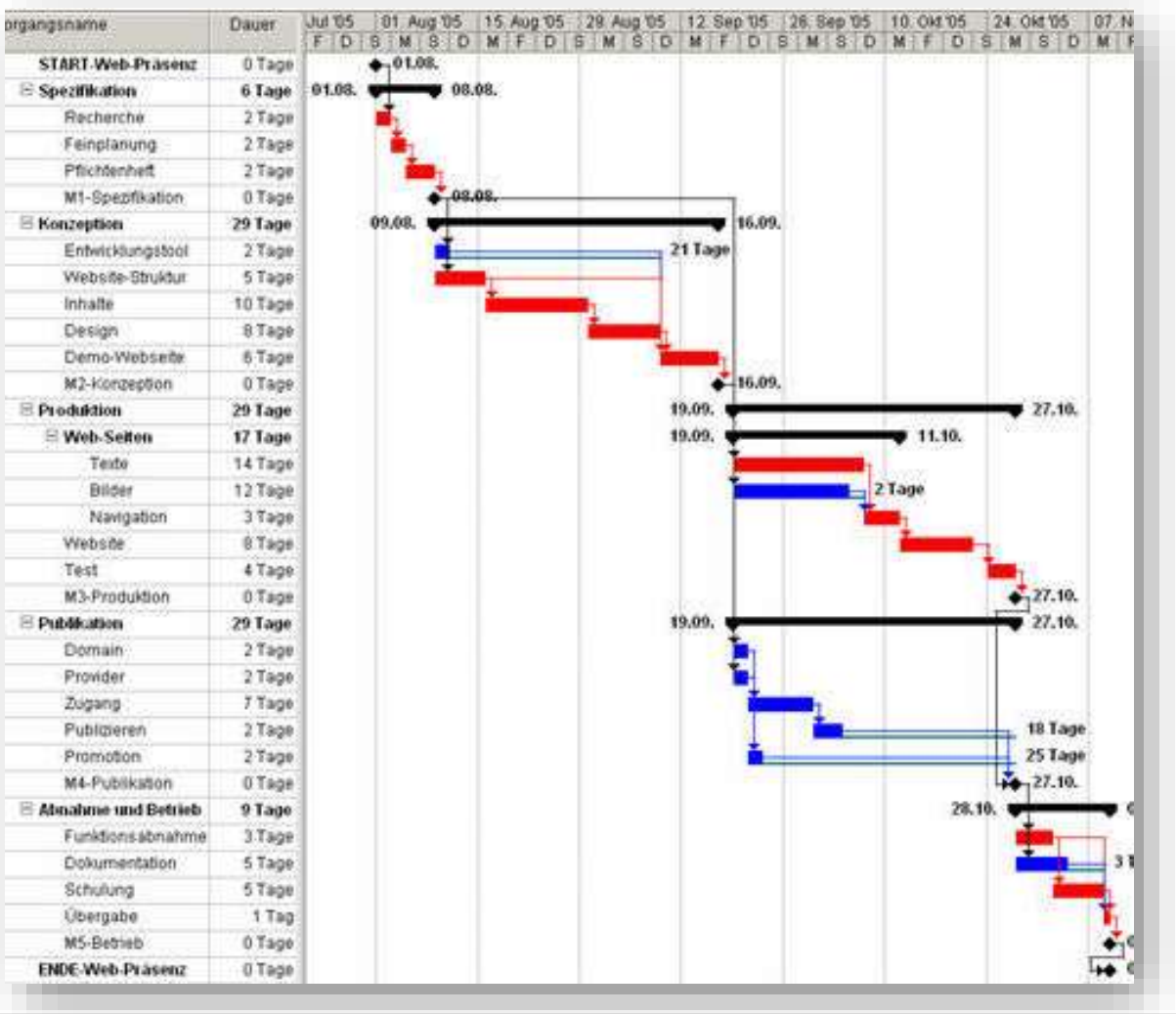
- Grundlagen
  - Bei CPM-Plänen werden die Vorgänge als gerichtete Kanten und die Ereignisse/Ergebnisse als Knoten dargestellt.
  - Sie werden speziell in Netzplänen bzw. Gantt-Charts eingesetzt
  - Voraussetzung ist, dass alle Vorgänge des Projekts mit der jeweiligen individuellen Dauer richtig zueinander in Beziehung gesetzt werden.
- Mehrwert und Ziele
  - Zwang zum exakten Durchdenken des Projektablaufs
  - übersichtliche Darstellung der Abhängigkeiten
  - Möglichkeit der Minimierung der Projektdauer
  - höhere Sicherheit in der Termineinhaltung
  - deutliche Hervorhebung von Projektengpässen
  - Rechtzeitiges Erkennen möglicher Verzögerungen
    → Abschätzen von Konsequenzen



Kritische Pfad Methode (CPM)

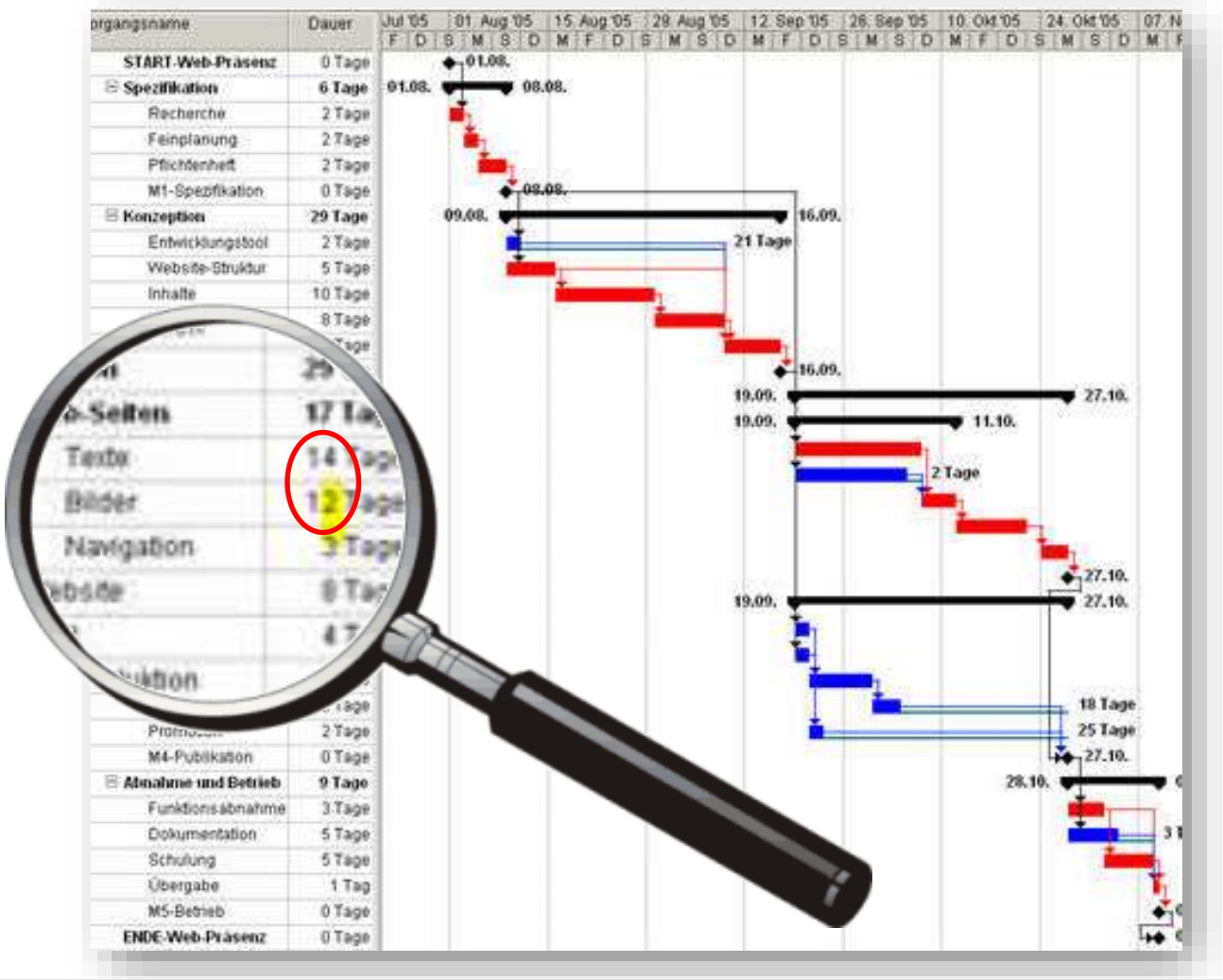
(s. IT I → Projektmanagement)

- Herausforderung: Kritischer Pfad in Gantt-Chart
  - Hier: Internet-Auftritt erstellen
- *Remark: Im Gegensatz zur Pfadsuche muss jeder Pfad gegangen werden*



Beispiel: Internet-Auftritt

- Die rot hervorgehobenen Vorgänge bilden den kritischen Weg.
  - Wenn hier was schiefgeht, ist der Endtermin des Gesamtprojektes gefährdet.
- Die blauen Vorgänge dagegen sind unkritisch
  - sie haben keinen Einfluss auf das Projektende.

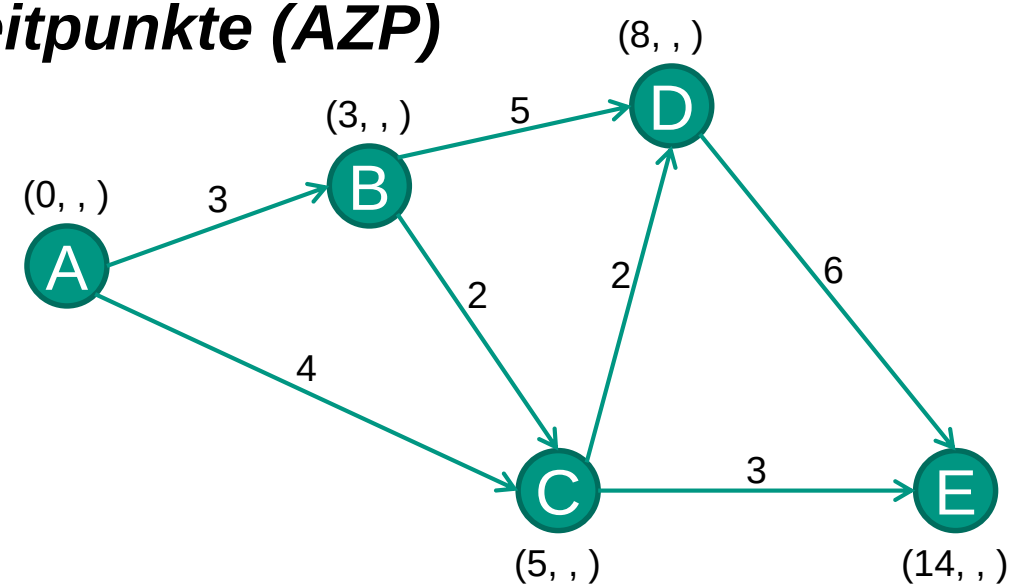


Berechnung des kritischen Pfades in 2 Schritten

- Vorwärtsterminierung
  - Ausgehend vom geplanten frühesten Startzeitpunkt des Projekts wird nacheinander für alle Vorgänge in die Zukunft hinein gerechnet.
  - Damit ergeben sich die frühesten Anfangszeitpunkte aller Vorgänge.
- Rückwärtsterminierung
  - Umgekehrtes Vorgehen:
    Vom Zeitpunkt des geplanten Projektendes wird nun in Richtung der Gegenwart gerechnet.
  - Es ergeben sich die spätesten Endzeitpunkte der einzelnen Vorgänge.
- Besteht für einen Vorgang zwischen dem Ergebnis der Vor- und der Rückwärtsrechnung eine Differenz, so liegt eine Pufferzeit vor.
  - Ein Vorgang mit Pufferzeit kann zwischen dem frühesten Anfangstermin und seinem spätesten Endtermin um diese Pufferzeit verschoben werden, ohne dass dadurch die Projektdauer negativ beeinflusst wird.
- Ein Weg durch den Netzplan vom Start- zum Zielknoten, bei dem kein Ereignis Pufferzeiten hat, wird als kritischer Pfad bezeichnet.
  - Eine Verzögerung bei einem der Ereignisse des kritischen Pfades hat immer eine Verzögerung des Endtermins des Projektes zur Folge.

CPM: *Früheste Anfangszeitpunkte (AZP)*

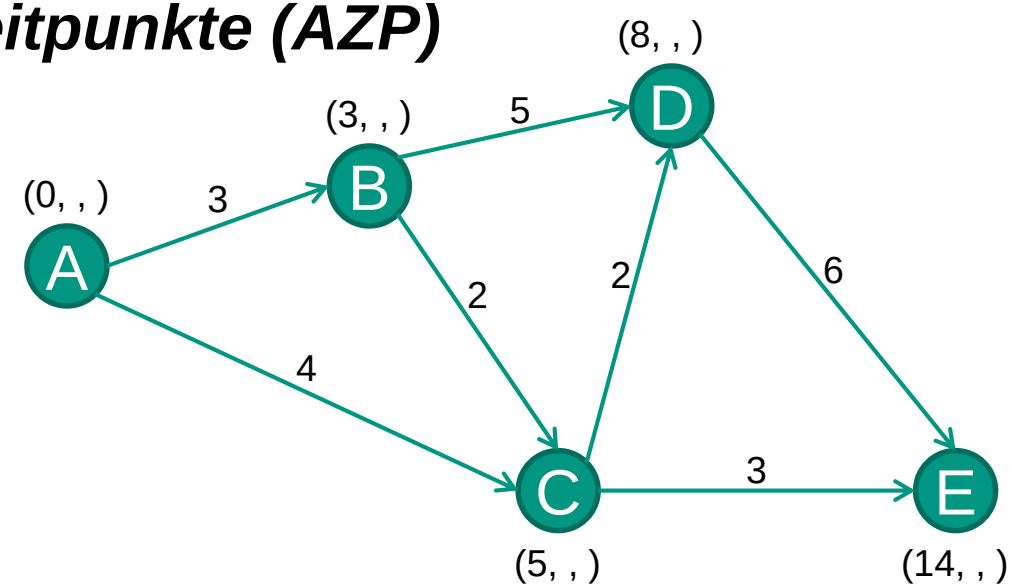
Vorwärtsterminierung



Fort-schritt	Vorgänger	AZP Vorgänger plus Zeit des Arbeitsschrittes	Größte Summe = Frühester Anfangszeitpunkt (eines Knotens)
A			
B			
C			
D			
E			

CPM: *Früheste Anfangszeitpunkte (AZP)*

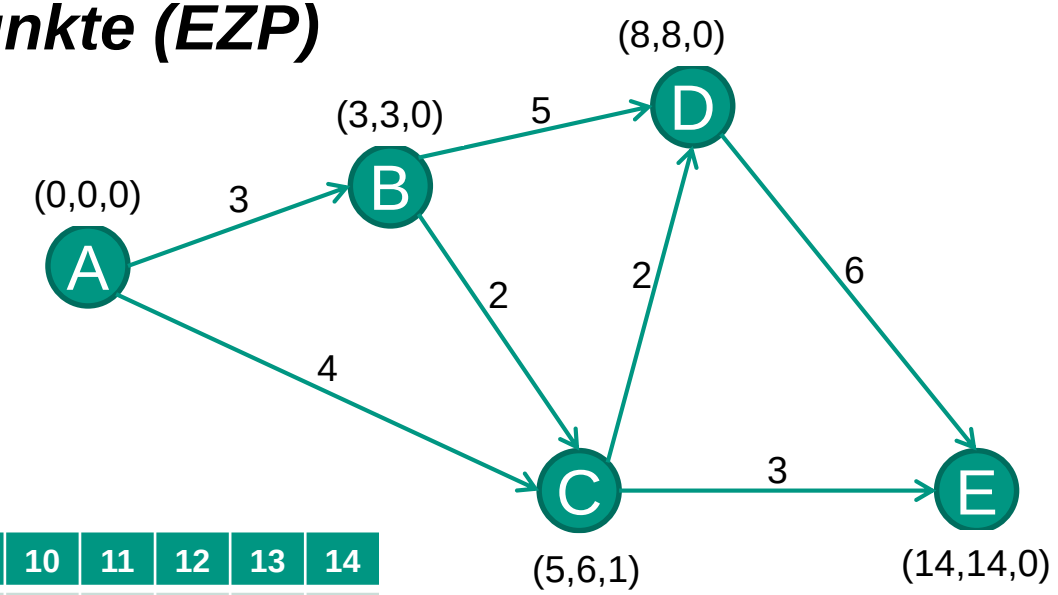
Vorwärtsterminierung



Fort-schritt	Vorgänger	AZP Vorgänger plus Zeit des Arbeitsschrittes	Größte Summe = Frühester Anfangszeitpunkt (eines Knotens)
A	-	0 + 0 = 0	0
B	A	0 + 3 = 3	3
C	A	0 + 4 = 4	
	B	3 + 2 = 5	5
D	B	3 + 5 = 8	8
	C	5 + 2 = 7	
E	C	5 + 3 = 8	
	D	8 + 6 = 14	14

CPM: *Späteste Endzeitpunkte (EZP)*

Rückwärtsterminierung



„Nuller“ zusammenstellen = CP

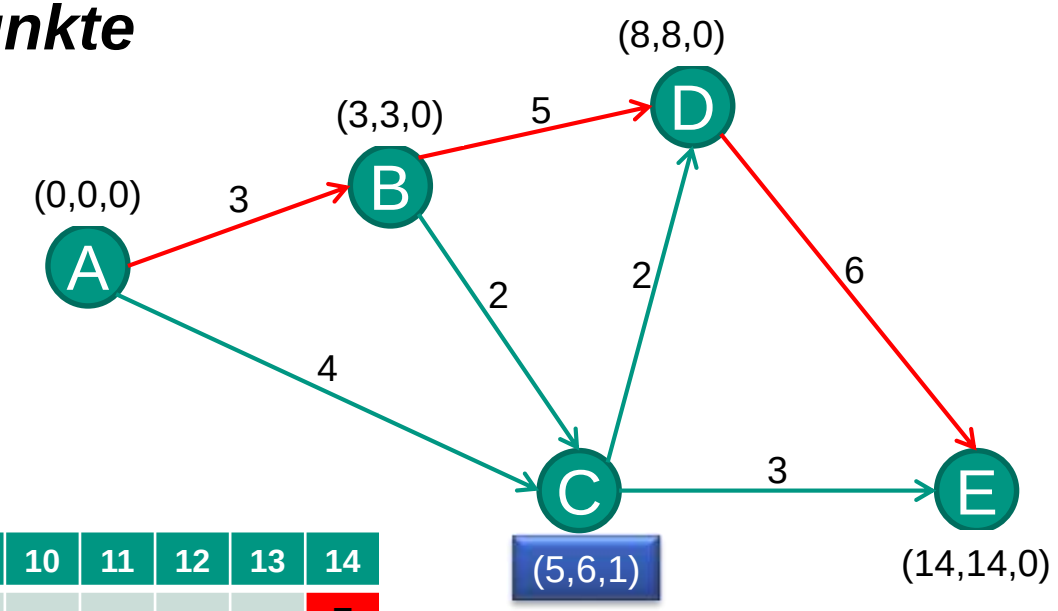
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A			B					D						E

Keine Pufferzeit

Fort-schritt	Nachfolger	EZP für Nachfolger minus Zeit des Arbeitsschrittes	Kleinste Differenz = Spätester Endtermin
E			
D			
C			
B			
A			

CPM: *Späteste Endzeitpunkte*

Rückwärtsterminierung



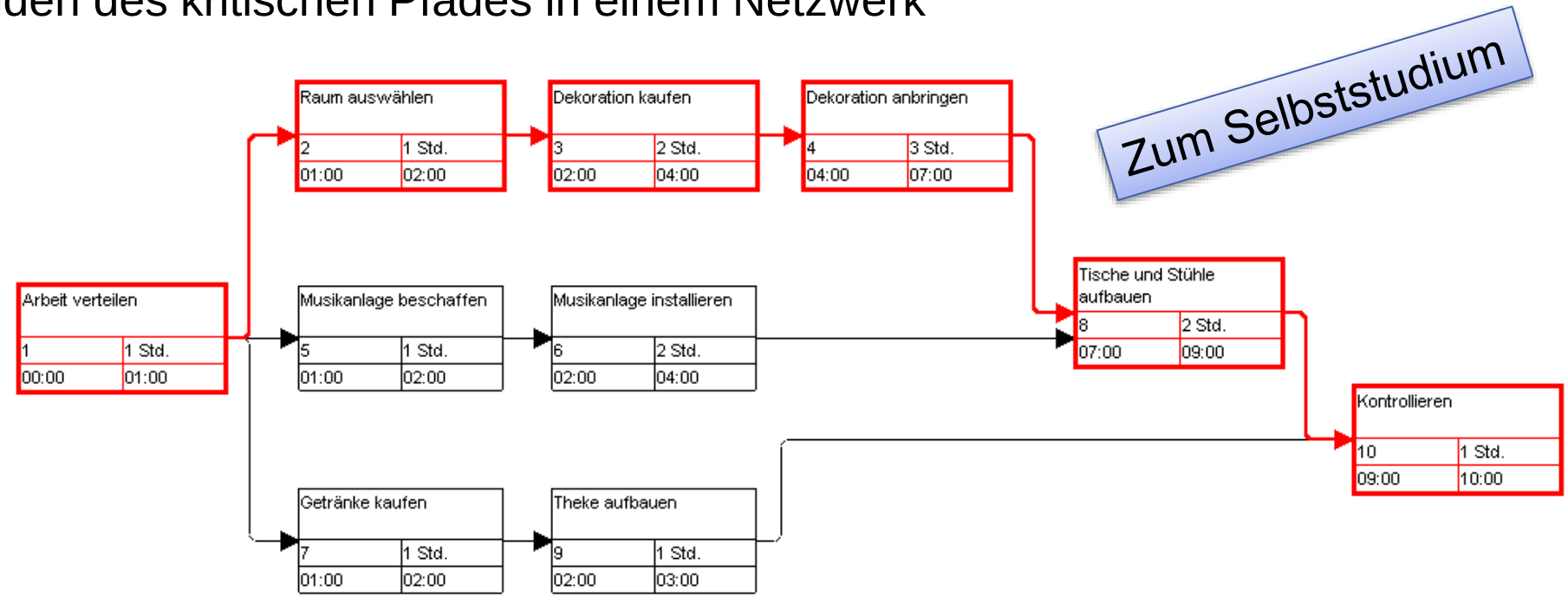
„Puffer“ darstellen

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A			B		C	C		D						E

Fort-schritt	Nachfolger	EZP für Nachfolger minus Zeit des Arbeitsschrittes	Kleinste Differenz = Spätester Endtermin
E	-	14 – 0 = 14	14
D	E	14 – 6 = 8	8
C	D	8 – 2 = 6	6
	E	14 – 3 = 11	
B	C	6 – 2 = 4	4
	D	8 – 5 = 3	3
A	B	3 – 3 = 0	0
	C	6 – 4 = 2	2

Beispiel: Betriebsfest organisieren

- Ziel: Finden des kritischen Pfades in einem Netzwerk



- Ein Vorgang kann erst beginnen, wenn alle vorhergehenden Vorgänge (Vorgänger) abgeschlossen sind.
- Müssen mehrere Vorgänge beendet sein, bevor ein folgender Vorgang (Nachfolger) beginnen kann, so enden sie dort.

Übung: Kundenauftrag durchführen

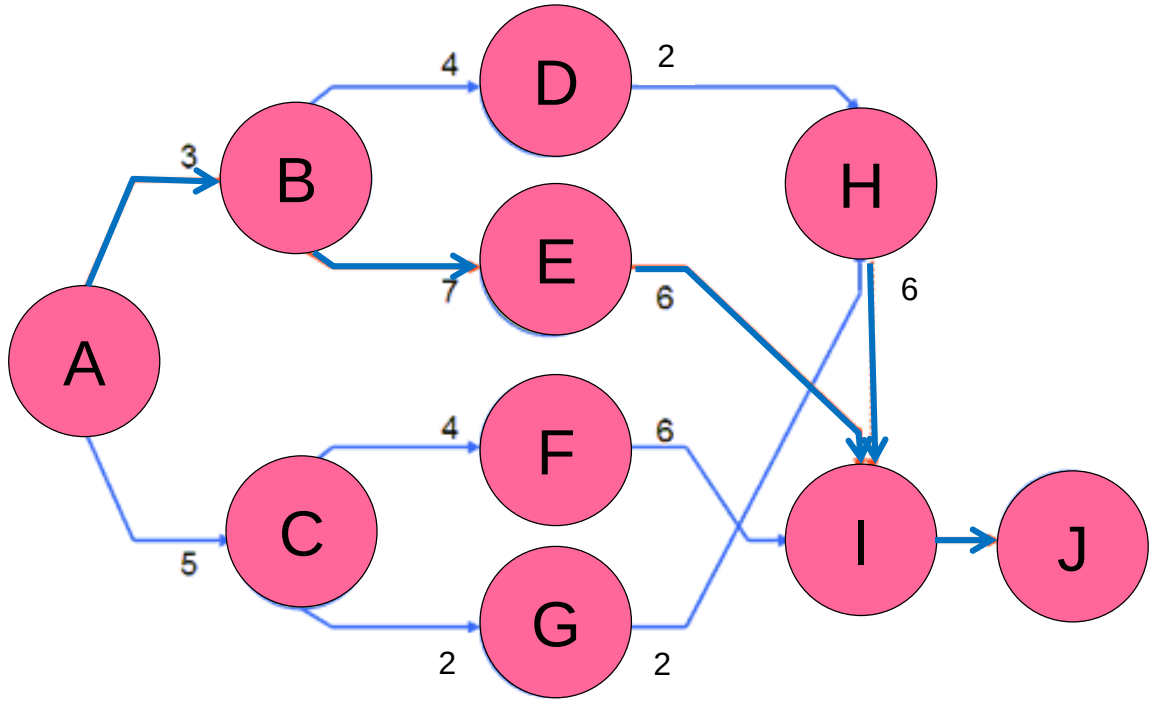
Vorgangsliste *mit* Zeitangaben und Zeitanalyse

Vorgangs-Nr.	Vorgangsbezeichnung	Vorgänger	Nachfolger	Dauer/ Tage
A	Auftragseingang			-
B	Material bestellen			3
C	Arbeitspläne erstellen			5
D	Materialkosten errechnen			4
E	Lieferzeit des Materials			7
F	Arbeitskräfte einweisen			4
G	Lohnkosten ermitteln			2
H	Selbstkosten kalkulieren			2
I	Arbeit ausführen			6
J	Auslieferung			-

Übung: Kundenauftrag durchführen

Vorgangsliste *mit* Zeitangaben und Zeitanalyse

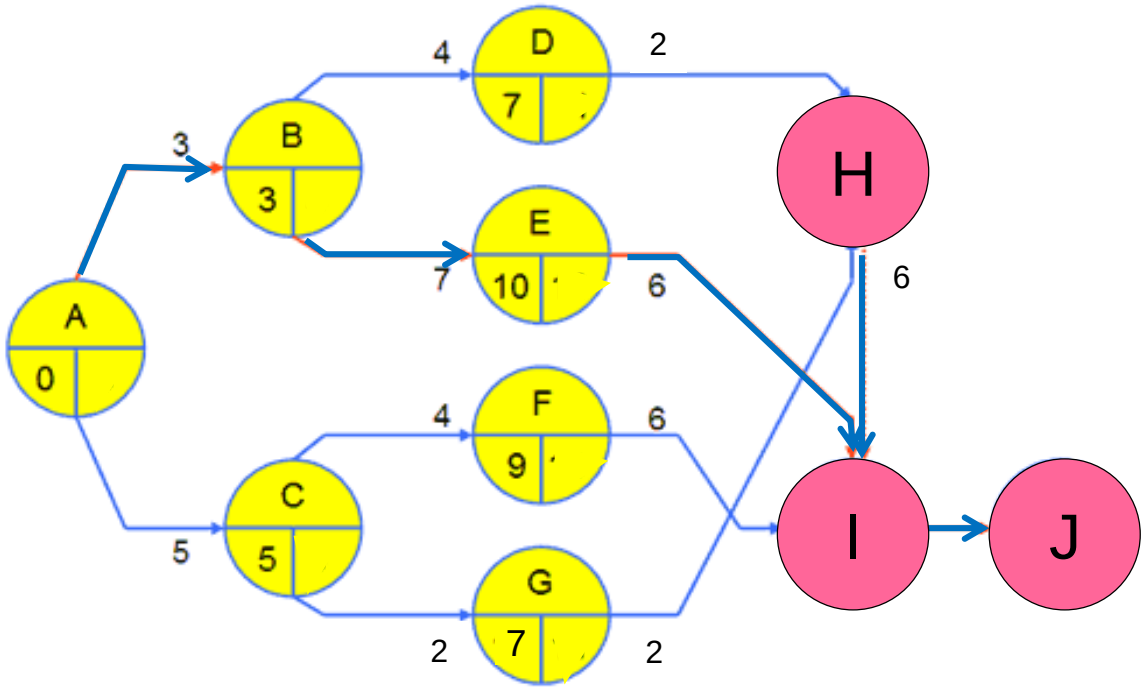
Vorgangs-Nr.	Vorgangsbezeichnung	Vorgänger	Nachfolger	Dauer/ Tage
A	Auftragseingang	-	B, C	-
B	Material bestellen	A	D, E	3
C	Arbeitspläne erstellen	A	F, G	5
D	Materialkosten errechnen	B	H	4
E	Lieferzeit des Materials	B	I	7
F	Arbeitskräfte einweisen	C	I	4
G	Lohnkosten ermitteln	C	H	2
H	Selbstkosten kalkulieren	D, G	I	2
I	Arbeit ausführen	E, F, H	J	6
J	Auslieferung	I	-	-



Übung: Kundenauftrag durchführen

Vorgangsliste *mit* Zeitangaben und Zeitanalyse

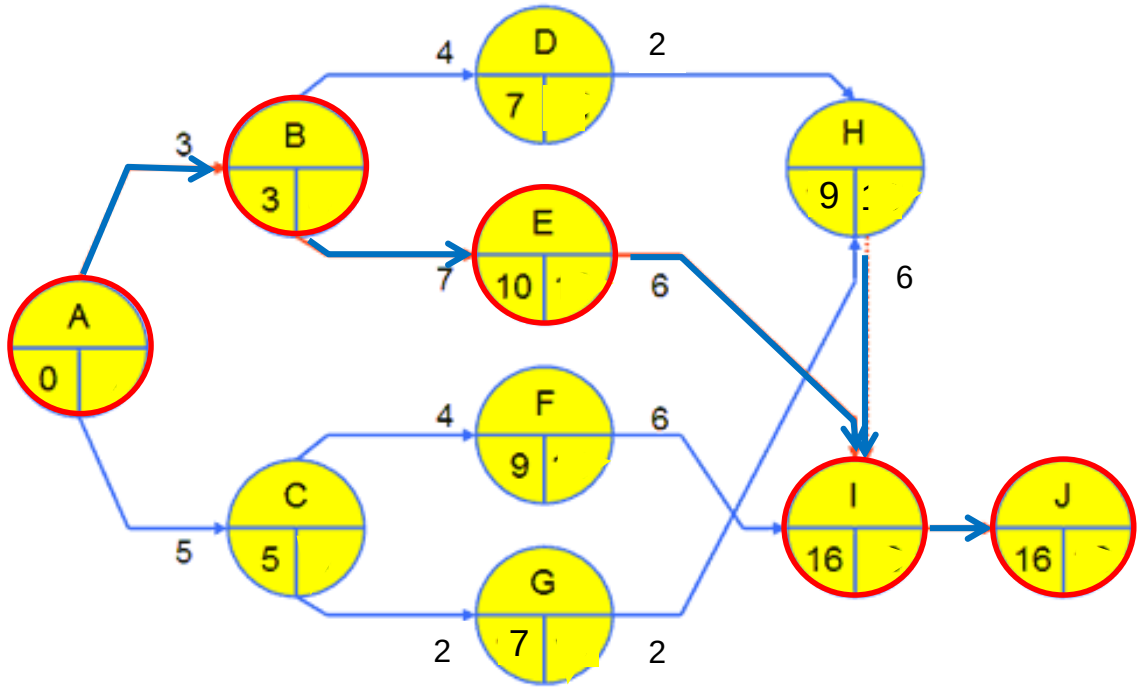
Vorgangs-Nr.	Vorgangsbezeichnung	Vorgänger	Nachfolger	Dauer/ Tage
A	Auftragseingang	-	B, C	-
B	Material bestellen	A	D, E	3
C	Arbeitspläne erstellen	A	F, G	5
D	Materialkosten errechnen	B	H	4
E	Lieferzeit des Materials	B	I	7
F	Arbeitskräfte einweisen	C	I	4
G	Lohnkosten ermitteln	C	H	2
H	Selbstkosten kalkulieren	D, G	I	2
I	Arbeit ausführen	E, F, H	J	6
J	Auslieferung	I	-	-



Übung: Kundenauftrag durchführen

Vorgangsliste *mit* Zeitangaben und Zeitanalyse

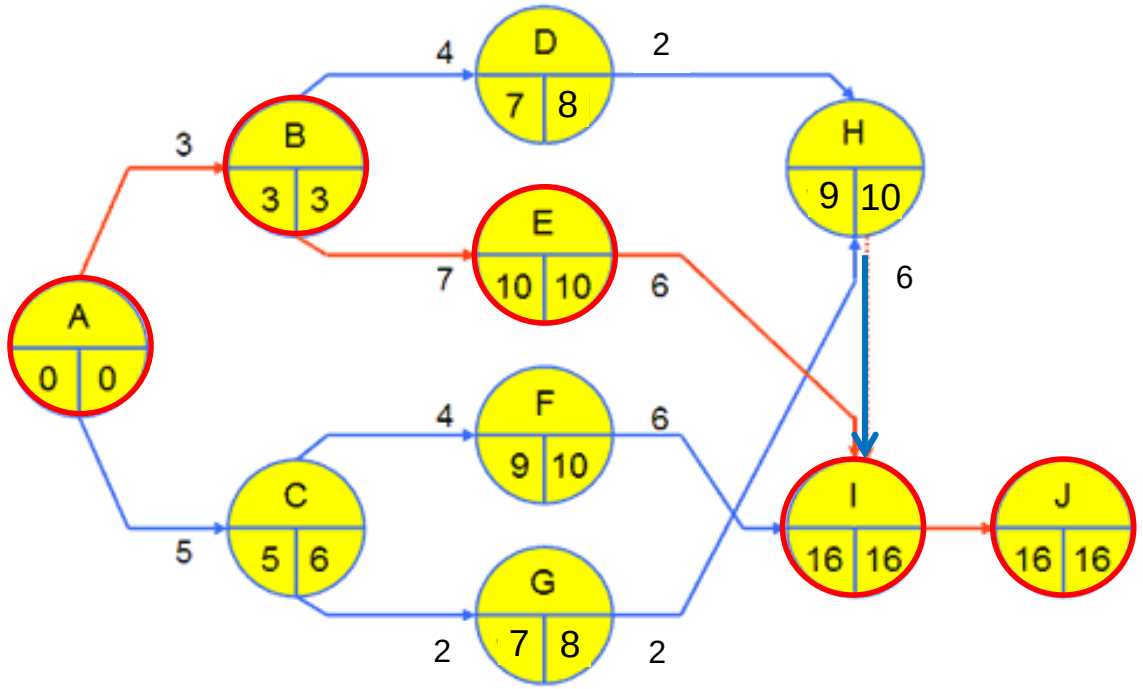
Vorgangs-Nr.	Vorgangsbezeichnung	Vorgänger	Nachfolger	Dauer/ Tage
A	Auftragseingang	-	B, C	-
B	Material bestellen	A	D, E	3
C	Arbeitspläne erstellen	A	F, G	5
D	Materialkosten errechnen	B	H	4
E	Lieferzeit des Materials	B	I	7
F	Arbeitskräfte einweisen	C	I	4
G	Lohnkosten ermitteln	C	H	2
H	Selbstkosten kalkulieren	D, G	I	2
I	Arbeit ausführen	E, F, H	J	6
J	Auslieferung	I	-	-



Übung: Kundenauftrag durchführen

Vorgangsliste *mit* Zeitangaben und Zeitanalyse

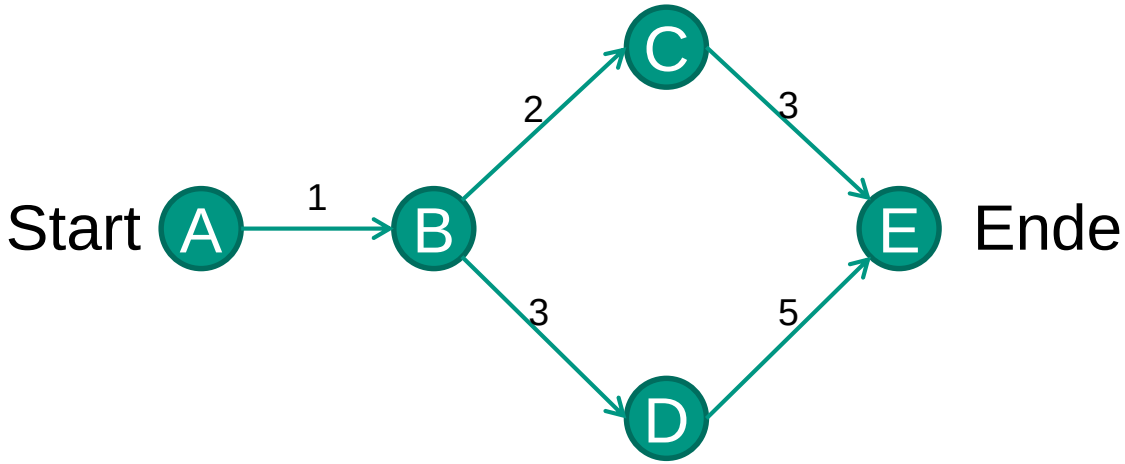
Vorgangs-Nr.	Vorgangsbezeichnung	Vorgänger	Nachfolger	Dauer/ Tage
A	Auftragseingang	-	B, C	-
B	Material bestellen	A	D, E	3
C	Arbeitspläne erstellen	A	F, G	5
D	Materialkosten errechnen	B	H	4
E	Lieferzeit des Materials	B	I	7
F	Arbeitskräfte einweisen	C	I	4
G	Lohnkosten ermitteln	C	H	2
H	Selbstkosten kalkulieren	D, G	I	2
I	Arbeit ausführen	E, F, H	J	6
J	Auslieferung	I	-	-



Übung: CPM

Zum Selbststudium

- Gegeben sei folgender Netzplan:

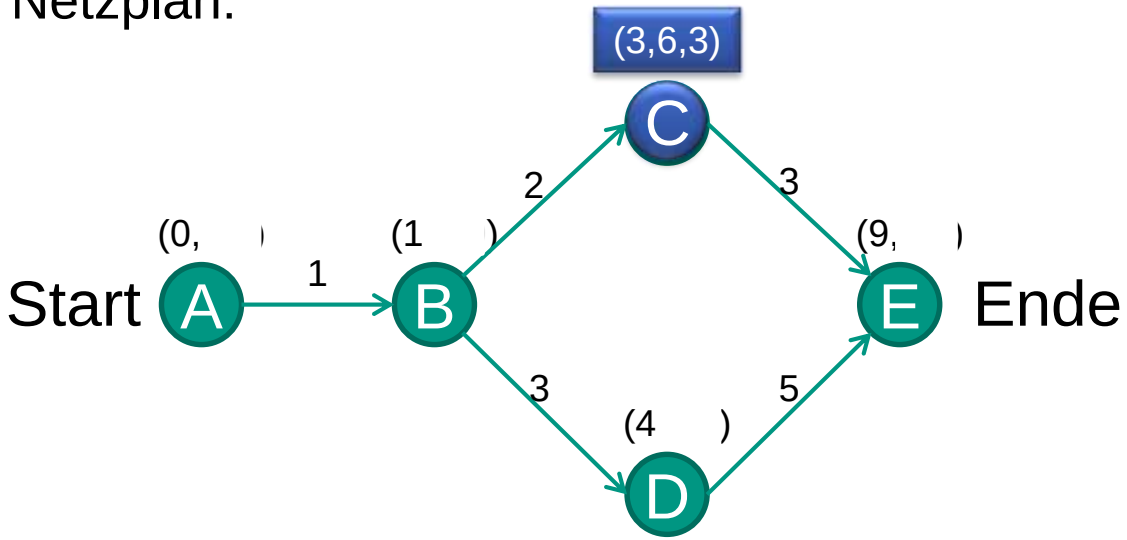


- Ermitteln Sie mit Hilfe der Critical Path Method, welcher Prozess später gestartet werden kann und in welchem Zeitraum er gestartet werden muss.

Übung: CPM – Lsg.

Zum Selbststudium

- Gegeben sei folgender Netzplan:

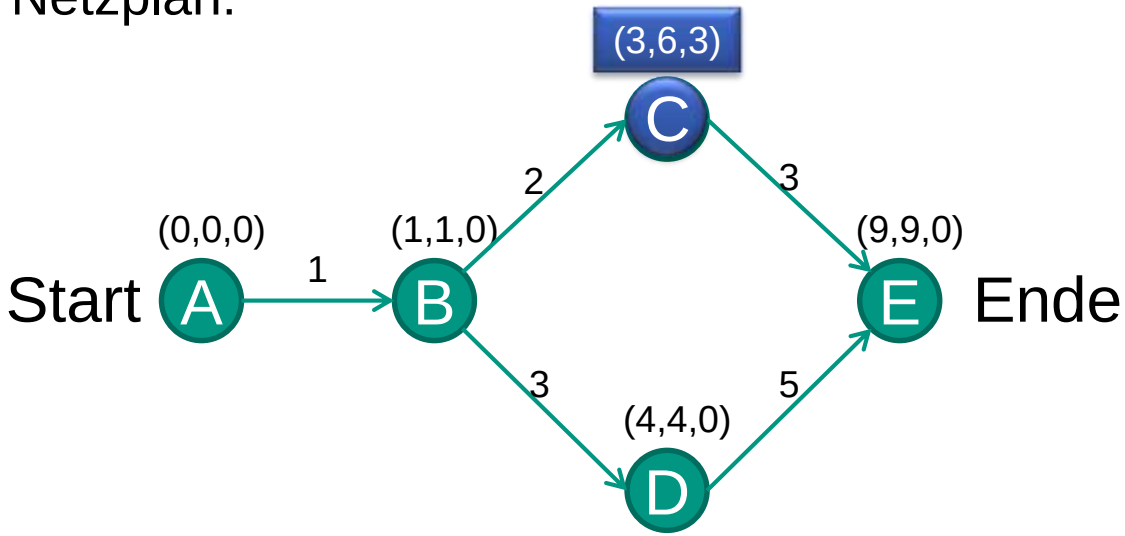


- Ermitteln Sie mit Hilfe der Critical Path Method, welcher Prozess später gestartet werden kann und in welchem Zeitraum er gestartet werden muss.

Übung: CPM – Lsg.

Zum Selbststudium

- Gegeben sei folgender Netzplan:



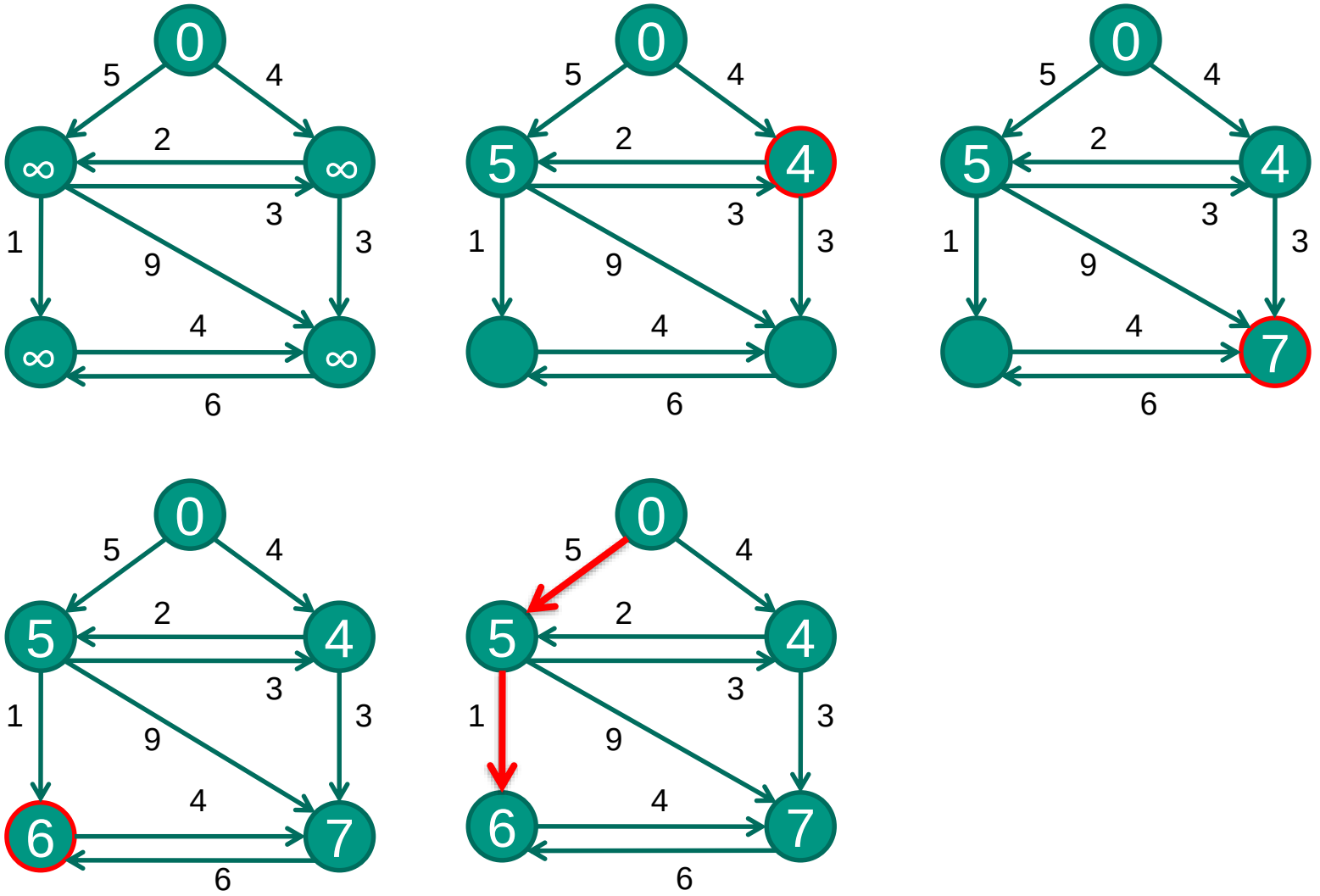
- Prozess C kann zwischen der 3. und 6. Woche gestartet werden.

- Kürzester Pfad
- Dijkstra-Algorithmus
- Critical Path Method




Backup
Shift

Ablauf: Dijkstra-Algorithmus



Karlsruher Institut für Technologie

Institut für Technik der
Informationsverarbeitung

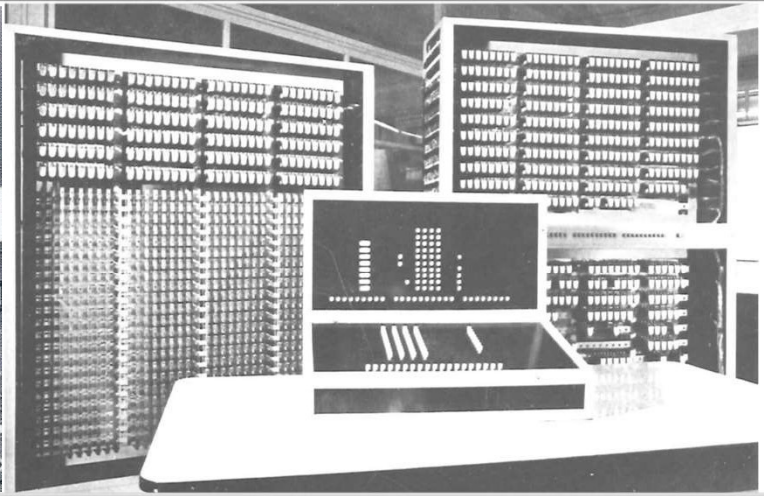
Institutsleitung
Prof. Dr.-Ing. J. Becker
Prof. Dr.-Ing. E. Sax
Prof. Dr. rer. nat. W. Stork

5. Optimierungsalgorithmen

Informationstechnik II

Prof. Dr.-Ing. Eric Sax







Inhalt IT2

4. Algorithmen auf Graphen

- Graphendefinition
- Binäre Suche
- Breitensuche
- Tiefensuche
- Zyklensuche
- Kürzester und kritischer Pfad

5. Optimierungsalgorithmen

- Partitionierung
- Anlagerungsverfahren
- Random Interchange
- Kernighan-Lin
- Greedy
- Simulated Annealing

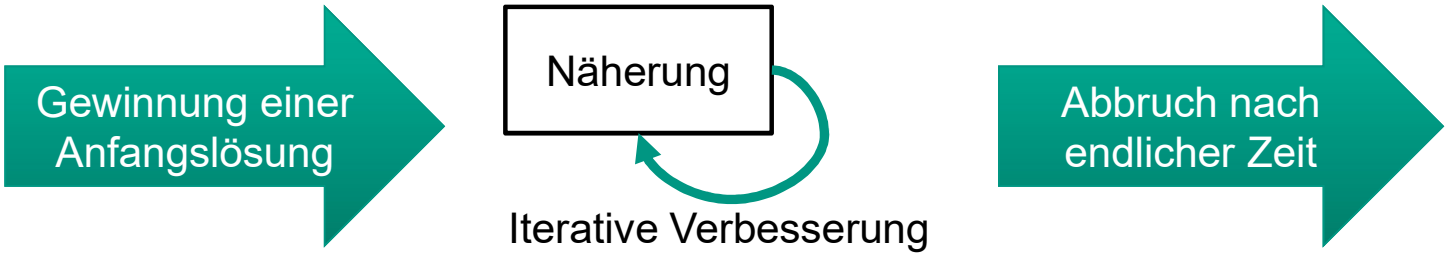


Was ist ein Optimierungsalgorithmus?

- Bei „komplexen“ Problemen ist die exakte Lösung oft nicht möglich oder sinnvoll.
  - Es gibt eine Vielzahl zulässiger Lösungen
  - Jeder Lösung ist ein Wert zugeordnet
  - Finden der Lösung mit dem „besten“ → dem „optimierten“ Wert
- Mögliche Gründe:
  - Formulierung als Algorithmus ist nicht möglich
  - Exakter Algorithmus hat Rechenzeit der Ordnung $O(n^x)$, $O(x^n)$ oder sogar $O(n!)$

Problemstellung

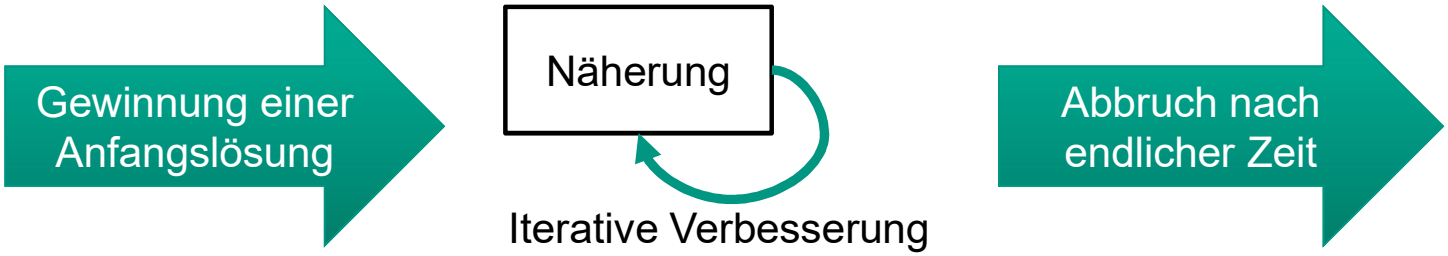
- Pragmatische Lösung: Heuristische Verfahren
  - Heuristik bezeichnet die Kunst, mit begrenztem Wissen und wenig Zeit zu guten Lösungen zu kommen.
  - Es bezeichnet ein analytisches Vorgehen, bei dem mit Hilfe von Mutmaßungen Schlussfolgerungen über das System getroffen werden.



Problemstellung: Optimierung

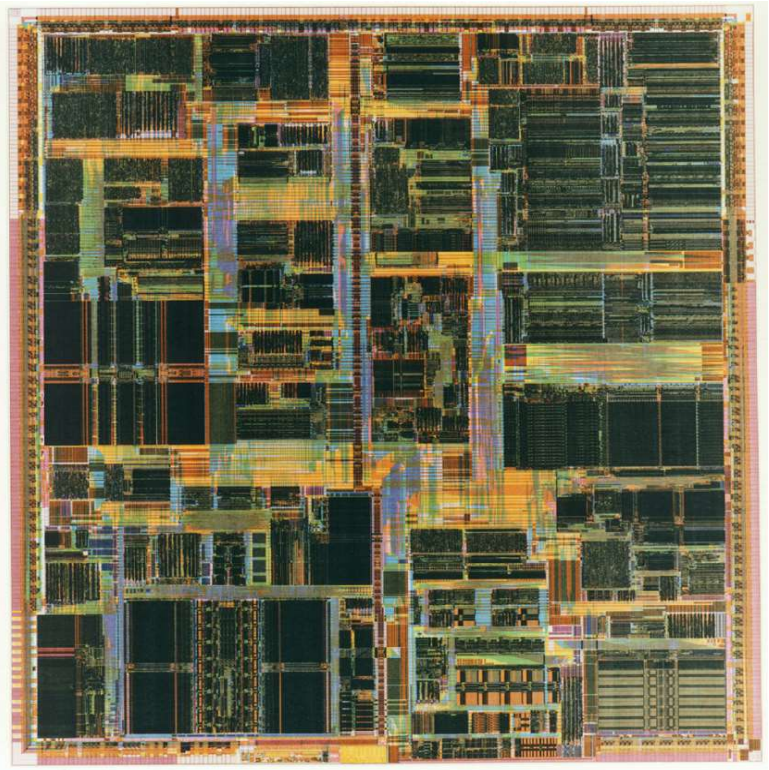
- Voraussetzung:
  - Bewertungsfunktion, Kostenfunktion
- Optimierungsziel (lokal):
  - Minimierung, Maximierung
- Ggf. auch nur ein Pareto-Optimum :

  Also ein Zustand, in dem es nicht möglich ist, eine Eigenschaft zu verbessern, ohne zugleich eine andere verschlechtern zu müssen.
- Pareto: 80-zu-20-Regel
  - 80 % der Ergebnisse mit 20 % des Gesamtaufwandes erreichen
  - Verbleibende 20 % der Ergebnisse benötigen mit 80 % die meiste Arbeit und werden vernachlässigt.



Anwendungsbeispiel – Chiplayout

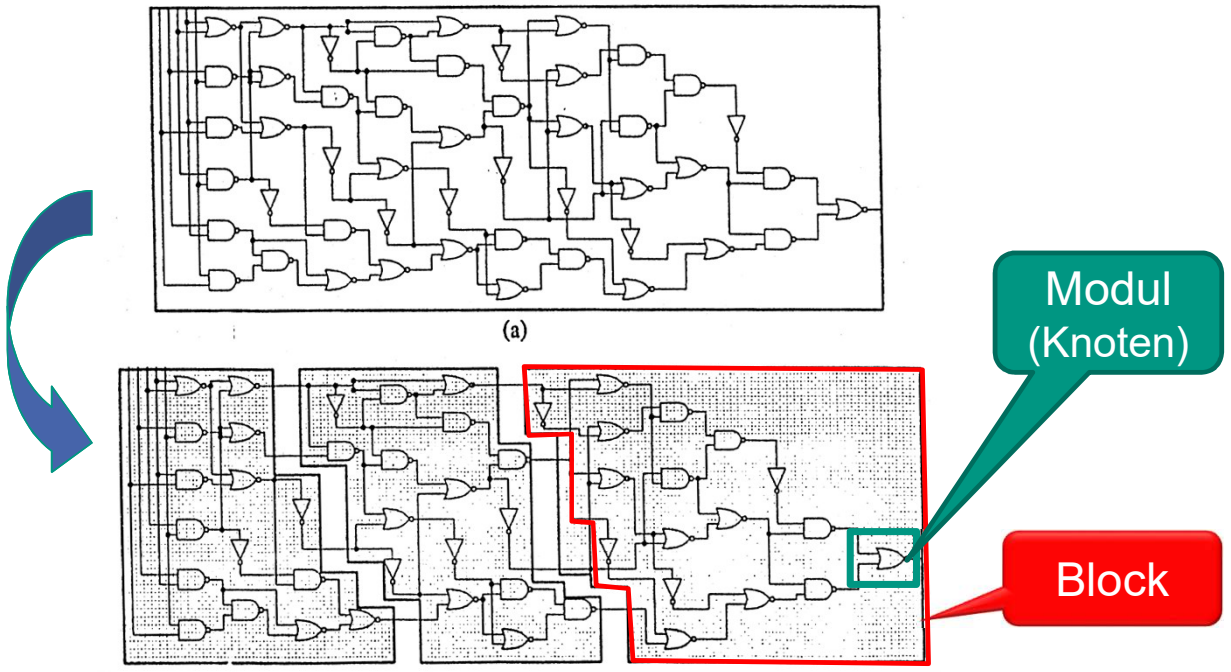
- Teilaufgaben beim Entwurf eines Chiplayout:
  - Partitionierung
  - Platzierung
  - Verdrahtung
- Viele Nebenbedingungen:
  - Geringe Gesamtfläche
  - Temperaturverhalten
  - Konnektivität
  - ...



P6 Dieshot

Partitionierung

- Chip-Entwicklung in kleinen Teilaufgaben (Wiederverwendung, Intellectual Property, Partnerentwicklungen, …)
  - ► Aufteilung der (geschätzten) Fläche eines Chips in einzelne funktionale Blöcke, die mit anderen Blöcken durch möglichst wenige kurze Leitungen verbunden sind:



Partitionierung – Aufgabenstellung

- Aufgabe: Finde eine Partition P der Modulmenge M = { m_1, ..., m_i, ...,m_n } aus Blöcken B mit den Unterscheidungskriterien:
  - Zahl der Verbindungen
  - Anzahl der Module
  - ...
- Vereinfachung:
  - Finde eine Bi-Partition P = { B_1, B_2 } mit minimaler Kostenfunktion
- Heuristische Verfahren:
  - Konstruktive Methoden:
    - Anlagerungsverfahren
  - Iteratives Verbessern:
    - Random Interchange
    - Kernighan-Lin
    - Greedy
    - Simulated Annealing
    - Evolutionäre Algorithmen (hier nicht)

Partitionierung – Überdeckung
(mathematische Definition)

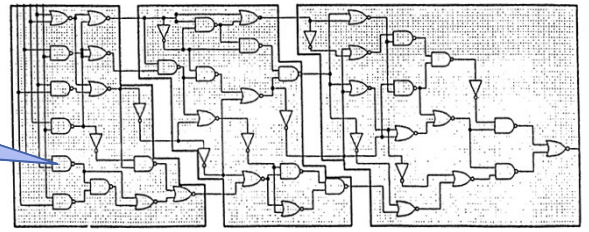
- Für „Partitionierung“ gilt:

  Modulmenge M = { m_1, ..., m_i, ..., m_n }

  Partition P = { B_1, B_2, B_3, …, B_k }

$$\bigcup_{i=1}^{k} B_i = M \quad \text{und} \quad \forall i, j \left| i \neq j \right\rangle \; B_i \bigcap B_j = \varnothing$$

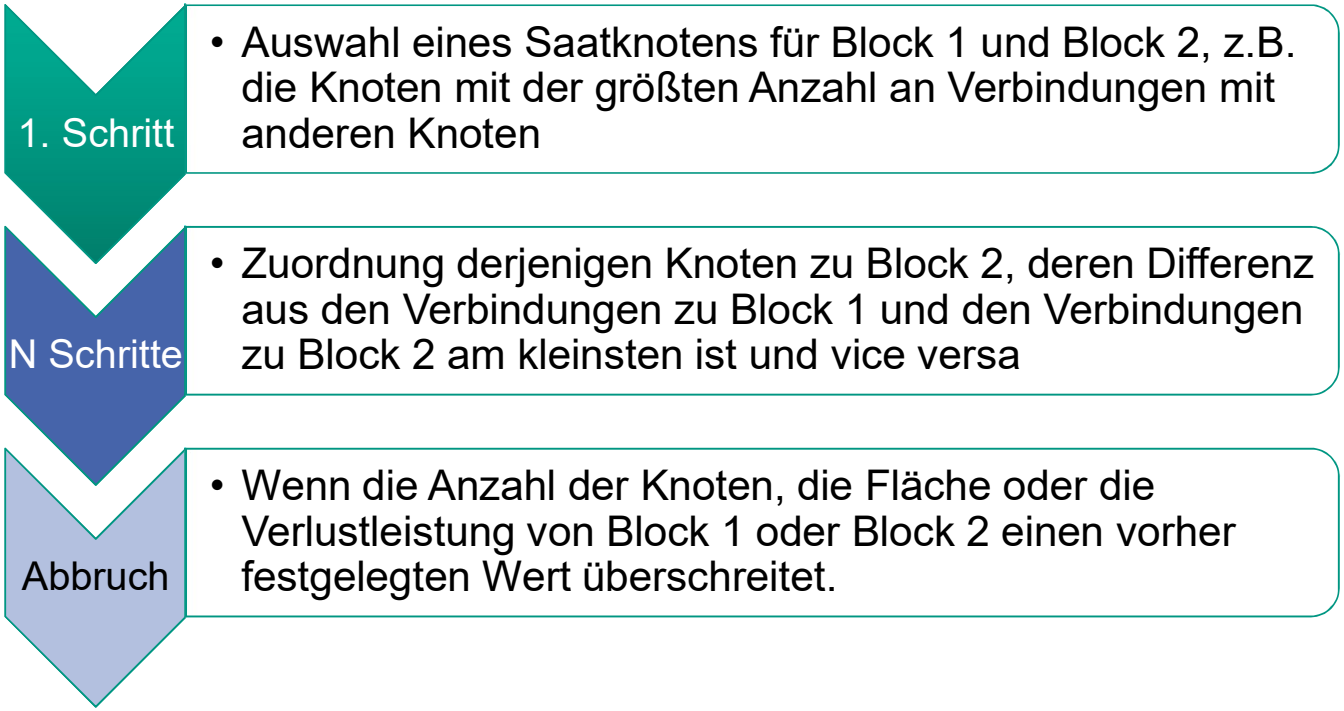
„Alle berücksichtigt“
„Keine Überlappungen“



- *Remark: Für „Überdeckung“ gilt lediglich*

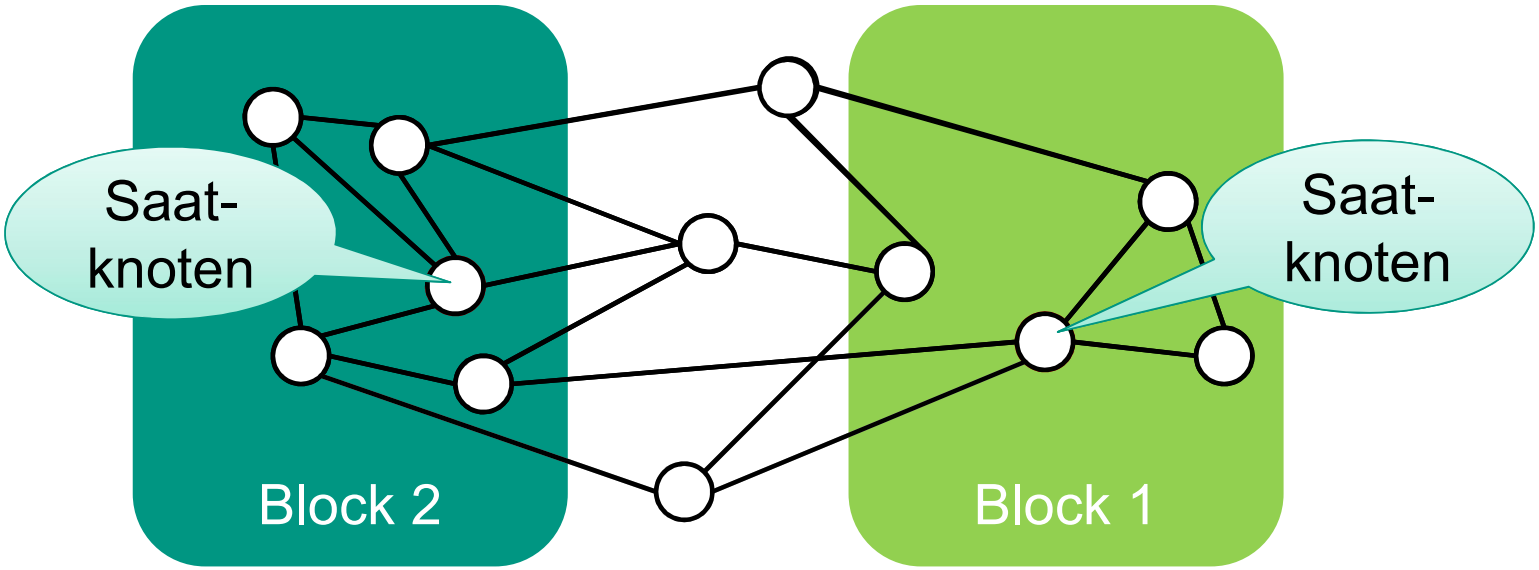
$$\bigcup_{i=1}^{k} B_i = M$$

Anlagerungsverfahren
(Bipartition)



- Problem: Kann in lokalem Minimum der Kostenfunktion enden

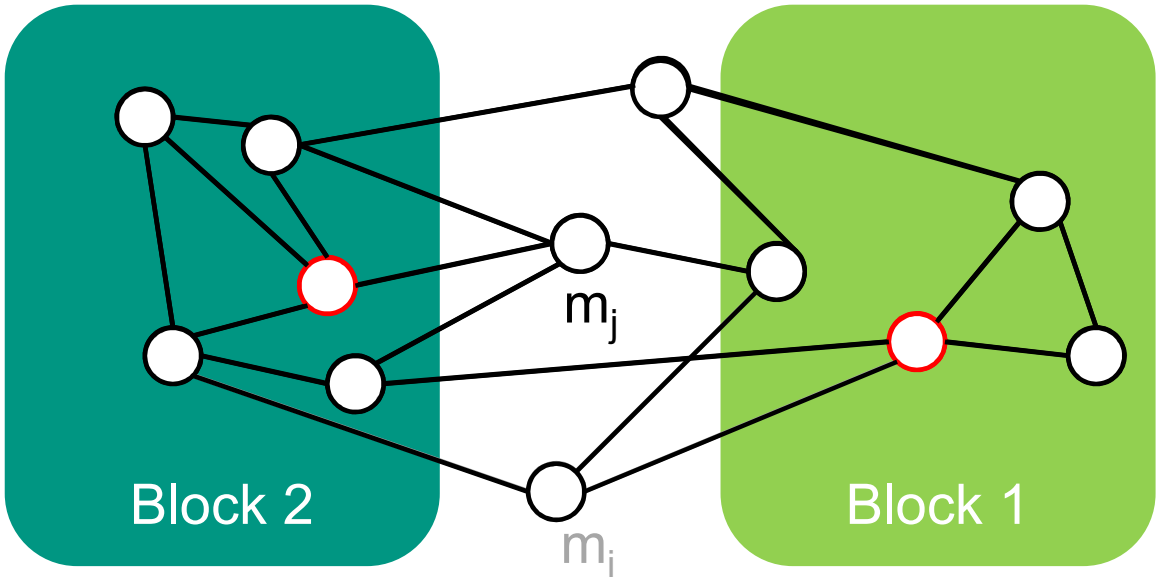
Beispiel: Anlagerungsverfahren



1. Schritt

- Auswahl eines Saatknotens für Block 1 und Block 2, z.B. die Knoten mit der größten Anzahl an Verbindungen mit anderen Knoten

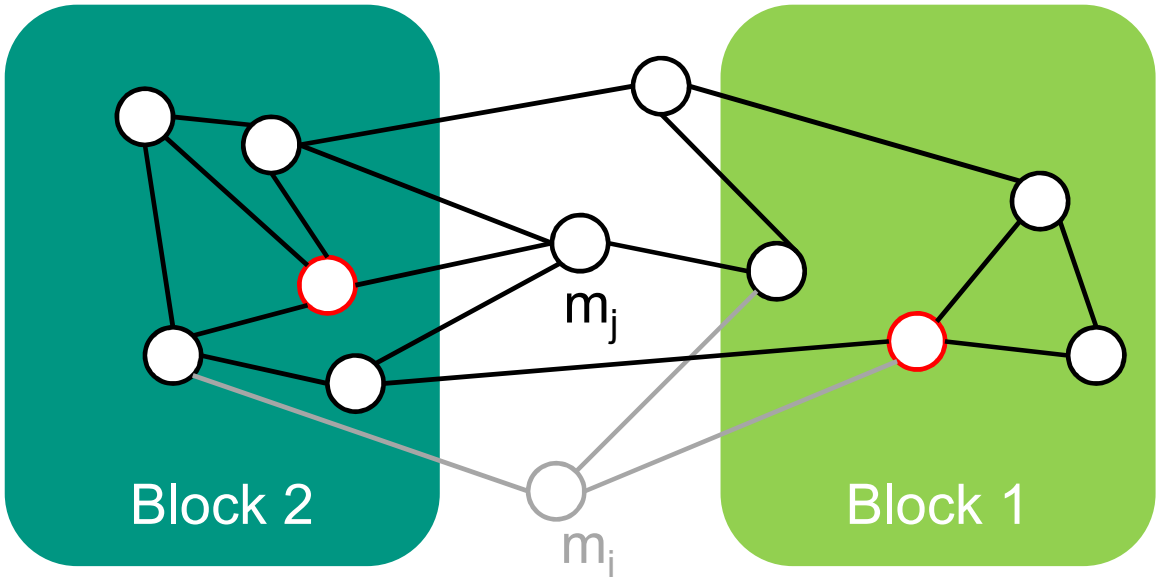
Beispiel: Anlagerungsverfahren



N Schritte

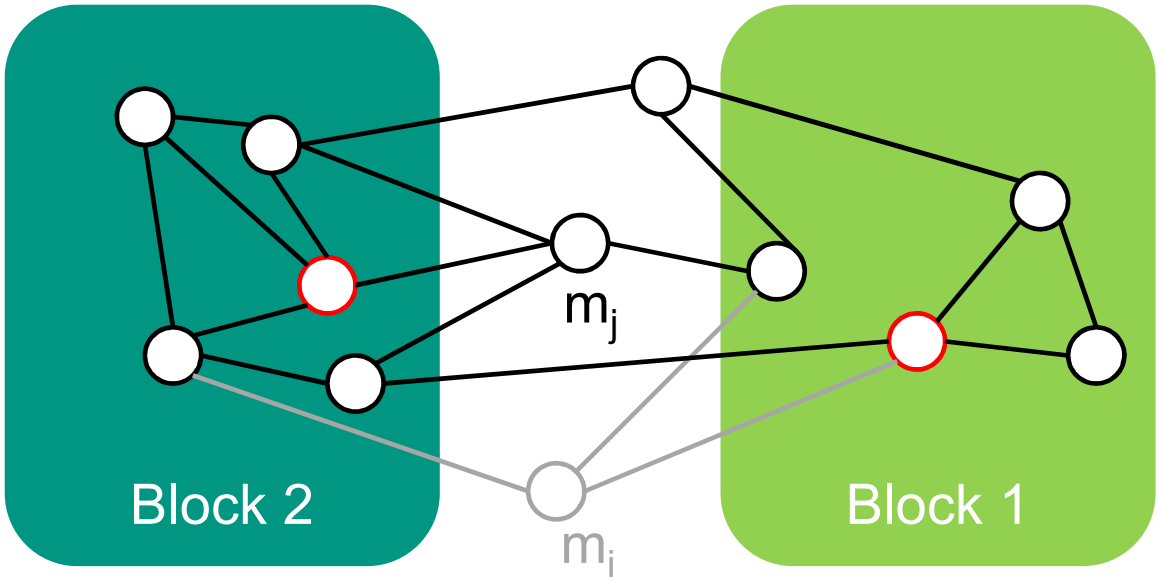
- Zuordnung derjenigen Knoten zu Block 2, deren Differenz aus den Verbindungen zu Block 1 und den Verbindungen zu Block 2 am kleinsten ist und vice versa

Beispiel: Anlagerungsverfahren



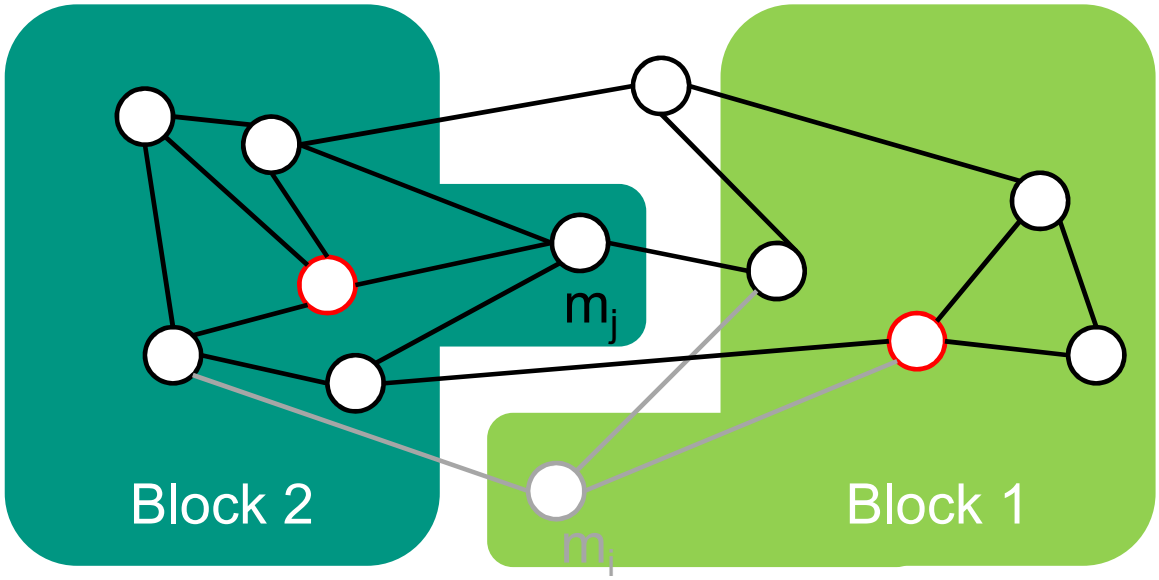
- Verbindungen zu Block 1: … mj: 1
- Verbindungen zu Block 2: … mj: 3
- Differenzen … -2
- Zuordnungen … ⇒ Block 2

Beispiel: Anlagerungsverfahren



- Verbindungen zu Block 1: … mj: 1 mi:2
- Verbindungen zu Block 2: … mj: 3 mi:1
- Differenzen … -2 1
- Zuordnungen … ⇒ Block 2 ⇒ Block 1

Beispiel: Anlagerungsverfahren



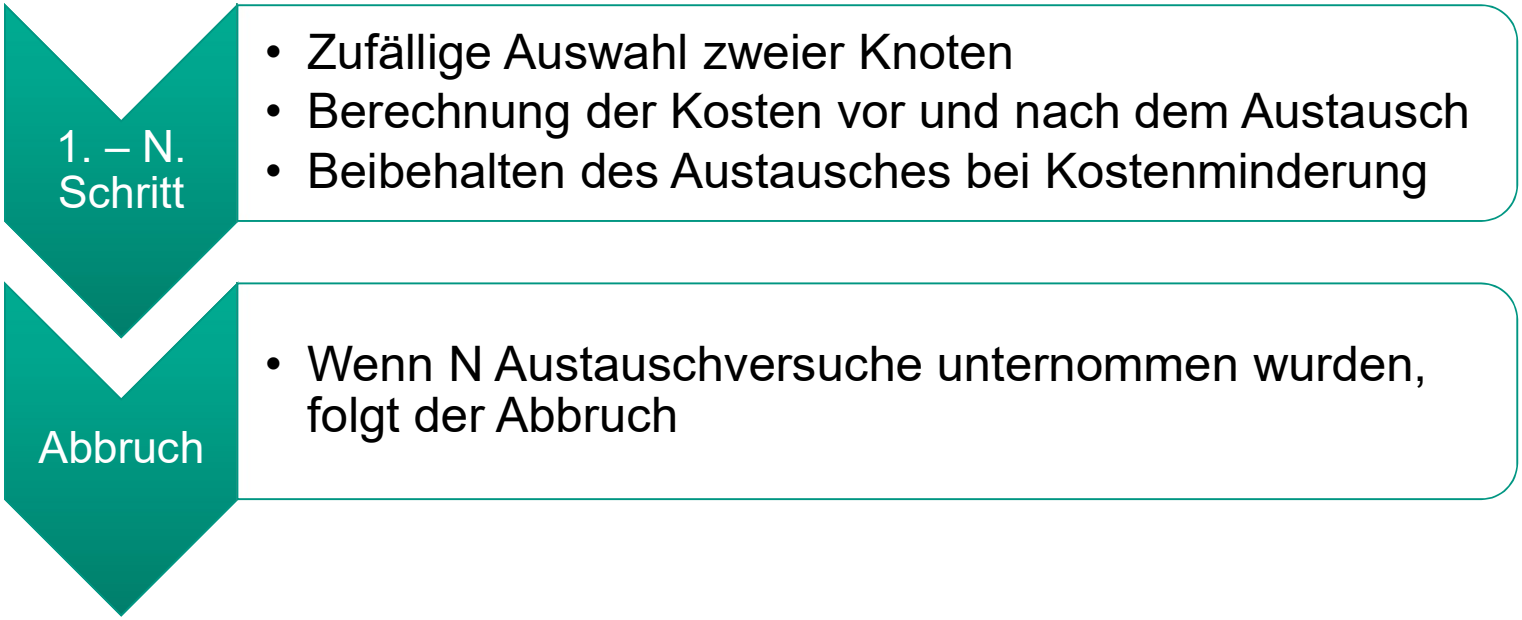
- Verbindungen zu Block 1: … mj: 1 mi:2
- Verbindungen zu Block 2: … mj: 3 mi:1
- Differenzen … -2 1
- Zuordnungen … ⇒ Block 2 ⇒ Block 1

- Anlagerungsverfahren



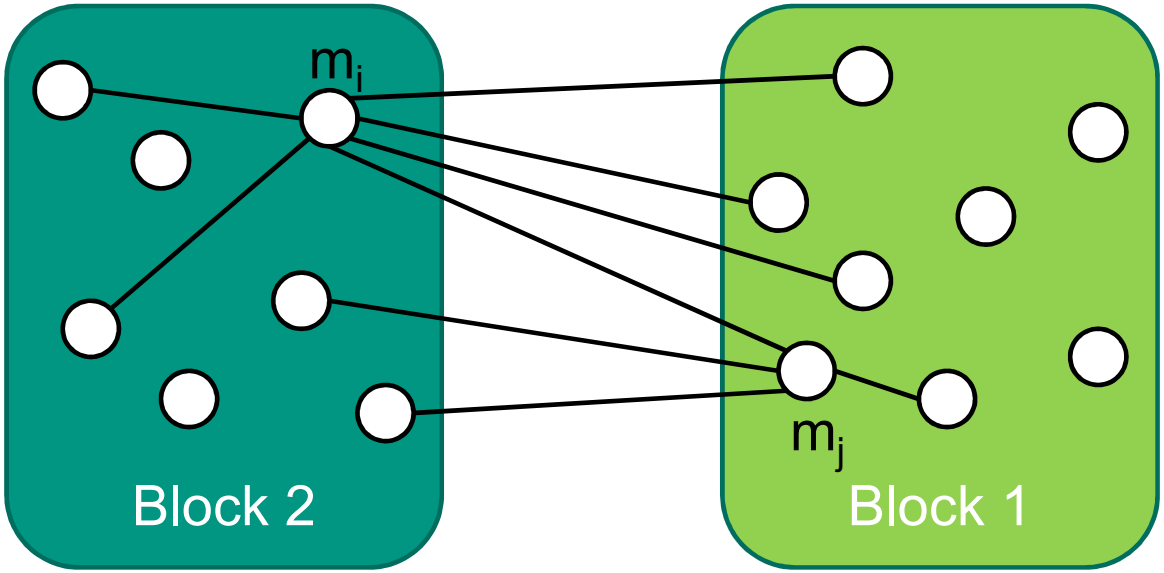
Random Interchange

- Iteratives Verbessern einer bestehenden Bipartition durch zufälliges Vertauschen von Knoten:



- Kosten entstehen durch interne und externe Kanten

Beispiel: Random Interchange



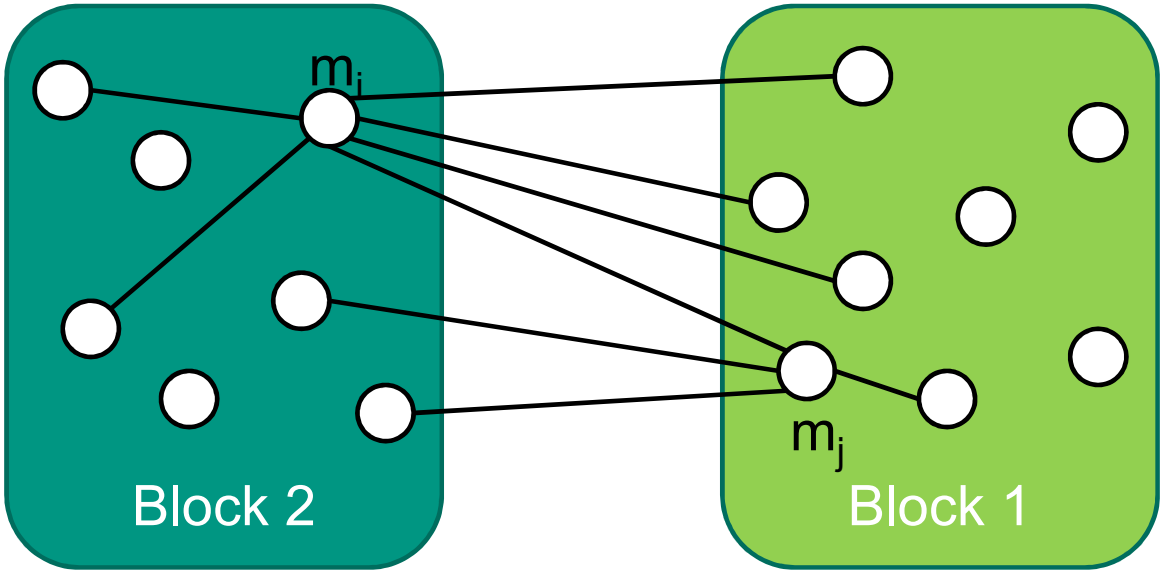
- Ziel ist, dass die Summe der Kantengewichte zwischen Block 1 und Block 2 minimal wird.
- Remark: *Hier sind alle Kantengewichte 1*

Kostenfunktion

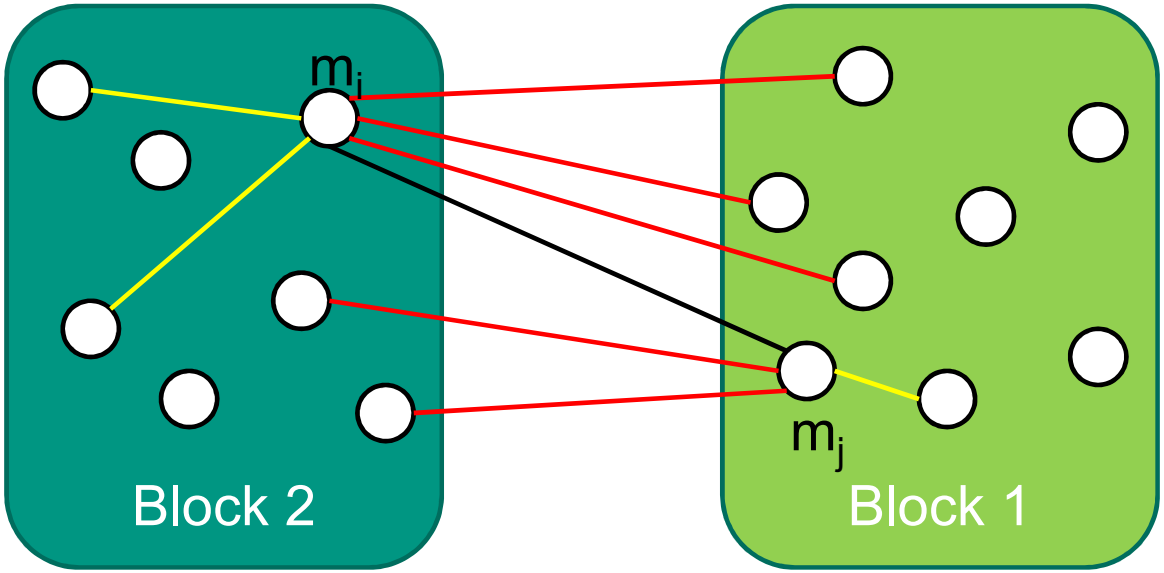
- Die internen Kosten von m_i, also die Summe aller Kantengewichte abgehend vom Knoten m_i in Block 2, wird als $I(m_i)$ bezeichnet.
- $E(m_i)$ seien die externen Kosten vom Knoten m_i, also die Summe aller Kosten der Kanten zwischen m_i und den Knoten in Block 1.
  - $D(m_i) = E(m_i) – I(m_i)$
    ist die Differenz der externen und internen Kantengewichte.
- Werden zwei Knoten m_i und m_j betrachtet, so ist die Kostenfunktion
  - $D = D(m_i) + D(m_j) - 2\ c_{ij}$
    $= E(m_i) + E(m_j) – I(m_i) –I(m_j) - 2\ c_{ij}$

    hierbei sind $c_{i,j}$ die Kosten der Kante zwischen mi und mj.
- Bei positiver Kostenfunktion wird getauscht.
- Remark zu c_{ij}: *Die direkte Verbindung der beiden Knoten würde zweimal gezählt und verbindet auch weiterhin die beiden Knoten.*

Beispiel: Random Interchange

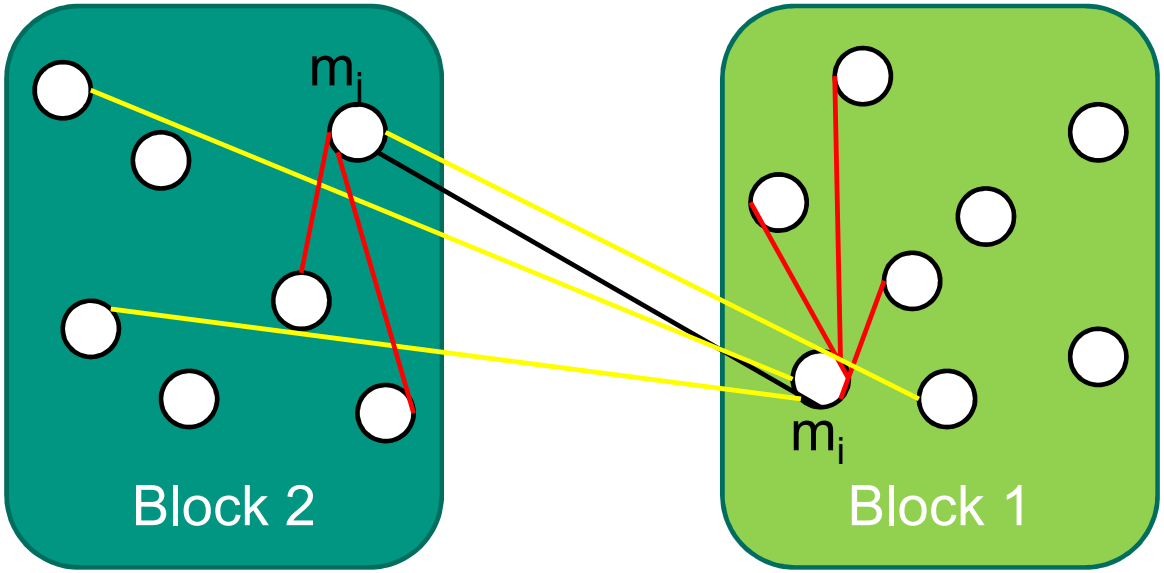


Beispiel: Random Interchange



- Knoten m_i: innere: 2 externe: 4 c_{ij}: 1
- Knoten m_j: innere: 1 externe: 3 c_{ij}: 1
- Bewertungsfunktion: $D = E(m_i) + E(m_j) - I(m_i) - I(m_j) - 2\,c_{ij} = 4+3-2-1-1-1$
- $D = 2 > 0$:
  - Verbesserung → also Austausch von m_i und m_j

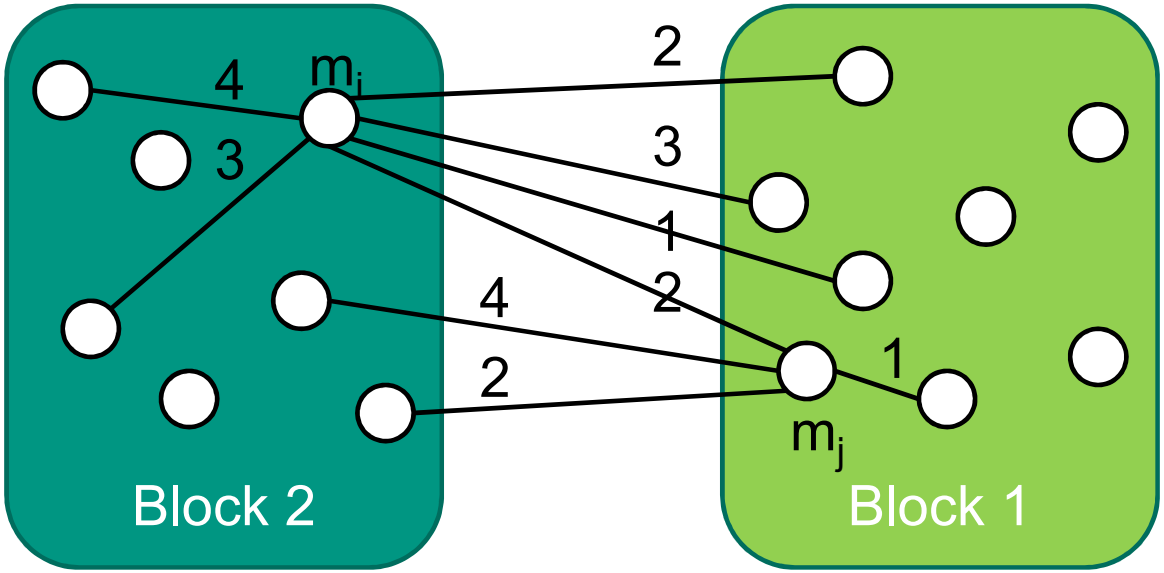
Beispiel: Random Interchange



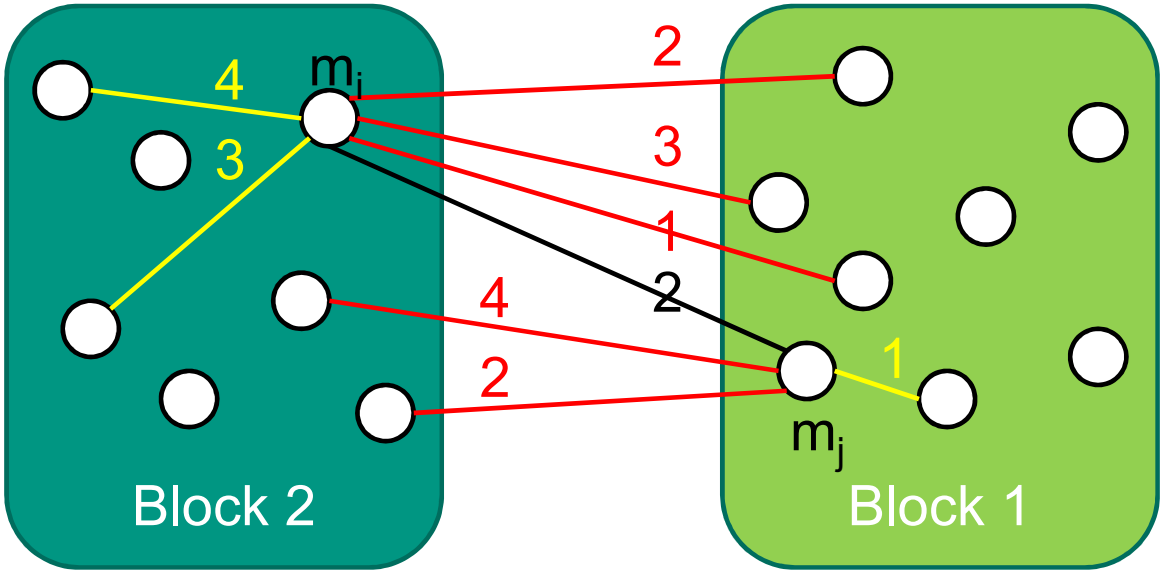
- Knoten m_i: innere: 2 externe: 4 c_{ij}: 1
- Knoten m_j: innere: 1 externe: 3 c_{ij}: 1
- Bewertungsfunktion: $D = E(m_i) + E(m_j) - I(m_i) - I(m_j) - 2\,c_{ij} = 4+3-2-1-1-1$
- $D = 2 > 0$:
  - Verbesserung → also Austausch von m_i und m_j

Beispiel: Random Interchange

Mit Kantengewichten



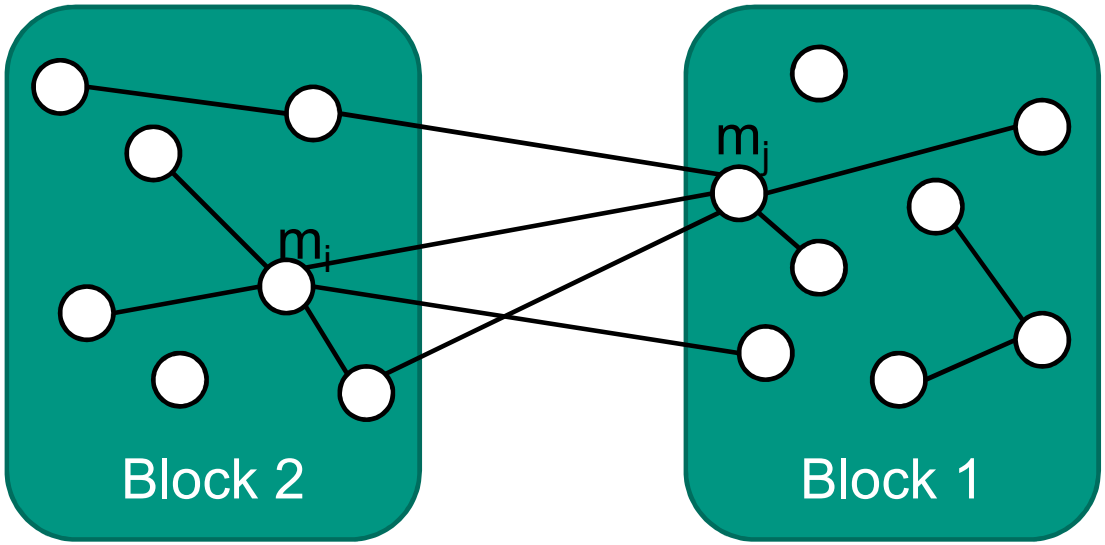
Beispiel: Random Interchange



- Knoten mi: innere: 4+3 externe: 2+3+1+2 cij: 2
- Knoten mj: innere: 1 externe: 2+4+2 cij: 2
- Bewertungsfunktion: D = E(mi) + E(mj) - I(mi) - I(mj) - 2 cij = 8+8-7-1-2-2
- D = 4 > 0 :
  - Verbesserung → also Austausch von mi und mj

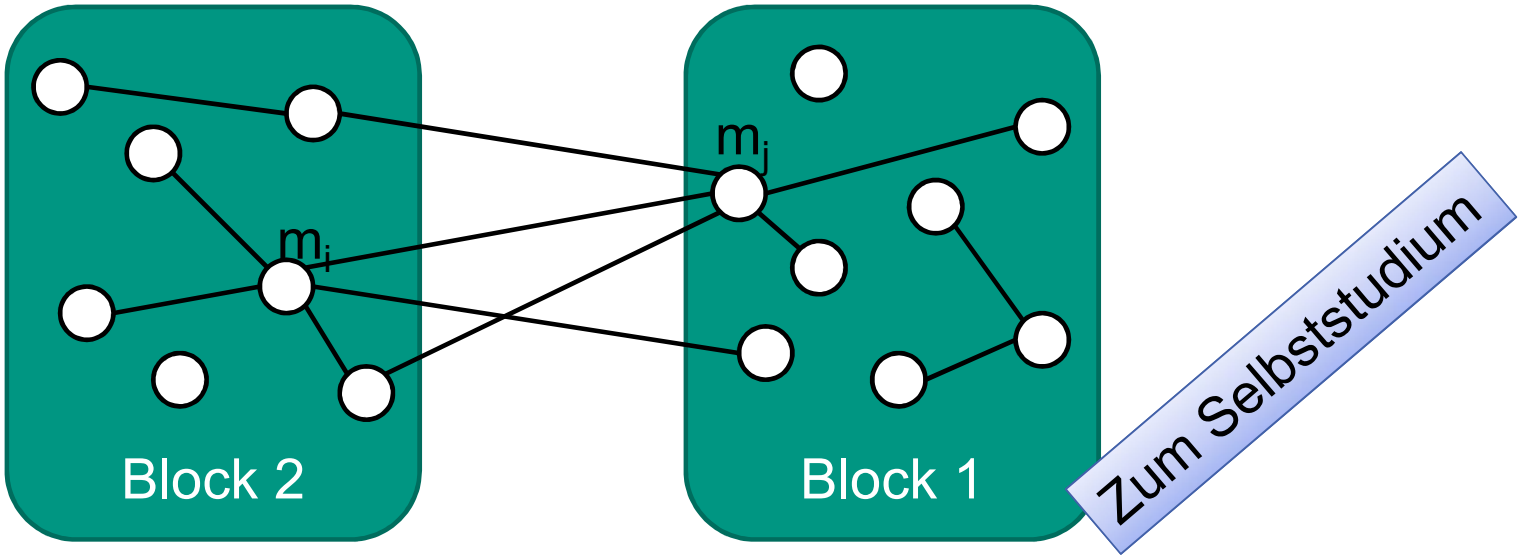
Zwischenübung 01: Random Interchange

- Bringt die Vertauschung von Knoten mi und mj einen Kostenvorteil?



Zwischenübung 01: Random Interchange – Lsg.

- Bringt die Vertauschung von Knoten mi und mj einen Kostenvorteil?



- Knoten mi: innere: 3 externe: 2 cij: 1
- Knoten mj: innere: 2 externe: 3 cij: 1
- Bewertungsfunktion: D = E(mi) + E(mj) - I(mi) - I(mj) – 2 cij = -2
- D = -2 < 0
  - Verschlechterung → kein Austausch von mi und mj

- Random Interchange



Greedy

- Grundprinzip:
  - Finden einer Lösung, indem bei jedem Entscheidungsschritt die lokal optimale Entscheidung getroffen wird.
  - Dadurch findet man für viele Probleme aber nicht zwingend die global optimale Lösung
  - Trotzdem oft verwendet, da bei günstigem Sortierkriterium meist eine hinreichend gute Lösung mit sehr geringem Aufwand gefunden wird
- Eine Anwendung des Greedy-Algorithmus im täglichen Leben ist z.B. die Herausgabe von Wechselgeld.
  - Greedy: *Nimm jeweils immer die größte Münze unter dem Zielwert und ziehe sie von diesem ab.*
  - Verfahre derart bis Zielwert gleich null.
- Beispiel:
  - Zielwert ist 15.
  - Es stehen Münzen mit den Werten 1, 5 und 11 zu Verfügung.
  - 15 = 5 + 5 + 5 ist globales Optimum
  - 15 = 11 + 1 + 1 + 1 + 1 mit Greedy kein globales Optimum

Rucksackproblem

Kerzenständer
Gewicht: 5kg
Nutzen: 20





Radio
Gewicht: 2kg
Nutzen: 60





Wanderkarte
Gewicht: 0,2 kg
Nutzen: 100

Taschenmesser
Gewicht: 0,3kg
Nutzen: 80





Regenjacke:
Gewicht: 3kg
Nutzen: 90

Rucksack
Maximalgewicht: 10kg



Blumentopf
Gewicht: 4kg
Nutzen: 10

Rucksackproblem

- Eingabe:
  - Ein leerer Rucksack mit einer maximalen Kapazität (Maximalgewicht)
  - Eine Auswahl an möglichen Gegenständen, wobei jeder Gegenstand ein Gewicht und einen Nutzen hat.
- Zu lösendes Problem: packe den Rucksack so, …
  - … dass das Gesamtgewicht der eingepackten Gegenstände die Kapazität nicht übersteigt und/oder …
  - … der Nutzen der eingepackten Gegenstände optimal, d.h. maximal, ist.
- Klassisches Optimierungsproblem, auf welches zahlreiche andere Auswahlprobleme abgebildet werden können.
- Komplexitätsklasse $O(2^n)$ (n ist Anzahl der wählbaren Gegenstände)

Rucksackproblem: Lösung mit Greedy-Algorithmus

- Idee:
  - man packe den Rucksack, indem man jeweils den aktuell besten Gegenstand auswählt, z.B. nach folgenden Kriterien:
    - A) den mit dem höchsten Nutzen,
    - B) den mit dem geringsten Gewicht oder
    - C) den mit dem besten Verhältnis von Nutzen und Gewicht.
  - Wiederhole diese Auswahl, bis der Rucksack voll ist bzw. nur noch Gegenstände übrig sind, die nicht mehr in den Rucksack passen



- Greedy



Kernighan-Lin-Algorithmus

- Systematisches Vertauschen aller Knoten der beiden Partitionsblöcke und Erstellen einer Kostenfunktion:

1. – N. Schritt

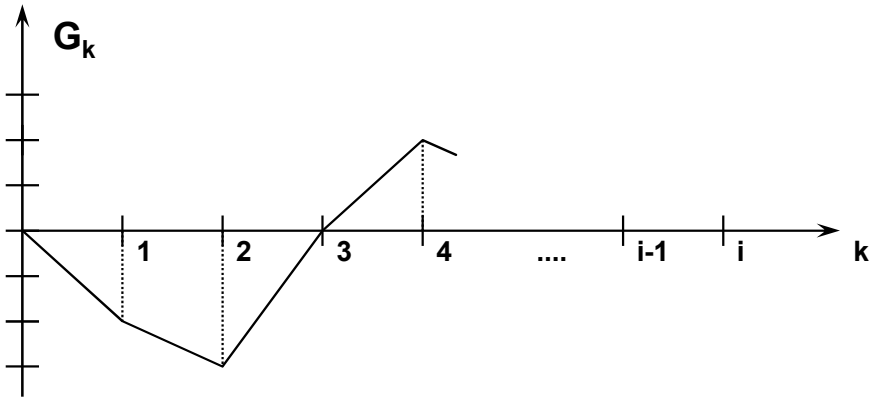
- Auswahl des Knotenpaars mit der besten Kostenfunktion, welches noch nicht getauscht wurde
- Vertauschen der beiden Knoten, trotz eventueller Verschlechterung der Gesamtkosten

Abbruch

- Wenn alle Knoten getauscht wurden, die Knoten also alle den Block gewechselt haben
- Auswahl der kostengünstigsten Konfiguration

Kernighan-Lin-Algorithmus

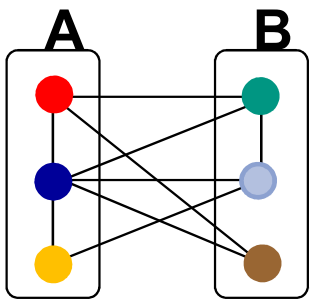
- Gleiche Bewertungsfunktion wie bei Random Interchange:
  - $D(m_i, m_j) = E(m_i) + E(m_j) - I(m_i) - I(m_j) - 2\, c_{ij}$
- Verlauf der Bewertungsfunktion über der Anzahl der Vertauschungen k, mit D_i = Gewinn beim i-ten Austausch
  - Hohe Kosten (positiver *D*-Wert) geben damit einen bestimmten Drang an, die Teilmenge zu wechseln, während niedrige Kosten (negativer *D*-Wert) ein gewisses „Bleiberecht“ verkörpern.
- Der ausgewählte Satz von Vertauschungen korrespondiert mit dem maximalen Gewinn [max G_k]
- Beispiel:



$$G_k = \sum_{i=1}^{k} D_i$$

Beispiel: Kernighan-Lin

1.



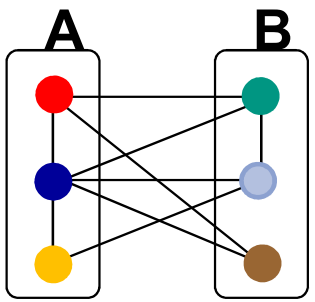
A \ B	1	2	3
1	0	2	1
2	0	0	1
3	1	-1	2

2.

- $D(m_i, m_j) = E(m_i) + E(m_j) - I(m_i) - I(m_j) - 2\, c_{ij}$

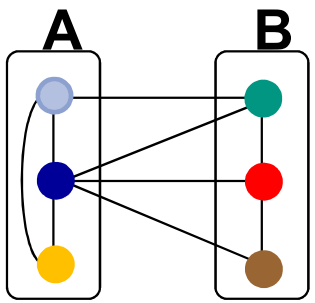
Beispiel: Kernighan-Lin

1.



A \ B	1	2	3
1	0	**2**	1
2	0	0	1
3	1	-1	2

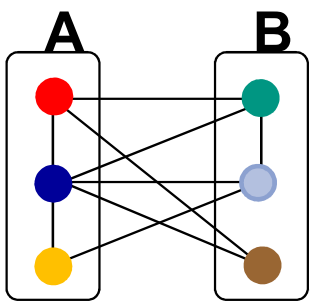
2.



A \ B	1	2	3
1	-	-	-
2	**0**	-	-1
3	-1	-	-2

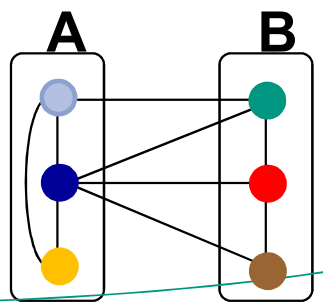
Beispiel: Kernighan-Lin

1.



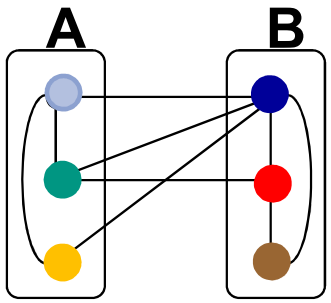
A \ B	1	2	3
1	0	**2**	1
2	0	0	1
3	1	-1	2

2.



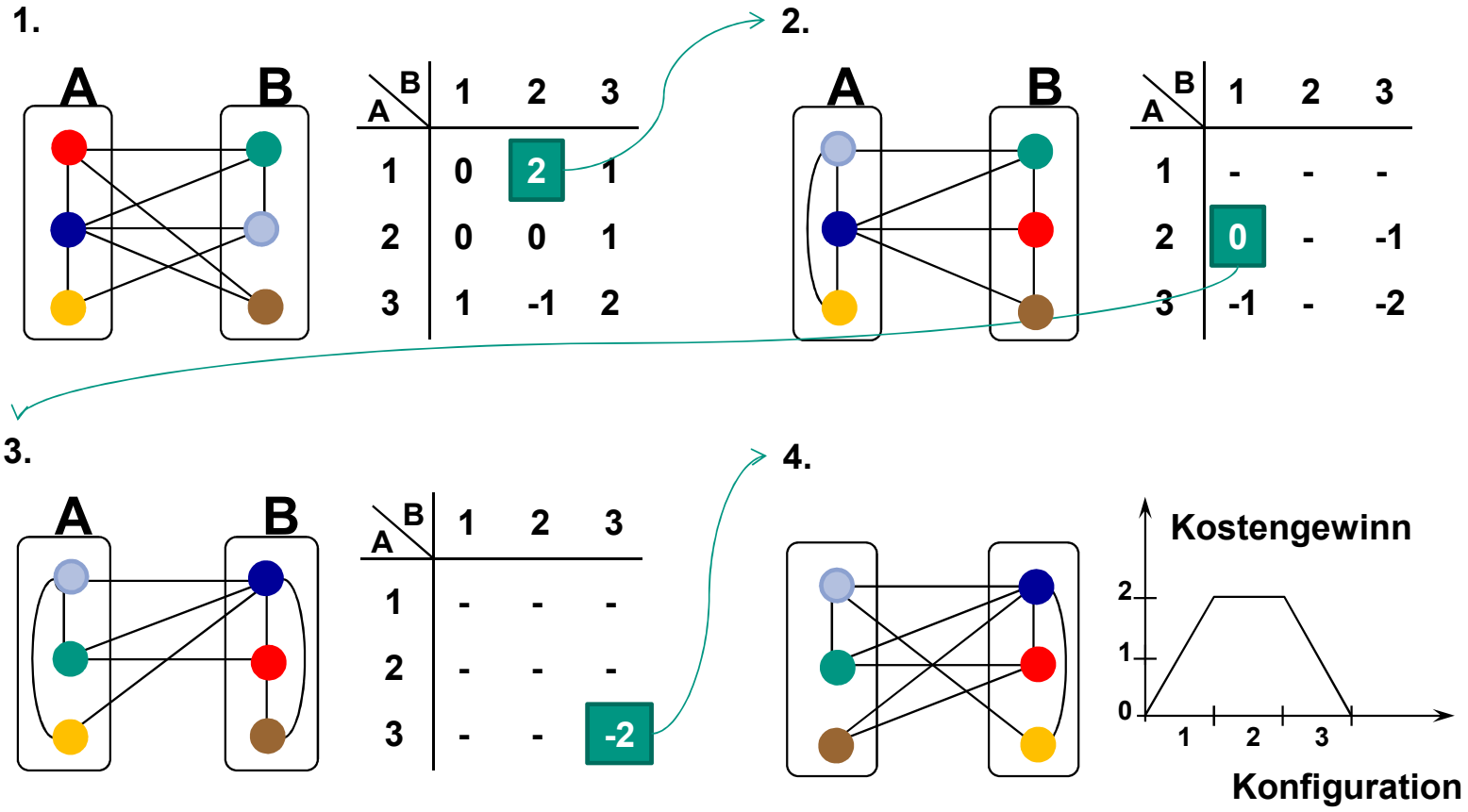
A \ B	1	2	3
1	-	-	-
2	**0**	-	-1
3	-1	-	-2

3.



A \ B	1	2	3
1	-	-	-
2	-	-	-
3	-	-	**-2**

Beispiel: Kernighan-Lin



- Kernighan-Lin-Algorithmus



Simulated Annealing (*simulierte Abkühlung*)

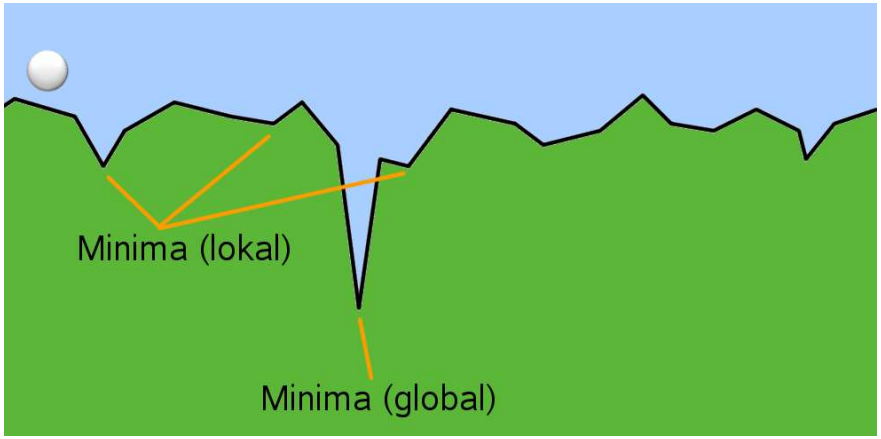
- Das Verfahren wird zum Auffinden einer approximativen Lösung von Optimierungsproblemen eingesetzt, die durch ihre hohe Komplexität das vollständige Ausprobieren aller Möglichkeiten und einfache mathematische Verfahren ausschließen.
- Benutzt wird dieses Verfahren zum Beispiel beim *Floorplanning* im Laufe eines Chipentwurfs.
- Grundidee ist die Nachbildung eines Abkühlungsprozesses, etwa beim Glühen in der Werkstoffkunde.
  - Nach Erhitzen eines Metalls sorgt die langsame Abkühlung dafür, dass die Atome ausreichend Zeit haben, sich zu ordnen und stabile Kristalle zu bilden.
- Übertragen auf das Optimierungsverfahren entspricht die Temperatur einer Wahrscheinlichkeit, mit der sich ein Zwischenergebnis der Optimierung auch verschlechtern darf.
  - Bzw. mit abnehmender Temperatur verringert sich der Lösungsraum.

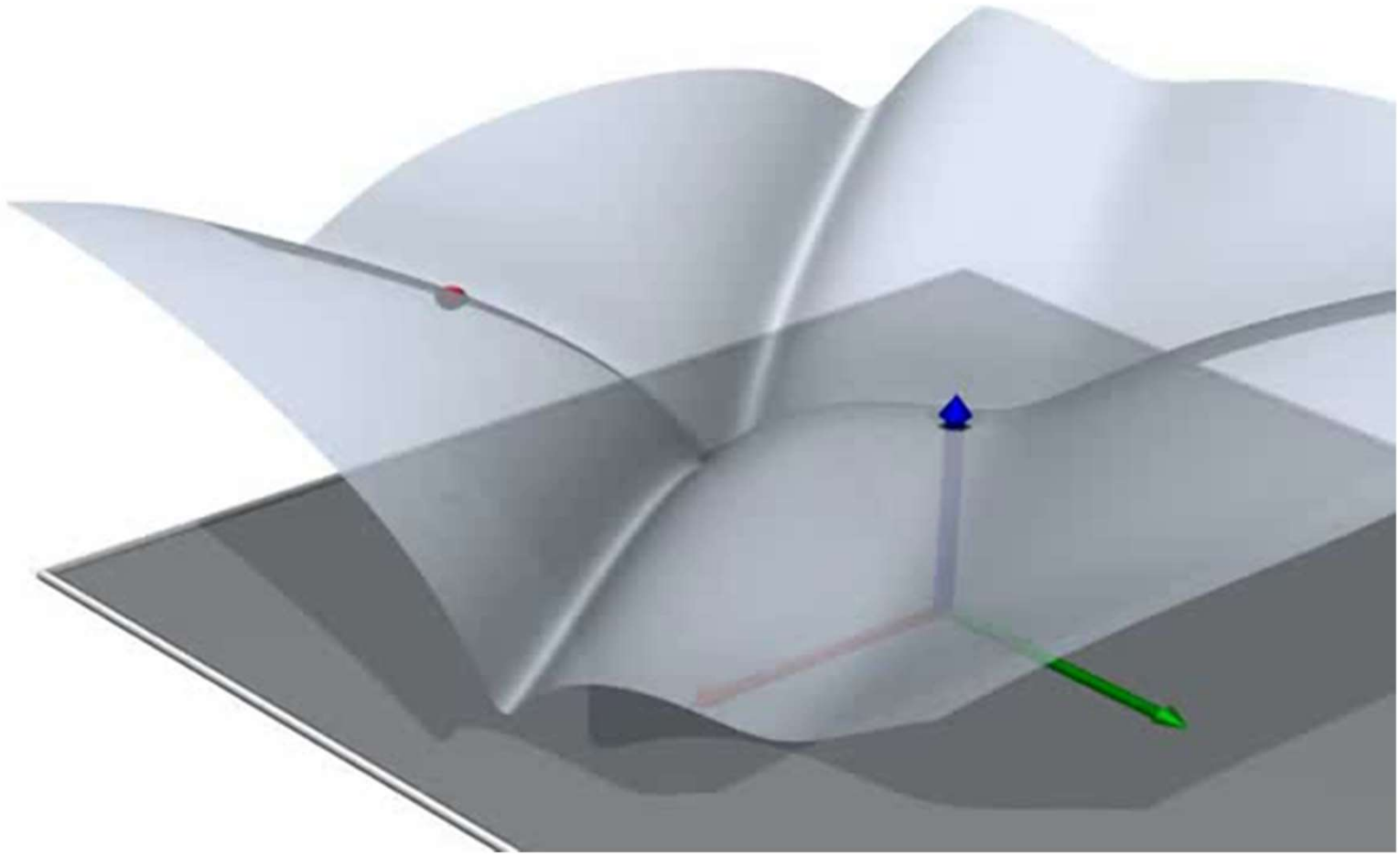
Analogie

Das simulierte Ausglühen kann man sich wie folgt verdeutlichen.

- Man sucht in einer Landschaft den (global) tiefsten Punkt.
- Die Landschaft selbst besteht aus vielen unterschiedlich tiefen Dellen.
- Die einfache Suchstrategie entspricht dem Verhalten einer Kugel, welche in dieser Landschaft ausgesetzt wird.
  - Sie rollt zum nächsten lokalen Minimum und bleibt dort.
- Bei der simulierten Abkühlung wird der Kugel immer wieder ein Stoß versetzt, der mit zunehmender „Abkühlung“ schwächer wird.

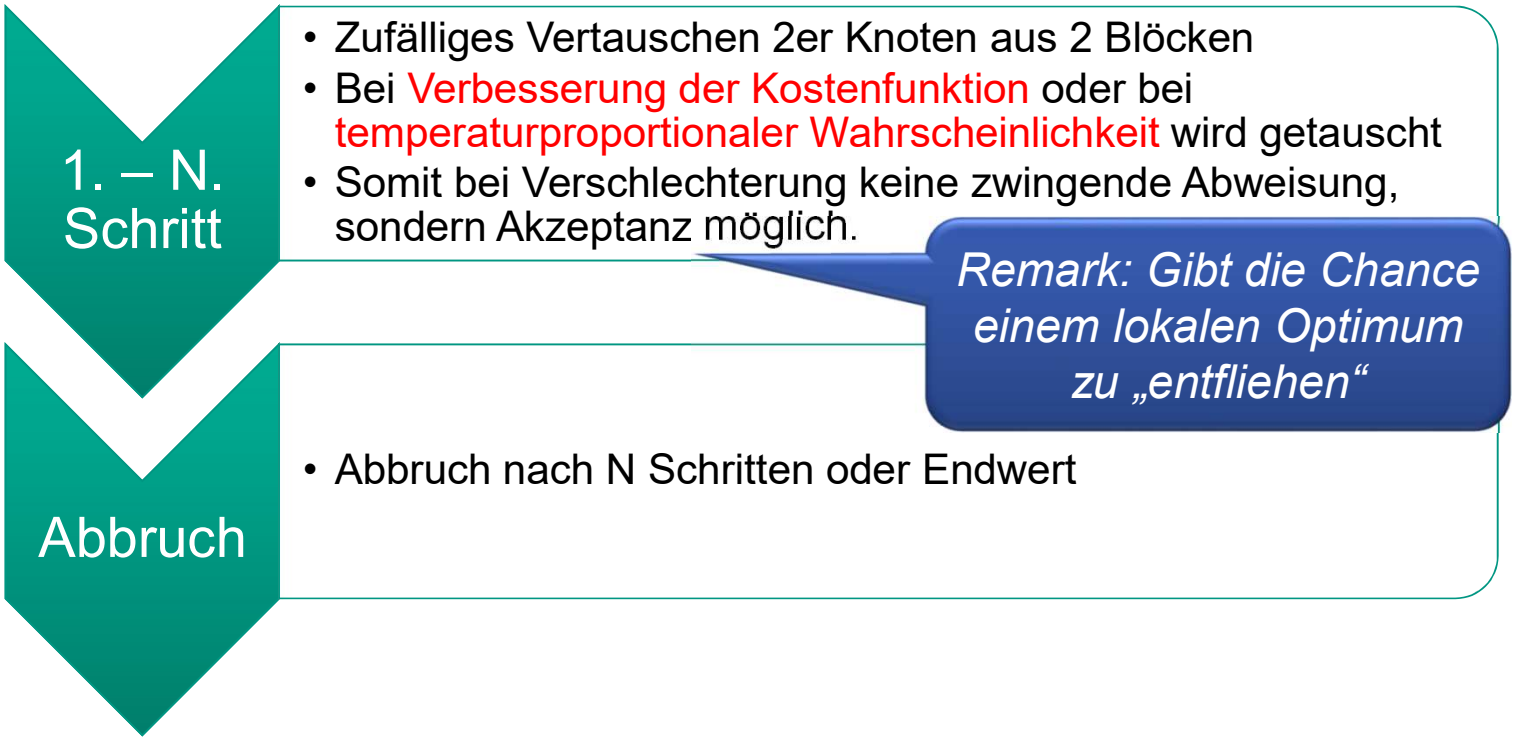
- Dieser ist idealerweise stark genug, um die Kugel aus einer flachen Delle (lokales Minimum) zu entfernen, reicht aber nicht aus, um aus dem globalen Minimum zu fliehen.





Simulated Annealing (*simulierte Abkühlung*)

- Verbesserung des Random Interchange-Algorithmus



Pseudo Code: Simulated Annealing

```
simulate_annealing() {
  T = T0;                                          //Anfangswert der Temperatur
  X = j0;                                          //Anfangskonfiguration
  while( !stop_criterion ) {                       //Endtemperatur, oder weniger als Min Austauschungen

    while( !loop_criterion ) {                     //Mindestanzahl Versuche bei einer Temperatur

      j = generate( X );                           //Generiere neue Konfiguration, also Austausch zweier Module

      if( accept( cost( j ), cost( X ), T ) )      //spezielle Akzeptanzfunktion
        X = j;                                     //akzeptierte neue Konfiguration abspeichern
    }
    T = update( T );                               //langsames Senken der Temp. (z.B. T = 0.9T)
   }
}
```

Pseudo Code: Simulated Annealing

```
simulate_annealing() {
  T = T0;                                   //Anfangswert der Temperatur
  X = j0;                                   //Anfangskonfiguration
  while( !stop_criterion ) {                //Endtemperatur, oder weniger als Min Austauschungen

    while( !loop_criterion ) {              //Mindestanzahl Versuche bei einer Temperatur

      j = generate( X );                    //Generiere neue Konfiguration, also Austausch zweier Module

      if( accept( cost( j ), cost( X ), T ) )   //spezielle Akzeptanzfunktion
        X = j;                              //akzeptierte neue Konfiguration abspeichern
    }
    T = update( T );                        //langsames Senken der Temp. (z.B. T = 0.9T)
   }
}
accept( cost( j ), cost( i ), T ) {         //liefert eine 1, wenn ein Austausch akzeptiert wird,
                                            //wenn nicht, eine 0
  diff_cost = cost( j ) - cost( i );        //Differenz der Kosten berechnen
  y = exp( -diff_cost / T );                //Wahrscheinlichkeitsfunktion nach Boltzmannstatistik
                                            //je höher die Temperatur, desto größer wird y für diff_cost>0
  r = random;                               //Zufallszahl zwischen 0 und 1
                                            // → den lokalen Optima entfliehen
if( r < y )                                 //bei höheren Temperaturen mit höherer Wahrscheinlichkeit
    return 1;                               //akzeptiert,
else
    return 0;                               //oder nicht
}
```

temperaturproportionale Wahrscheinlichkeit

Simulated Annealing Akzeptanz

```
simulate_annealing() {
  T = T0;
  X = j0;
  while( !stop_criterion ) {

    while( !loop_criterion ) {

      j = generate( X );

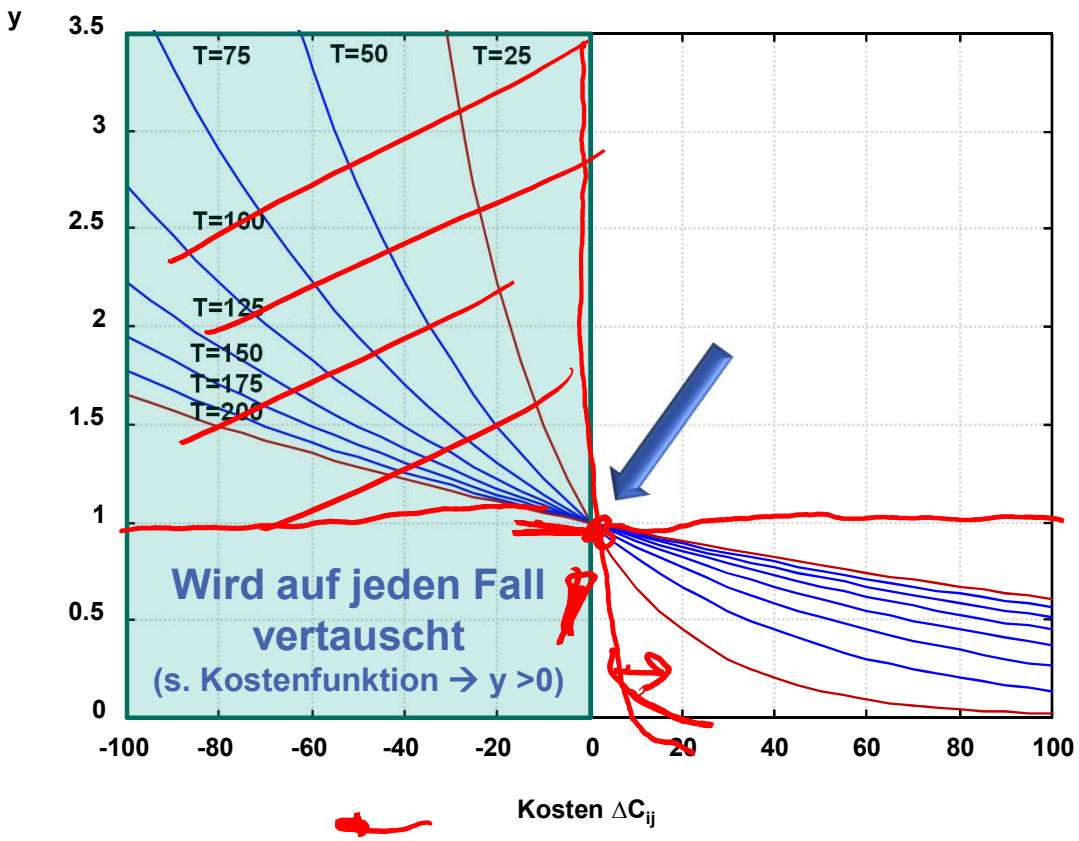
      if( accept( cost( j ), cost( X ), T ) )
        X = j;
    }
    T = update( T );
   }
}
accept( cost( j ), cost( i ), T ) {

  diff_cost = cost( j ) - cost( i );
  y = exp( -diff_cost / T );

  r = random;

if( r < y )
    return 1;
else
    return 0;
}
```

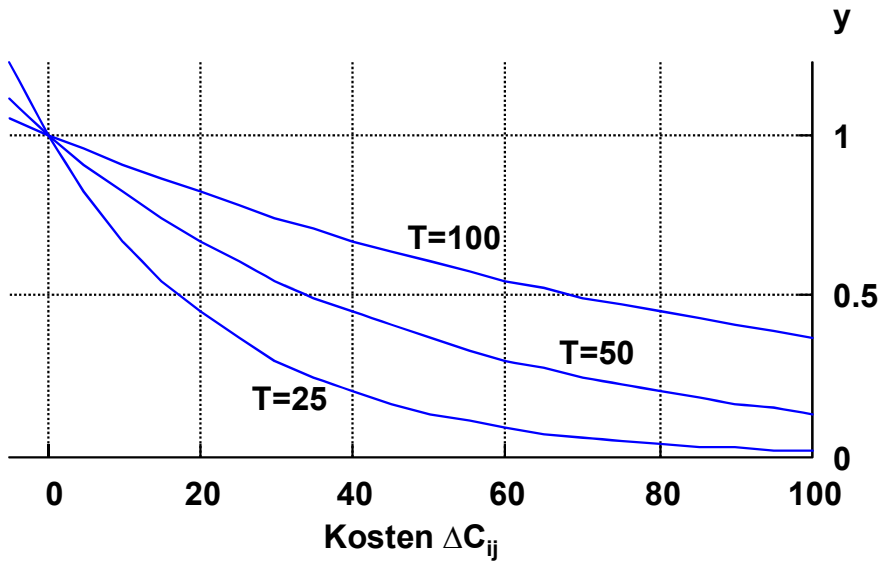
temperaturproportionale Wahrscheinlichkeit




```
y = exp( -diff_cost / T );
```

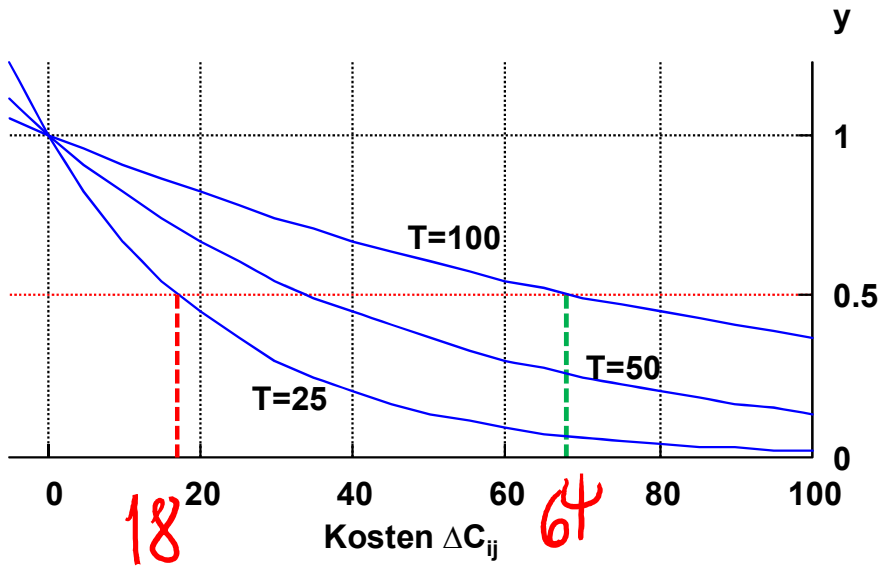
Übung: Simulated Annealing

- Sie sehen die grafische Darstellung für den Fall einer Kostenverschlechterung bei verschiedenen Temperaturen. Beantworten Sie die folgenden Fragen:
- Welche Kostendifferenz wird für T=25 mit einer Wahrscheinlichkeit von 50% akzeptiert?
- Wie groß darf die Kostendifferenz für dieselbe Wahrscheinlichkeit bei T=100 maximal sein?



Übung: Simulated Annealing – Lsg.

- Sie sehen die grafische Darstellung für den Fall einer Kostenverschlechterung bei verschiedenen Temperaturen. Beantworten Sie die folgenden Fragen:
- Welche Kostendifferenz wird für T=25 mit einer Wahrscheinlichkeit von 50% akzeptiert?
  Abgelesen: $\Delta C_{ij} = 18$
- Wie groß darf die Kostendifferenz für dieselbe Wahrscheinlichkeit bei T=100 maximal sein?
  Abgelesen: $\Delta C_{ij} = 64$



- Simulated Annealing



TSP


Traveling Salesman Problem
Discover the shortest route for visiting a group of cities

- Optimierungsalgorithmen




Backup
Shift
Insert

C++ Lösung

```
class Gegenstand {

   public :
      Gegenstand() {
         gewicht = 1+rand()%MAX_GEWICHT;
         nutzen = 1+rand()%MAX_NUTZEN;
      };
      int gewicht;
      int nutzen;
};
```

```
class Rucksack {

   public :
      int gesamtnutzen();
      int gesamtgewicht();
      bool einpacken(Gegenstand*);
      void ausgabe();
   private :
      set<Gegenstand*> inhalt;
};
```

Pseudo Code

- Sortierung der Gegenstände über Vergleichsfunktion
- 1. Alternative: Sortierung absteigend nach Nutzen

```
bool vergleichsfunktion(Gegenstand* g1, Gegenstand* g2) {
   if (g1->nutzen > g2->nutzen) return true;
   if (g1->nutzen == g2->nutzen && g1->gewicht < g2->gewicht)
         return true;
   return false;
}
```

- 2. Alternative: Sortierung nach Verhältnis Nutzen/Gewicht liefert bessere Ergebnisse

```
bool vergleichsfunktion2(Gegenstand* g1, Gegenstand* g2) {
   if ((float)g1->nutzen/g1->gewicht >
      (float)g2->nutzen/g2->gewicht) return true;
   return false;
}
```

Greedy Algorithmus

- Sortieren der Liste nach Greedy-Präferenz (Vergleichsfunktion)
- Dann entsprechend dieser Reihenfolge „in den Rucksack stopfen“

```
Rucksack packenGreedy(list<Gegenstand*> auswahl) {

   Rucksack rs;
   auswahl.sort(vergleichsfunktion);
   list<Gegenstand*>::const_iterator pos;
   for (pos = auswahl.begin();
         rs.gesamtgewicht() < KAPAZITAET
         && pos != auswahl.end(); pos++) {
      Gegenstand* g = *pos;
      rs.einpacken(g);
   }
   return rs;
}
```

Auswahl nach Gewicht

```
static Rucksack packeGierigNachGewicht() {
        Rucksack r = new Rucksack();
        while (true) {
                int pos=-1; int bestesGewicht = MAX_GEWICHT+1;
                for (int i=0; i<auswahlObjekte.length; i++)
                        if (r.auswahl[i] == false &&
                                auswahlObjekte[i].gewicht < bestesGewicht &&
                                r.gewicht() + auswahlObjekte[i].gewicht <=
                                        kapazitaet) {
                           bestesGewicht = auswahlObjekte[i].gewicht;
                           pos = i;
                        }
                if (pos == -1) break;
                else r.auswahl[pos] = true;
        }
        return r;
}
```

Auswahl nach Nutzen

```
static Rucksack packeGierigNachNutzen() {
        Rucksack r = new Rucksack();
        while (true) {
                int pos=-1; int besterNutzen = 0;
                for (int i=0; i<auswahlObjekte.length; i++)
                        if (r.auswahl[i] == false &&
                                auswahlObjekte[i].nutzen > besterNutzen &&
                                r.gewicht() + auswahlObjekte[i].gewicht <=
                                        kapazitaet) {
                           besterNutzen = auswahlObjekte[i].nutzen;
                           pos = i;
                        }
                if (pos == -1) break;
                else r.auswahl[pos] = true;
        }
        return r;
}
```

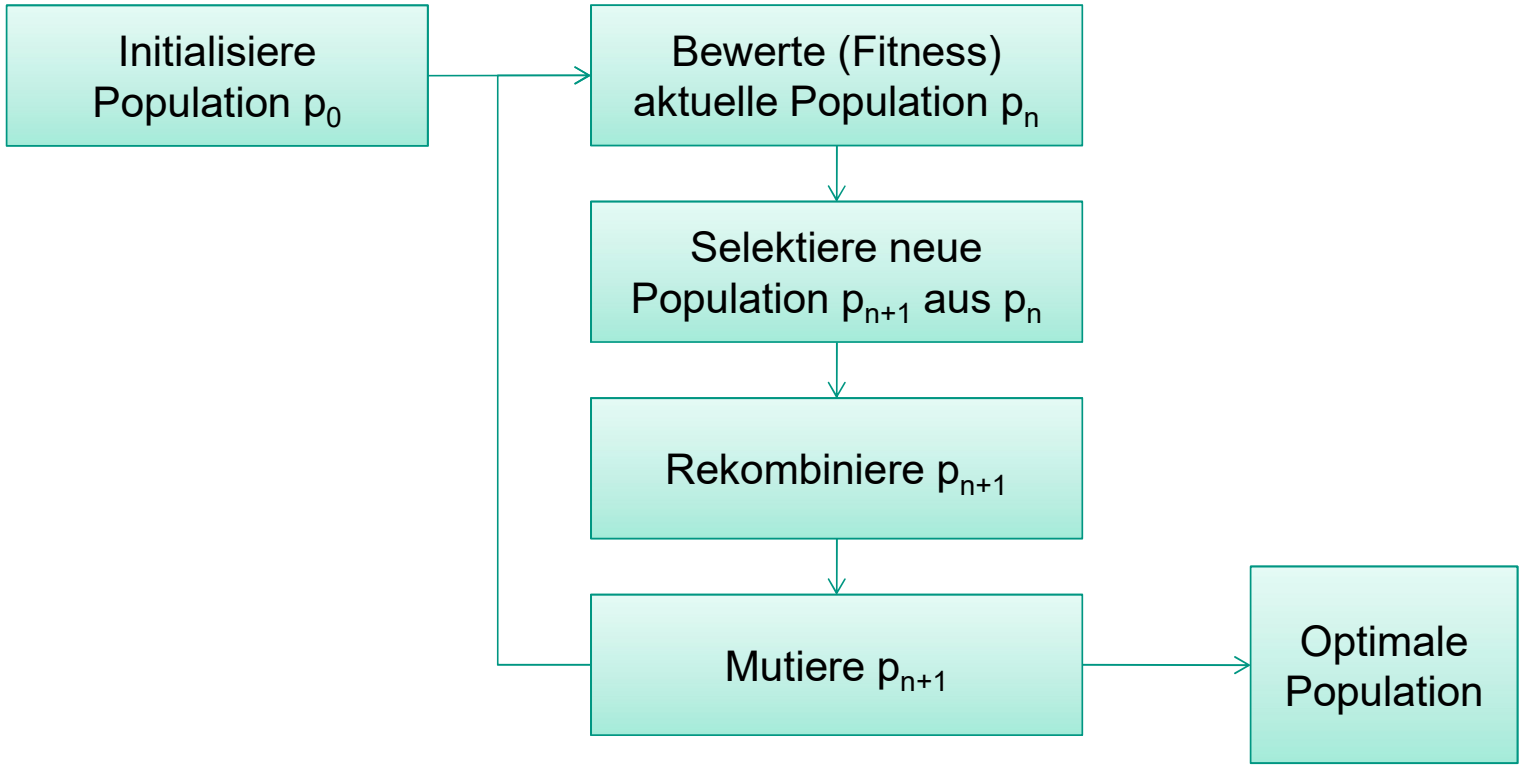
Merkmale Evolutionäre Algorithmen

- „Die Natur ist der beste Baumeister“
- Evolutionäre Algorithmen (EA) versuchen nun, die Prinzipien der natürlichen Evolution im Computer nachzubilden, um damit Problemlösungen quasi zu züchten anstatt sie analytisch zu berechnen.
- Entsprechend dem natürlichen Vorbild stellen sie kaum Voraussetzungen an den Suchraum.
  - Die Qualitätsfunktion muss weder stetig noch differenzierbar sein
  - Es wird lediglich gefordert, dass sie so beschaffen ist, dass sie eine Suche in irgendeiner Weise zum Ziel führen kann.
- Es findet keine Verbesserung von Punkt zu Punkt, sondern von Population zu Population statt, d.h. es wird stets aus einer Menge (Population) von Konfigurationen (sog. Individuen) eine neue Population (sog. Generation) erzeugt.
- Die Übergangsregeln sind probabilistisch, nicht deterministisch.
- Der Übergang von einer Population zur nächsten, d.h. die Erzeugung einer neuen Generation, erfolgt dabei mit Hilfe von Operatoren. Die Veränderungen finden dabei stets anhand der Codierungen statt.

Operatoren Evolutionärer Algorithmen

- Selektion (Reproduktion)
  - ungeschlechtliche Fortpflanzung: Hierbei werden identische Nachkommen (Clone) gebildet. Je besser die Zielfunktion (engl. fitness) eines Individuums, um so häufiger kann es seine Erbmasse, d.h. sich selbst reproduzieren. Dieses Prinzip entspricht dem Prinzip des Überlebens des Besten (engl. survival of the fittest).
- Rekombination (Kreuzung)
  - Geschlechtliche Fortpflanzung: Zwei Individuen vermischen ihre Erbanlagen und “zeugen” zwei Nachkommen, die die Position ihrer Eltern in der Population übernehmen, d.h. Nachkommen ersetzen ihre Vorfahren.
- Mutation
  - Zufällige Änderungen der Erbmasse: Dieser Operator ist von untergeordneter Bedeutung, kann bei bestimmten Problemen jedoch zur Überwindung lokaler Optima hilfreich sein.

Evolutionäre Algorithmen



Verdrahtung

- *Aufgabe: „Bestimmung der geometrischen Strukturen für die Verbindungen zwischen den Zellen gemäß der Netzliste unter Berücksichtigung technologie- und schaltungsspezifischer Designregeln“*

- Technologieabhängigkeit:
  - Standard- und Makrozellen
    - Verdrahtungsfläche in einzelne Verdrahtungskanäle aufteilen
    - Globale Wegesuche
    - Festlegen der Verdrahtungsreihenfolge der Kanäle
    - Erzeugen der Endverdrahtung für jeden Kanal
  - Moderne Gate Arrays (Sea of Gates)
    - Aufteilung in Blöcke
    - Block für Block behandeln

Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

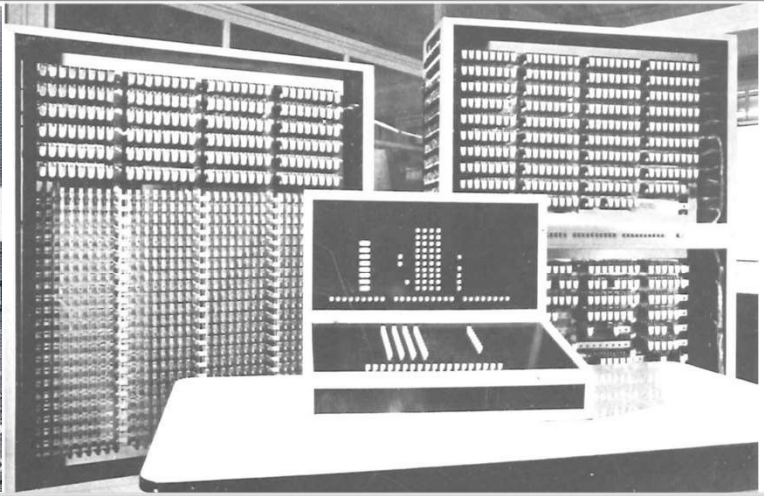
Institutsleitung
Prof. Dr.-Ing. J. Becker
Prof. Dr.-Ing. E. Sax
Prof. Dr. rer. nat. W. Stork

6.1 Big Data & Maschinelles Lernen (A)

Informationstechnik II

Prof. Dr.-Ing. Eric Sax







Inhalt IT2

6. Big Data & Maschinelles Lernen

- Definitionen, Verwendung (Charakteristik, Risiken, Chancen)
- Motivation und Anwendungsfälle
- Data Science Prozesse:
  - KDD
  - CRISP-DM
- CRISP-DM im Detail
  - Business Understanding
  - Data Understanding
  - Data Preparation
  - Modeling
  - Evaluation
- Infrastruktur für Big Data



Literatur

- http://crisp-dm.eu/
- Planning for Big Data. A CIO‘s Handbook to the Changing Data Landscape – O‘Reilly
  online-PDF: https://eecs.wsu.edu/~yinghui/mat/courses/fall%202015/resources/planning-for-big-data.pdf
- Big Data Insider: https://www.bigdata-insider.de/was-ist-big-data-a-562440/
  generell sehr informativ über Big Data, man kann sich durch verschiedene aufeinander abgestimmte Artikel klicken
- Doing Data Science – Rachel Schutt, Cathy O‘Neil
- *Industriebeispiele aus:* Big Data – Systeme und Prüfung
  978-3503144013
- *KNN:* Klassifikation und Variablenselektion bei höherdimensionalen Daten unter Verwendung von Nearest Neighbor Ensembles – Lena Schober

Big Data
Inhalt der Vorlesungen

- Theorie



- Algorithmen

  Ablaufplan Data Science
  Data Science als Prozess: CRISP-DM
  Algorithmen des Maschinelles Lernens

- Anwendung

  Anwendung von Big Data und Data Mining

Big Data
Motivation

„Big Data ist wenn die Daten selbst ein Teil des Problems werden“

Roger Magoulas O‘Reilly Media

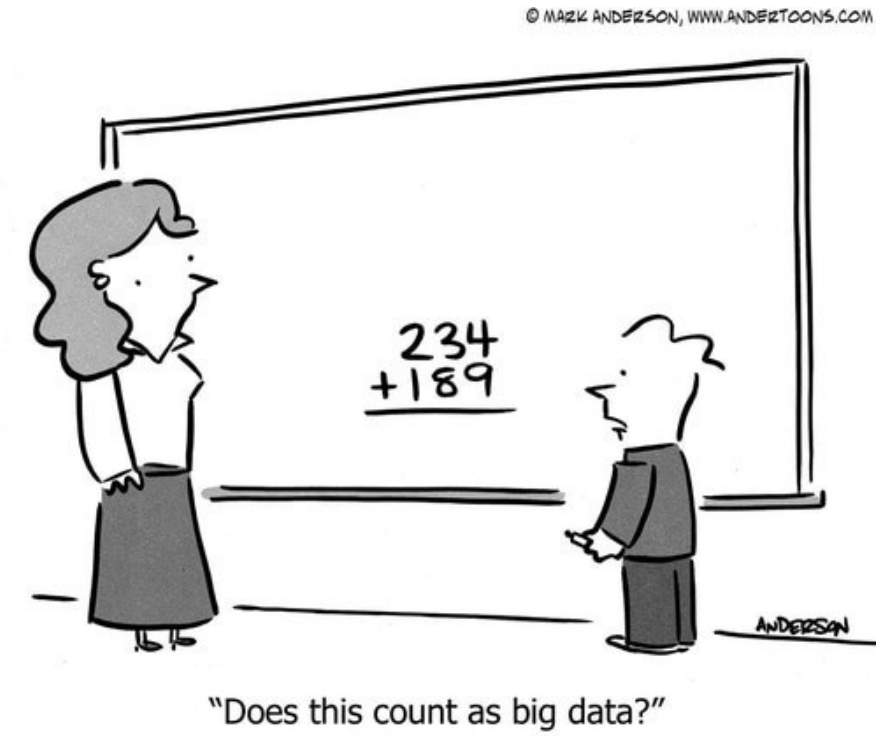
„Big Data ist ein Synonym für die Bedeutung großer Datenvolumen […] sowie der damit verbundenen Herausforderung diese verarbeiten zu können“

Enzyklopädie der Wirtschaftsinformatik



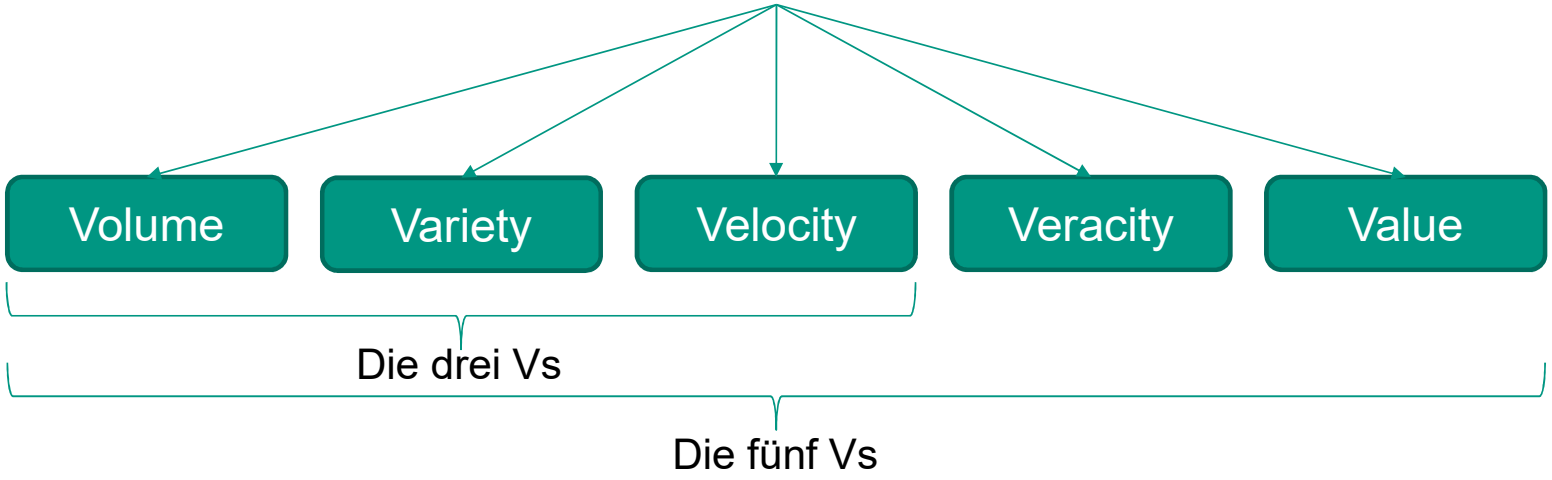
Big Data
Motivation

- Technologien für automatisiertes Generieren von Daten (z.B. durch Sensoren, GPS, RFIDs) nehmen zu.
- Performantere und günstigere Möglichkeiten der Datenübertragung und –speicherung wachsen stetig.
- Digitalisierung verschiedener Prozesse in allen Lebenslagen
  - Einkaufen, Fertigung, Mobilität
- Big Data setzt sich aus vielen Teilaspekten zusammen, dient unterschiedlichen Bewertungen und entwickelt sich stetig weiter.



Big Data – Begriffe: Die fünf Vs

- In Fachliteratur wird Big Data über „die fünf Vs“ definiert



- Definition nach Gartner (2001): Volume, Variety und Velocity
- Erweiterung der Definition durch IBM: drei Vs + Veracity und Value

Die drei grundlegenden Vs

- Volume

  „Data Size“

  Definiert die enorme Menge an Daten, welche es zu speichern gilt.
  Aufkommen ist verflochten und schwer fassbar, so dass die Daten mit herkömmlichen Methoden der Datenverarbeitung nicht mehr speicherbar oder analysierbar sind

- Variety

  „Different forms of data sources“

  Vielfalt der Datentypen und -quellen (Tabellen, Bilder, Videos..)
  Mit Big Data-Suchalgorithmen werden die Informationen strukturiert eingeordnet und Zusammenhänge untersucht

- Velocity

  „Speed of Change“

  Geschwindigkeit mit der Daten generiert, ausgewertet und weiterverarbeitet werden sollen und können

Die zwei entscheidenden zusätzlichen Vs

- Veracity

  „*Uncertainty of data*“

  Richtigkeit, Wahrhaftigkeit, Zuverlässigkeit, Sinnhaftigkeit, Vertrauenswürdigkeit der Daten und -quellen, Sicherstellung der Datenqualität
  Big Data mit allen Qualitätsgraden → Qualitativ schlechte Daten liefern auch mit guter Analysesoftware zu minderwertigen Ergebnissen
  Glaubwürdigkeit der Daten ist dabei ein zunehmendes Problem

- Value

  „business value“

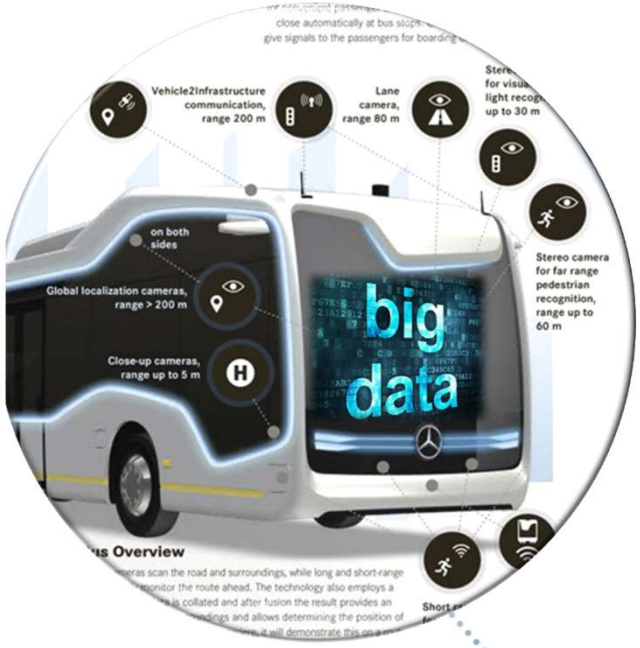
  Unternehmerischer/Wirtschaftlicher Mehrwert der Daten, welcher durch geeignete Analysen gewonnen werden kann

Verbindung der 5 V‘s

„Für viele geht es um unternehmerischen Erfolg (Value) und Gewinnung neuer Informationen - die in kürzester Zeit (Velocity) sehr vielen Nutzern zur Verfügung stehen müssen – mittels enorm großer Datenbestände (Volume) aus unterschiedlichsten Quellen (Variety) und unterschiedlichsten Qualitäten (Validity) um dadurch schneller wettbewerbsfähige Entscheidungen treffen zu können“

Datenquellen und Datenarten

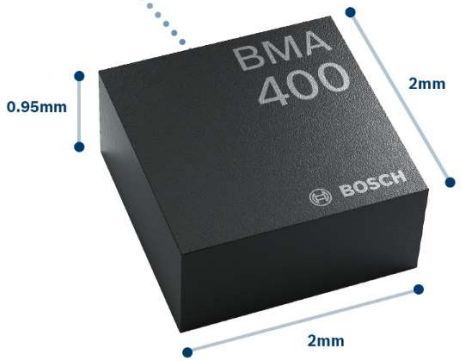
- Daten werden aus unterschiedlichen Quellen gewonnen und liegen daher in unterschiedlichen Formaten vor
  - Näherungssensoren
  - RFID-Chips
  - Lesegeräte
  - Bildsensoren
  - Temperatursensoren
  - LIDAR
  - Einkaufverhalten (Payback, Kreditkarten,...)
  - ….











Datenquellen und Datenarten

- Verschiedene Datenquellen weisen verschiedene Datenformate auf.
- Vielfalt muss differenziert werden
  - ► Daten werden nach weiteren Merkmalen unterschieden
    - Unternehmensinterne und -externe Daten
    - Strukturierte und unstrukturierte Daten
- „80%-Regel“:
  - ► 80% der Daten in einem Unternehmen sind unstrukturierte Daten.



- Strukturierte Daten sind ein wachsender Faktor z.B. in der Welt der Suchmaschinenoptimierung.
  - ► Sie helfen unter anderem, den Web-Crawlern ein besseres Verständnis von Seitenelementen zu geben.
- Die Web-Crawler wissen zum größten Teil bereits sehr gut über die Inhalte auf der Seite Bescheid.
  - ► Wenn es aber um spezifische Details wie Bewertungen, Rezepte oder Veranstaltungen geht, helfen strukturierte Daten den Crawlern aus, um den Web-Daten mehr Bedeutung bzw. Relevanz zu geben.

Was können wir mit „Big Data“ machen?



- Wie funktioniert die intelligente Informationsverarbeitung?
  - Alternative 1: Hypothesen zur Prüfung anhand Daten aufstellen
  - Alternative 2: Mathematische Abbildungen (Signale und Systeme)
  - Alternative 3: Data Mining

Hypothesengetriebene Abfrage vs. Data Mining

- Data Mining bildet eine Ergänzung zu traditionellen, hypothesen-getriebenen Abfragemethoden.
- Data Mining bietet Informationen ohne dass vorher präzise (quantitative) Fragen gestellt werden müssen.

spezifisch

Hypothesen-getriebene Abfrage	Data Mining
Wieviele Kunden kauften im 1. Halbjahr 2018 Bier und wie hoch war deren Umsatz?	Welche auffälligen Kaufmuster gibt es im Bereich Getränke und wie kann man diese nutzen?
Wie hoch waren die Kosten für Garantiefälle im Bereich High Definition Television in Süddeutschland in der letzten drei Monaten?	Welche Bauteile/Funktionen sind kritisch für die Qualität von High Definition Televisions und wie kann man diese proaktiv erkennen?
Wieviele illegale Insidergeschäfte bei Aktientransaktionen wurden im letzten Kalenderjahr bei Kontrollen entdeckt?	Gibt es auffällige Aktientransaktionen die auf mögliche Insidergeschäfte hindeuten?

offen

Hypothesengetriebene Abfrage vs. Data Mining

- Data Mining bildet eine Ergänzung zu traditionellen, hypothesen-getriebenen Abfragemethoden.
- Data Mining bietet Informationen ohne dass vorher präzise (quantitative) Fragen gestellt werden müssen.

spezifisch

Hypothesen-getriebene Abfrage	Data Mining
Wieviele Kunden kauften im 1. Halbjahr 2018 Bier und wie hoch war deren Umsatz?	Welche auffälligen Kaufmuster gibt es im Bereich Getränke und wie kann man diese nutzen?
Wie hoch waren die Kosten für Garantiefälle im Bereich High Definition Television in Süddeutschland in der letzten drei Monaten?	Welche Bauteile/Funktionen sind kritisch für die Qualität von High Definition Televisions und wie kann man diese proaktiv erkennen?
Wieviele illegale Insidergeschäfte bei Aktientransaktionen wurden im letzten Kalenderjahr bei Kontrollen entdeckt?	Gibt es auffällige Aktientransaktionen die auf mögliche Insidergeschäfte hindeuten?

offen

Data Mining und maschinelles Lernen

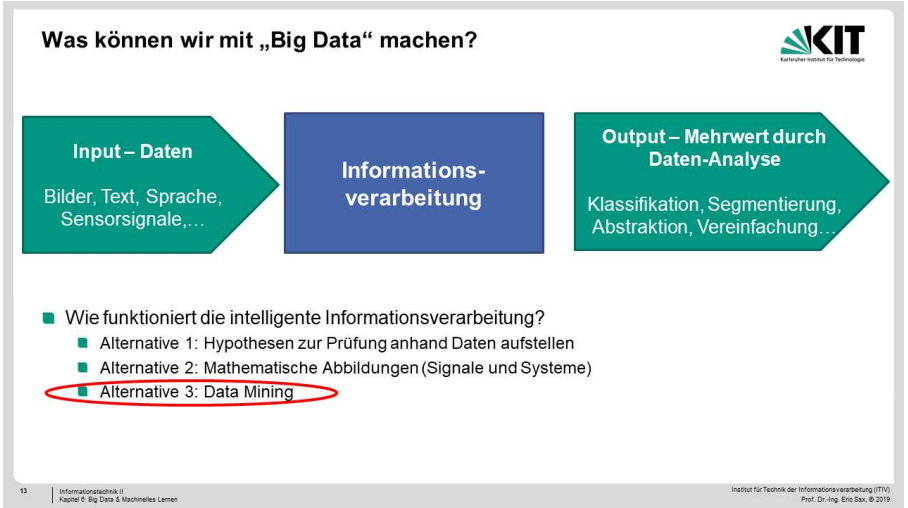
- Datenbestände wachsen stetig und enthalten eine Fülle von ungenutzten, aber eventuell wertvollen Informationen.



- *„Als Data Mining bezeichnet man Verfahren zur Analyse großer Datenbestände, die aus der unübersehbaren Fülle von Details bisher unbekannte Strukturen und Zusammenhänge heraus filtern und diese Informationen so aufbereiten und bewerten, dass sie eine verständliche Entscheidungshilfe darstellen“*

Data Mining als Informationsverarbeitung

- Suchalgorithmen liefern Informationen aus Daten
- Von Hand aufgestellte Regeln liefern Klassifikation
- Mathematische Modelle vereinfachen Daten
- Aber:
  - Raum der Modelle ist unendlich groß
  - Sehr großer Aufwand, Anpassung eines Modells „von Hand“ vorzunehmen
  - Big Data ist „zu groß“ damit ein Mensch darin eine Struktur finden kann
  - ► Sinnvolle „Suche“ nach passendem Modell / Muster, oder interessanten Daten notwendig

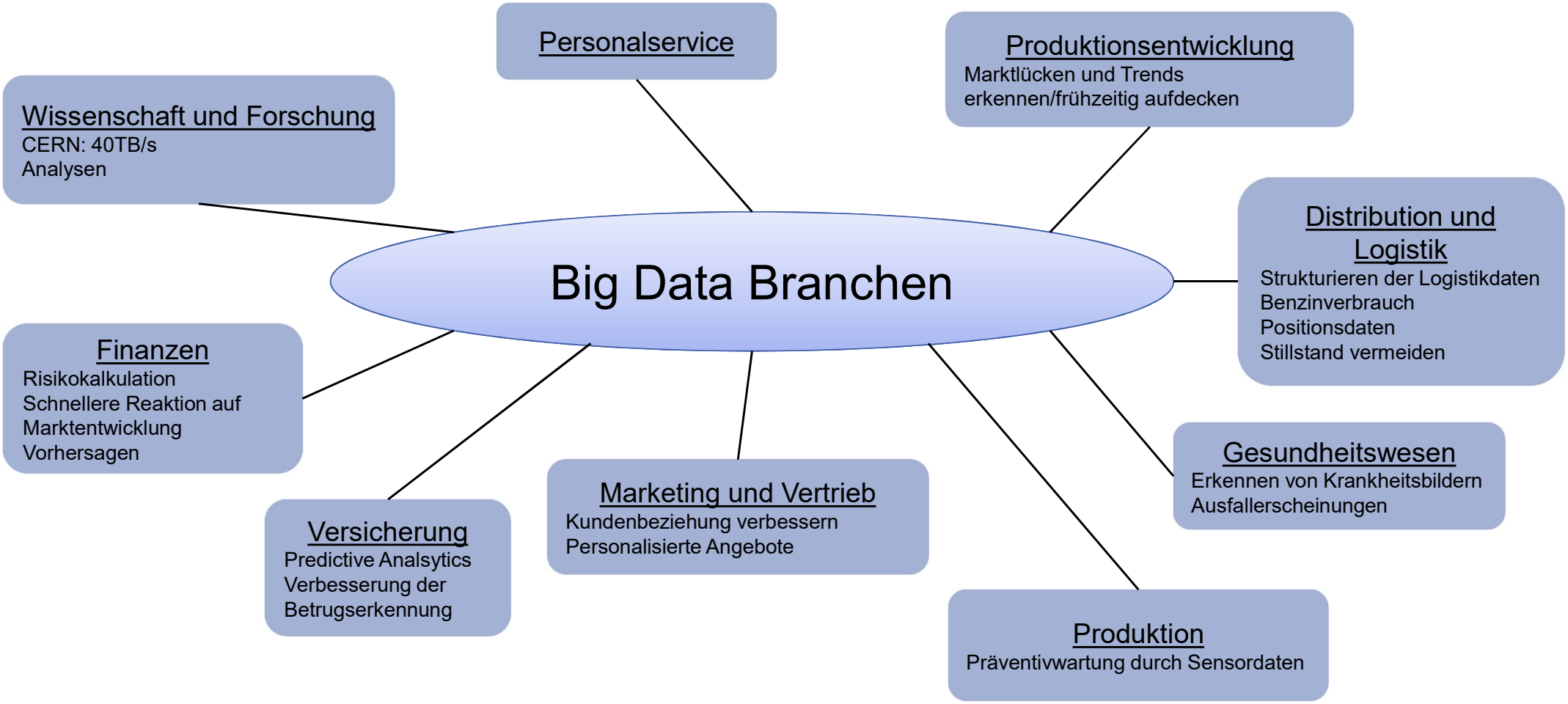






Maschinelles Lernen

Big Data Branchen bzw. Industriebeispiele



Praxisbeispiele
Einkaufsverhalten

- Prägnante Aussagen aus Bereich „Warenkorbanalyse“
- Einkaufspositionen systematisch auf interessante Kaufverhalten untersucht
- Nach Assoziation Ende der 90-er Jahre:
  - ► *„wer Produkt A kauft, kauft in x% der Fällen auch Produkt B“*

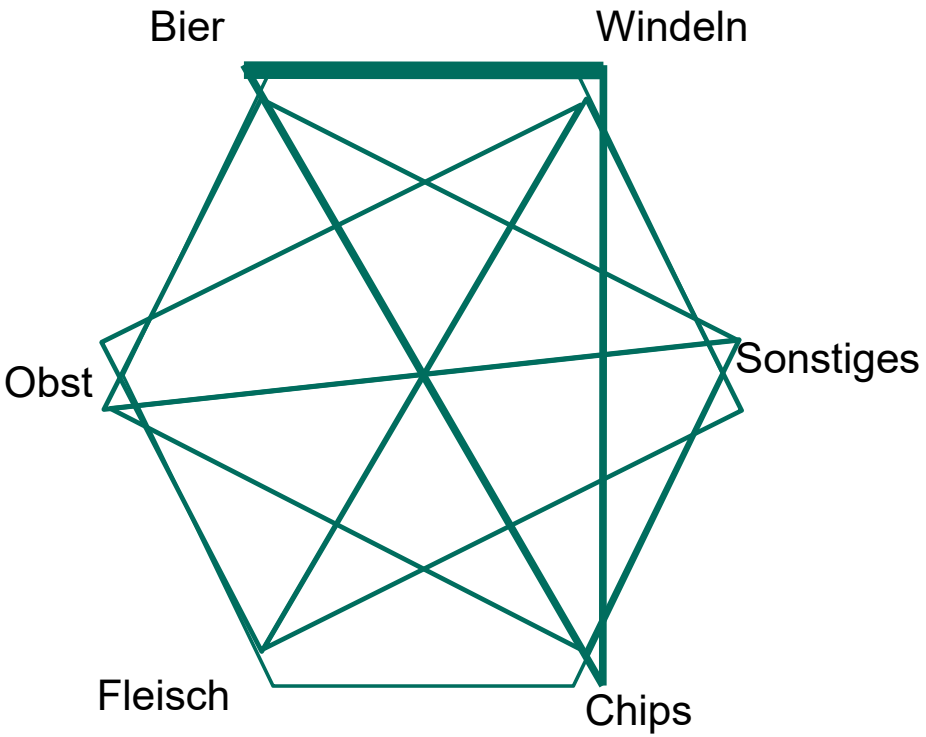
- Möglichkeiten
  - Für Onlinehändler Werbung im Warenkorb zu schalten („Kunden kaufen auch …“)
  - Für Supermärkte der Sortimentsoptimierung und -anordnung



Praxisbeispiele

Bier, Windeln und Big Data

- Wer Windeln kauft, kauft häufig auch Bier
  - *“Wal-Mart discovered through data mining that the sales of diapers and beer were correlated on Friday nights.”* [1]
- Ergebnis aus Kaufverhalten ersichtlich, Erklärung und Konsequenzen:
  - Frustrierte Väter, welche zum Windel holen geschickt wurden und sich mit einem Sixpack Bier belohnten [1]
  - In Supermärkten finden sich Bier und Windeln häufig nicht fern voneinander wieder



[1]: Hospel, H., Down the Rabbit Hole, Executive Update Online o.Jg. (2001) 3

[2]: Björn Bloching, Marketing- und Vertriebschef der Unternehmensberatung Roland Berger

Praxisbeispiele

Bier, Windeln und Big Data

- Wer Windeln kauft, kauft häufig auch Bier
  - *“Wal-Mart discovered through data mining that the sales of diapers and beer were correlated on Friday nights.”* [1]
- Ergebnis aus Kaufverhalten ersichtlich, Erklärung und Konsequenzen:
  - Frustrierte Väter, welche zum Windel holen geschickt wurden und sich mit einem Sixpack Bier belohnten [1]
  - In Supermärkten finden sich Bier und Windeln häufig nicht fern voneinander wieder



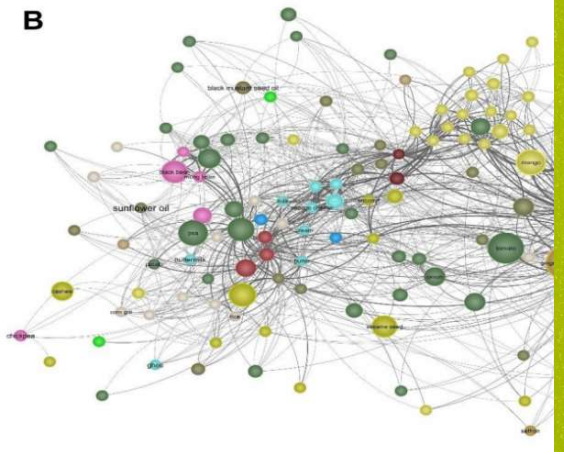
[1]: Hospel, H., Down the Rabbit Hole, Executive Update Online o.Jg. (2001) 3

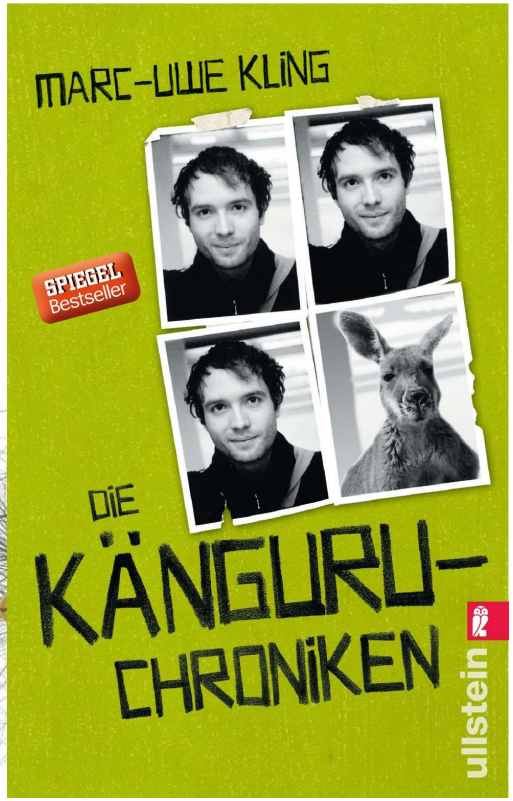
[2]: Björn Bloching, Marketing- und Vertriebschef der Unternehmensberatung Roland Berger

Praxisbeispiele

Einzigartigkeit der indischen Küche

- „Schwedische Wissenschaftler“ untersuchten verschiedene Rezepte und fanden heraus, dass in den Zutatenzusammenstellung auf das Ansprechen gleicher Geschmacksrezeptoren geachtet werden sollte.
- Folgende Kombinationen haben sich mit Hilfe von Maschinellem Lernen als „schmackhaft/lecker“ herausgestellt:
  - Weiße Schokolade und Kaviar
  - Erdbeeren und Erbsen
  - Spargel und Butter
  - Schokolade und Blauschimmelkäse



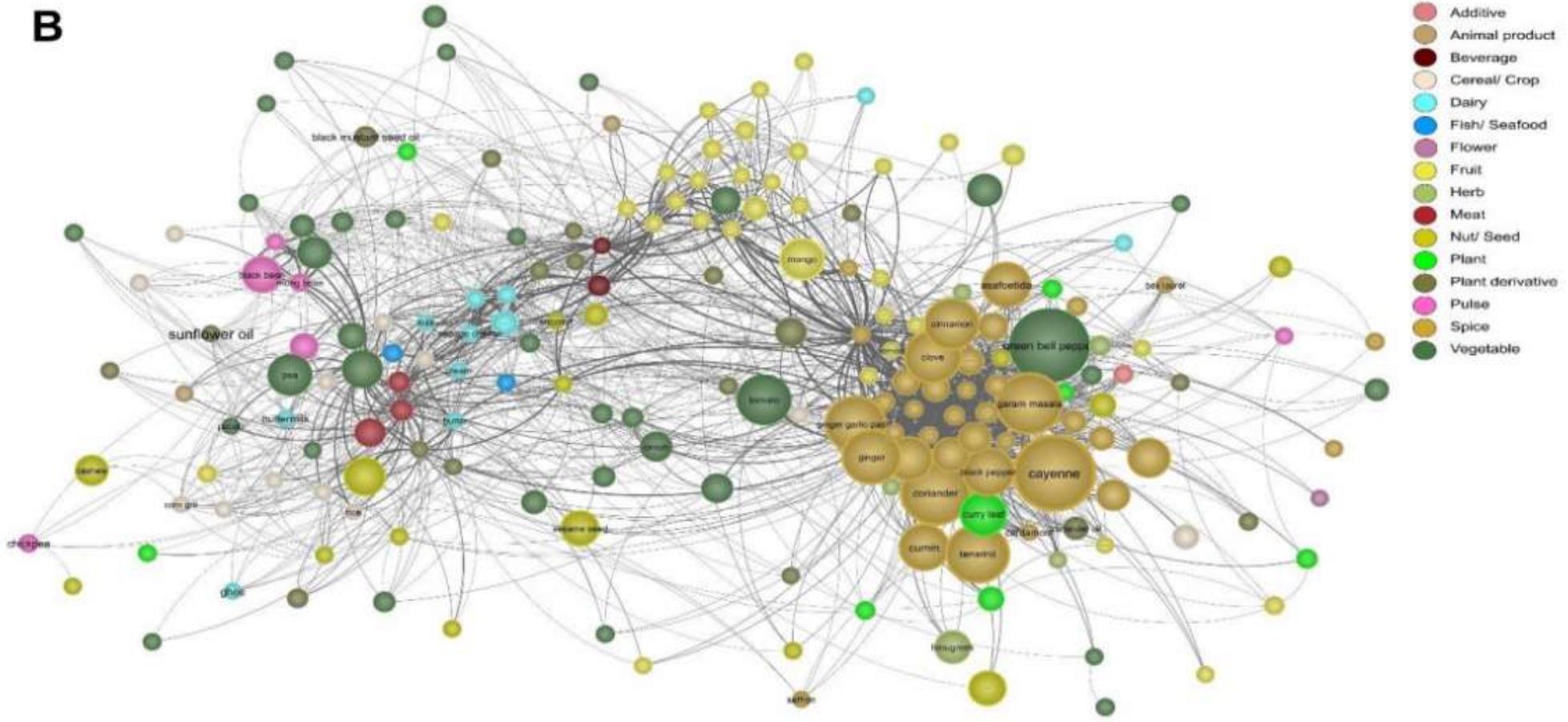


Die Zutaten-Kombinationshypothese funktioniert für jede Küche der Welt
→ außer für die indischen Küchen

Quelle: https://www.technologyreview.com/s/535451/data-mining-indian-recipes-reveals-new-food-pairing-phenomenon/

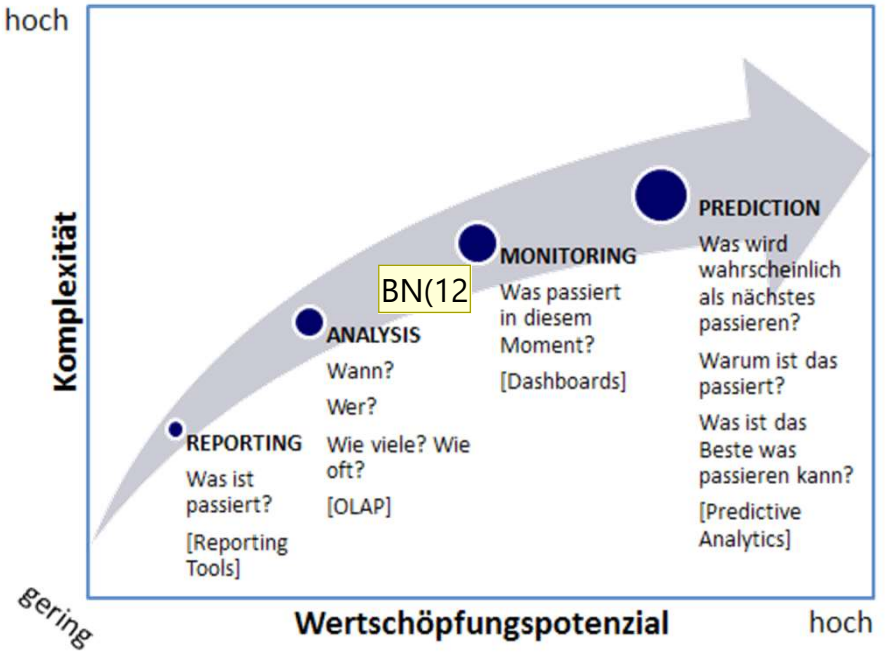
Einzigartigkeit der indischen Küche

Praxisbeispiel



Industriebeispiele:
Versicherungsbranche

- Große Datenmengen bei Versicherungsgesellschaften
- Neue Erkenntnisse bei Produktentwicklung möglich
  - Marketing und Vertrieb
  - Bestandsverwaltung
  - Schadens-Management
- Möglichkeit Risiken schneller und besser zu bewerten
- Versicherungsbeiträge oder Prämien berechnen



www.blog.affinis.de

Unfallkasse Hessen

- Ca 2.2 Millionen Versicherte
  (Feuerwehrleute, Studenten, Mitarbeiter im Gesundheitswesen, etc.)
- Pro Unfall 40 Merkmale, wie Ort, Alter des Beschädigten, Körperteil, etc.

BN(12 Grafik selber zeichnen

Brenner, Nathalie (ITIV); 15.01.2019

Industriebeispiele:

Versicherungsbranche - Unfallkasse Hessen

- Interessante Auffälligkeiten, Häufungen, Muster in Unfallmeldungen
  - 26% aller Unfälle von Krankenhauspersonal passieren durch (versehentlich) selbst zugefügten Stichverletzungen mit Spritzen
  - 60% aller verletzten Schüler beim Handball sind Mädchen – Anteil über alle Unfälle im Schulsport lediglich 43%
  - 25% aller Feuerwehrunfälle bei Jungfeuerwehr (7-14 Jahre) beim Sport (z.B. Knochenbrüche beim Fussballspielen)

Unfallkasse Hessen
Partner für Sicherheit

Auffälligkeiten, Häufungen, Muster identifizieren und als Ansatzpunkt zum Aufsetzen geeigneter Präventionsmaßnahmen nutzen

→ Versicherungsbeiträge bzw. -angebote anpassen

- Die 5 V‘s
- Quellen- und Datenvielfalt
- Analysegeschwindigkeit
- Data Mining und Maschinelles Lernen
- Wieso Big Data
- Wer verwendet Big Data



Backup

Deep Learning und Maschinelles Lernen

- Deep learning = Learning representations/features
- The traditional model of pattern recognition (since the late 50's)
  - Fixed/engineered features (or fixed kernel) + trainable classifier

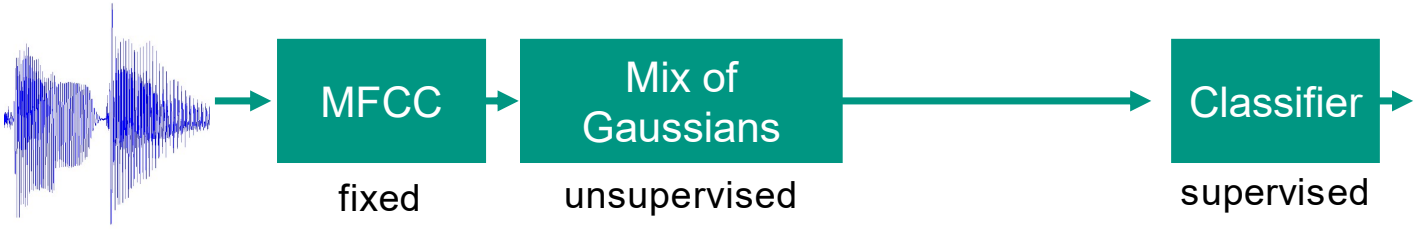


- End-to-end learning / Feature learning / Deep learning
  - Trainable features (or kernel) + trainable classifier

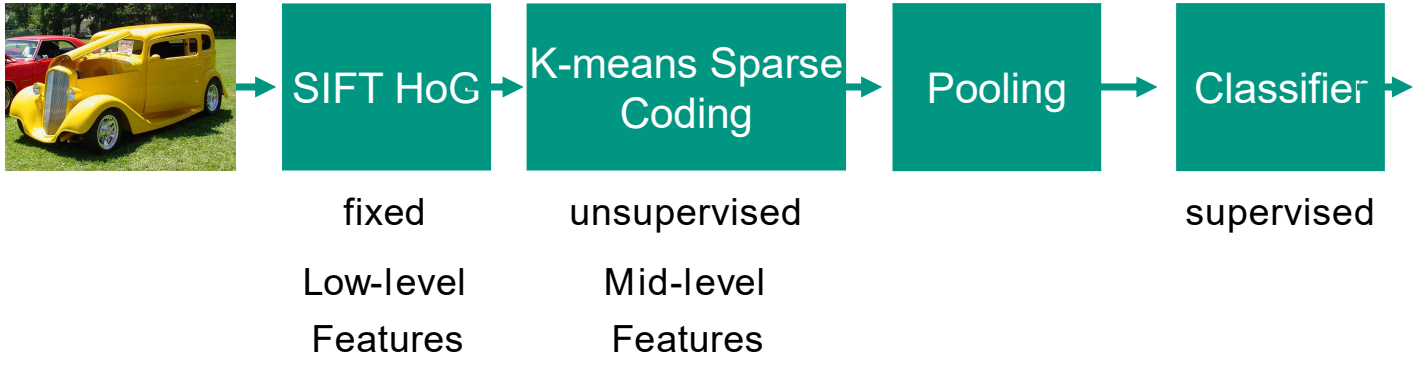


Architecture of “mainstream” pattern recognition systems

- Modern architecture for pattern recognition
  - Speech recognition: early 90's – 2011

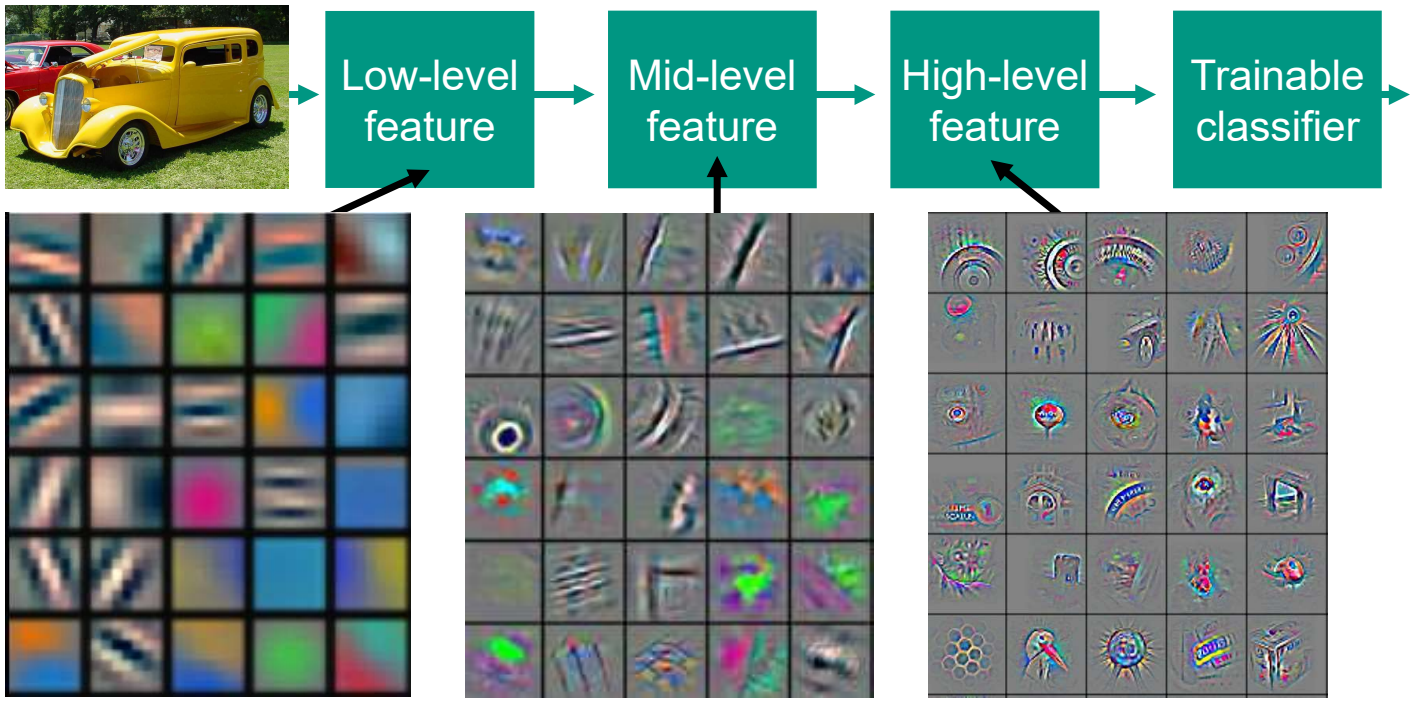


  - Object Recognition: 2006 - 2012



Deep learning = learning hierarchical representations

It's deep if it has more than one stage of non-linear feature transformation



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Karlsruher Institut für Technologie

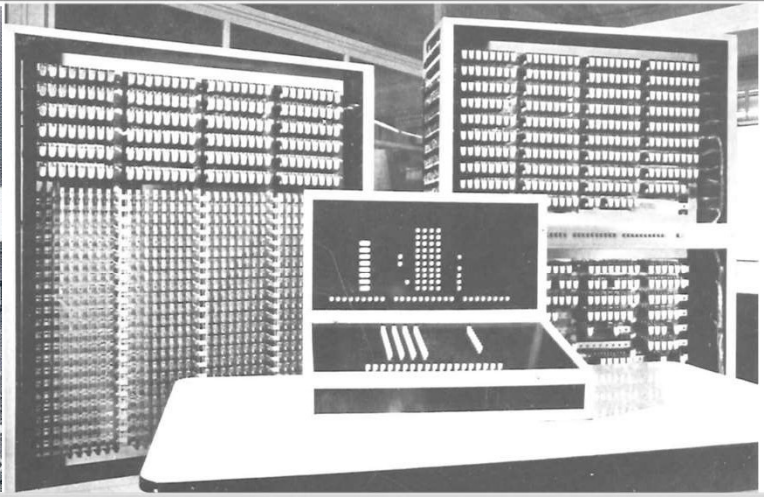
Institut für Technik der Informationsverarbeitung

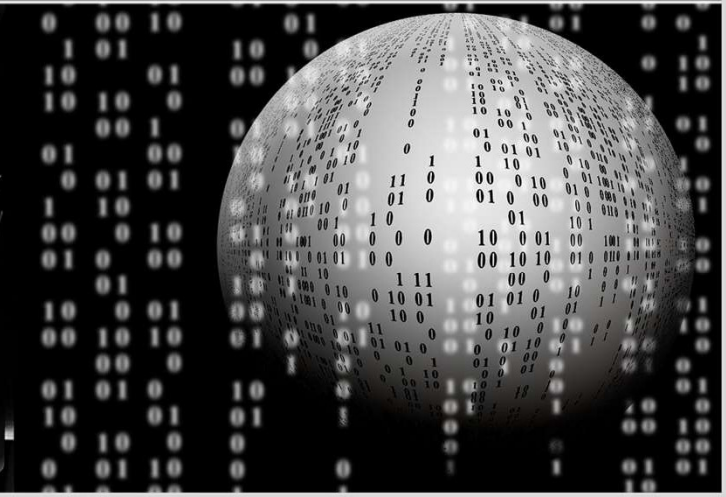
Institutsleitung
Prof. Dr.-Ing. J. Becker
Prof. Dr.-Ing. E. Sax
Prof. Dr. rer. nat. W. Stork

6. Big Data & Maschinelles Lernen (B)

Informationstechnik II

Prof. Dr.-Ing. Eric Sax



Inhalt IT2

6. Big Data & Maschinelles Lernen

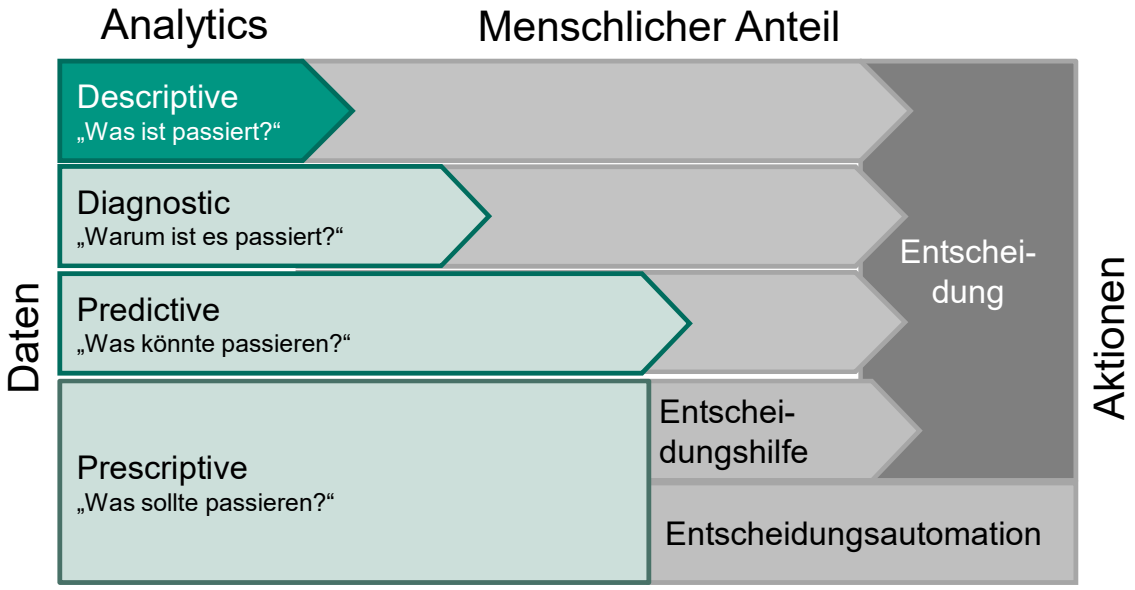
- Definitionen, Verwendung (Charakteristik, Risiken, Chancen)
- Motivation und Anwendungsfälle
- Data Science Prozesse:
  - KDD
  - CRISP-DM
- CRISP-DM im Detail
  - Business Understanding
  - Data Understanding
  - Data Preparation
  - Modeling
  - Evaluation
- Infrastruktur für Big Data



Prozess

Analyseformen auf Daten

- Es geht darum Daten zu analysieren, um daraus Erkenntnisse zu gewinnen.
- Erkenntnisse dienen als Grundlage einer Entscheidung und lösen Handlungen aus.

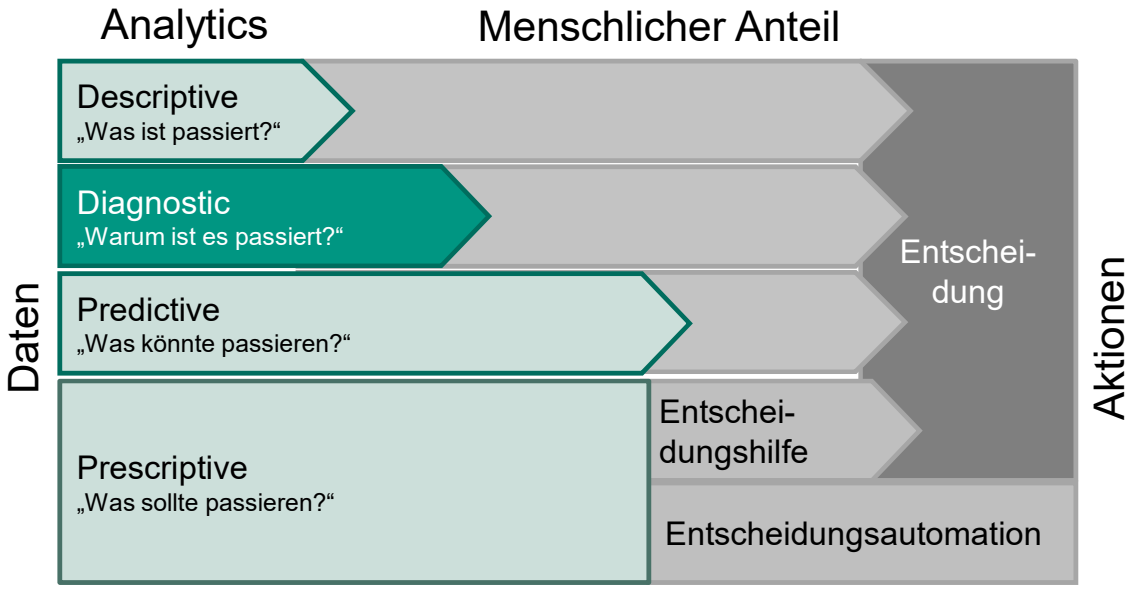


- Descriptive Analytics
  - Die Blickrichtung ist auf die Vergangenheit gerichtet und beschreibt, was passiert ist.
  - Der Anteil, den ein Mensch noch beitragen muss, um zu einer Entscheidung und einer Handlung zu kommen, ist groß.
  - Beispiel:
    - Eine Bank analysiert die Daten von Kunden, die ihre Kreditraten nicht bezahlt haben.
    - Sie gruppiert sie vielleicht oder ordnet sie geografischen oder sozioökonomischen Faktoren zu.

Prozess

Analyseformen auf Daten

- Es geht darum Daten zu analysieren, um daraus Erkenntnisse zu gewinnen.
- Erkenntnisse dienen als Grundlage einer Entscheidung und lösen Handlungen aus.

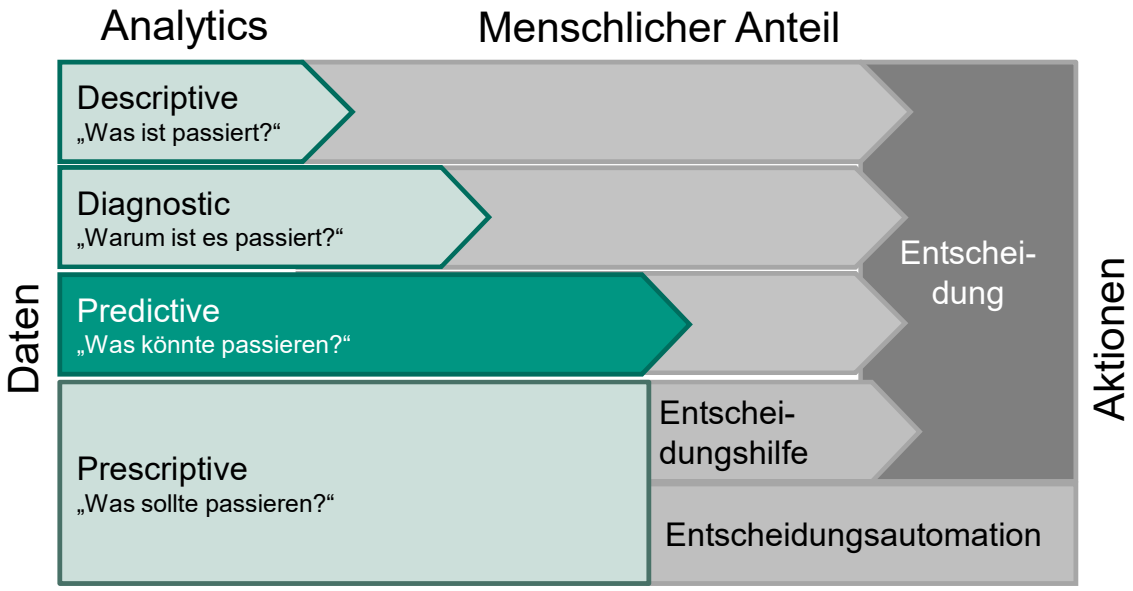


- Diagnostic Analytics:
  - Will man diagnostizieren, wieso bestimmte Sachverhalte eingetreten sind, können mit der Anwendung entsprechender Verfahren (Kausalanalysen) die Gründe gesucht werden.
  - Die Blickrichtung ist zwar immer noch auf die Vergangenheit gerichtet, aber das Ziel ist es, nicht nur Sachverhalte darzustellen, sondern auch Gründe für deren Eintreten zu finden.
  - Der menschliche Anteil bei der Entscheidungsfindung ist schon etwas geringer, da die Analyse eine weitere Entscheidungsgrundlage liefert.
  - Beispiel:
    - Durch die Analyse der Kreditausfalldaten erkennt die Bank, welche Faktoren die Zahlungsausfälle begünstigt haben.

Prozess

Analyseformen auf Daten

- Es geht darum Daten zu analysieren, um daraus Erkenntnisse zu gewinnen.
- Erkenntnisse dienen als Grundlage einer Entscheidung und lösen Handlungen aus.



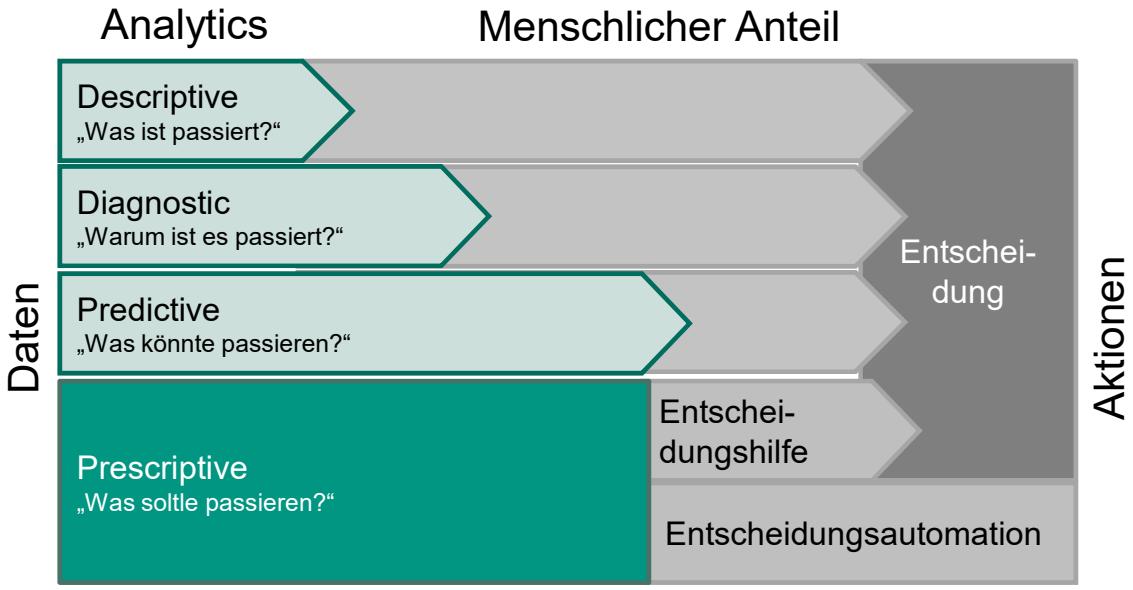
- Predictive Analytics
  - nutzt die Erkenntnisse aus der Datenanalyse, um eine Prognose für zukünftige Fälle zu geben.
  - Die Blickrichtung ist also auf die Zukunft gerichtet.
  - Beispiel:
    - Die Bank erstellt anhand der vorangegangenen Analysen ein Scoring-Modell.
    - Ein neuer Kreditantragsteller wird mit diesem Modell bewertet.
    - Der Scoringwert ist ein Entscheidungskriterium für die Frage, ob und zu welchen Konditionen der Kunde den Kredit bekommt.
    - Es muss aber nicht unbedingt das einzige Kriterium sein.

    ► Die „Nase“ des Antragstellers wird bei einem persönlichen Gespräch mit dem Bankmitarbeiter eventuell auch noch bewertet.

Prozess

Analyseformen auf Daten

- Es geht darum Daten zu analysieren, um daraus Erkenntnisse zu gewinnen.
- Erkenntnisse dienen als Grundlage einer Entscheidung und lösen Handlungen aus.



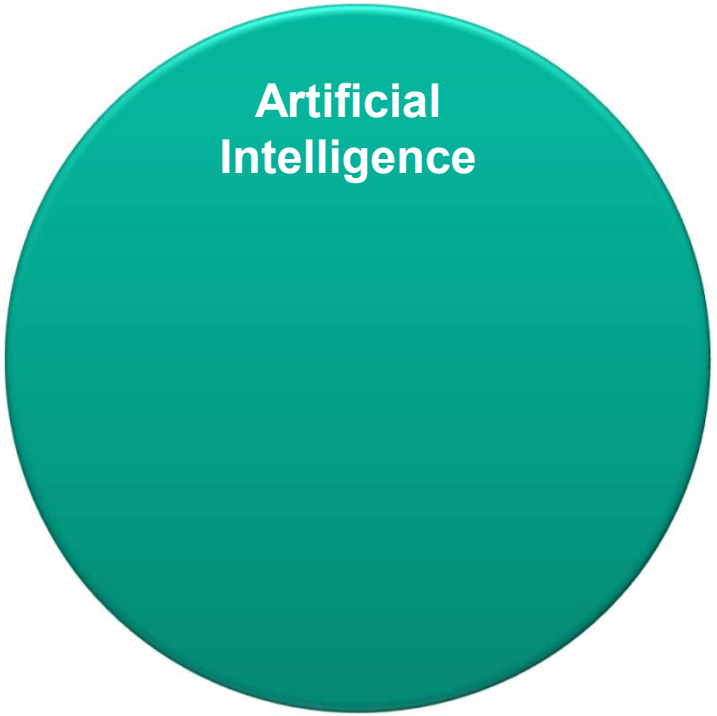
- Prescriptive Analytics
  - geht noch einen Schritt weiter und lässt den menschlichen Anteil an einem Entscheidungsprozess unnötig werden.
  - Es werden anhand der Daten automatische Entscheidungen oder sogar Handlungen ausgelöst.
  - Beispiel:
    - Für kleinere Konsumentenkredite bietet die Bank im Web die komplette Abwicklung eines Kreditantrages an.
    - Anhand der eingegebenen und vorhandenen Daten über den Antragsteller fällt die Kreditentscheidung automatisch und die Auszahlung wird veranlasst.

Begriffe (1)

- **Business Intelligence:**
  - Verfahren und Prozesse zur systematischen Analyse (Sammlung, Auswertung und Darstellung) von Daten (1990er).
  - Ziel ist die Gewinnung von Erkenntnissen, die in Hinsicht auf die Unternehmensziele bessere operative oder strategische Entscheidungen ermöglichen.
  - Der analytische Teil spielt eine untergeordnete Rolle und wir bewegen uns im Bereich der deskriptiven Analytics, also auf der ersten Stufe (s. Folie zuvor).
  - Mit Intelligenz (im Sinne von Verfahren der künstlichen Intelligenz) hat das noch wenig zu tun.
- **Knowledge Discovery (KD):**
  - Als man damit begann, systematisch Daten zu analysieren, um damit Erkenntnisse zu gewinnen (1980er), wurde oft noch der Begriff Knowledge Discovery (Process) verwendet.
  - Damit wurde der gesamte analytische Prozess des Erkenntnisgewinns aus Daten gemeint, der die Diagnostic- und Predictive-Aspekte miteinbezog.
- **Data-Mining:**
  - Eng verbunden mit der KD ist „Data-Mining", also das Schürfen nach Goldschätzen in Datensätzen.
  - Damit sind vor allem die Diagnostic- und Predictive- Verfahren gemeint.
  - Data-Mining ist damit eigentlich nur ein „fancy" Begriff für die Knowledge Discovery und wird zum Teil synonym, zum Teil nur auf den eigentlichen, analysierenden Teil des Gesamtprozesses bezogen, verwendet.
- **Big Data Analytics:**
  - Diese analytischen Verfahren und Vorgehensweisen werden auf Big Data (5 V's) angewandt.

Begriffe (2)

- **Künstliche Intelligenz (KI) – Artificial Intelligence (AI)**:
  - KI ist ein Teilgebiet der Informatik, das sich mit der Automatisierung intelligenten Verhaltens befasst.
  - KI ist der Versuch, einen Computer so zu programmieren, dass er – ähnlich einem Menschen – eigenständig Probleme bearbeiten kann.
  - Problemlösung bedeutet oft, anhand von Daten unterschiedlicher Art (strukturierte Daten, Bilder, Sensordaten) Entscheidungen zu treffen.
  - Damit kann die KI als Oberbegriff für die *Prescriptive*- Analytics-Verfahren gesehen werden.
  - Sie geht aber darüber hinaus, da auch Bereiche wie selbstlernende Systeme, autonomes Handeln (z. B. Fahren), eigenständige Erkenntnisprozesse etc. darin umfasst sind.



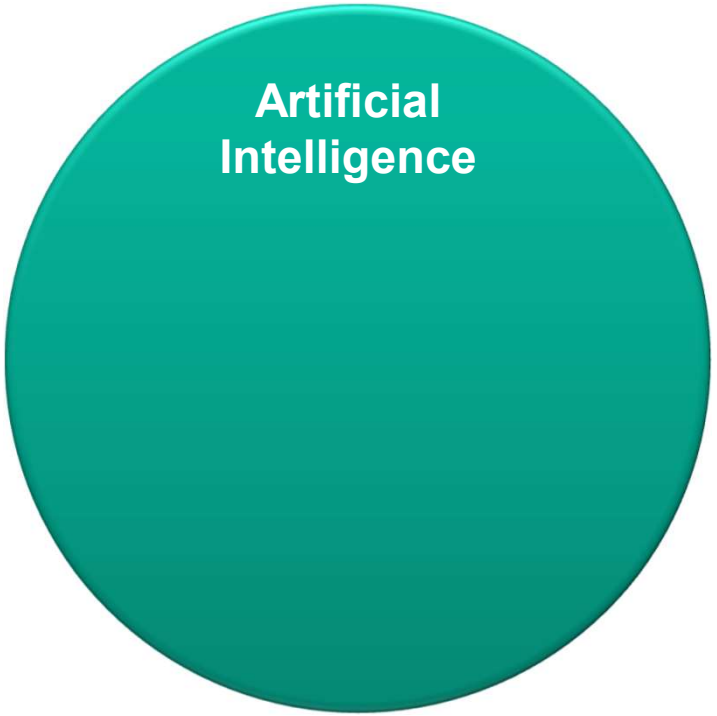
- Remark → Passenderer ist der Begriff „künstlicher Fachidiot“

Begriffe (2)

- **Künstliche Intelligenz (KI) – Artificial Intelligence (AI)**:
  - KI ist ein Teilgebiet der Informatik, das sich mit der Automatisierung intelligenten Verhaltens befasst.
  - KI ist der Versuch, einen Computer so zu programmieren, dass er – ähnlich einem Menschen – eigenständig Probleme

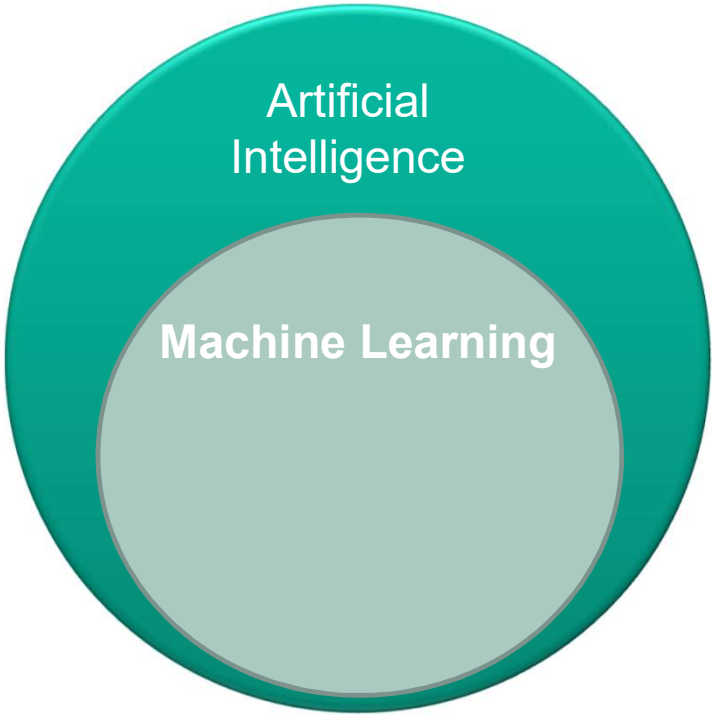


„künstlicher Fachidiot“



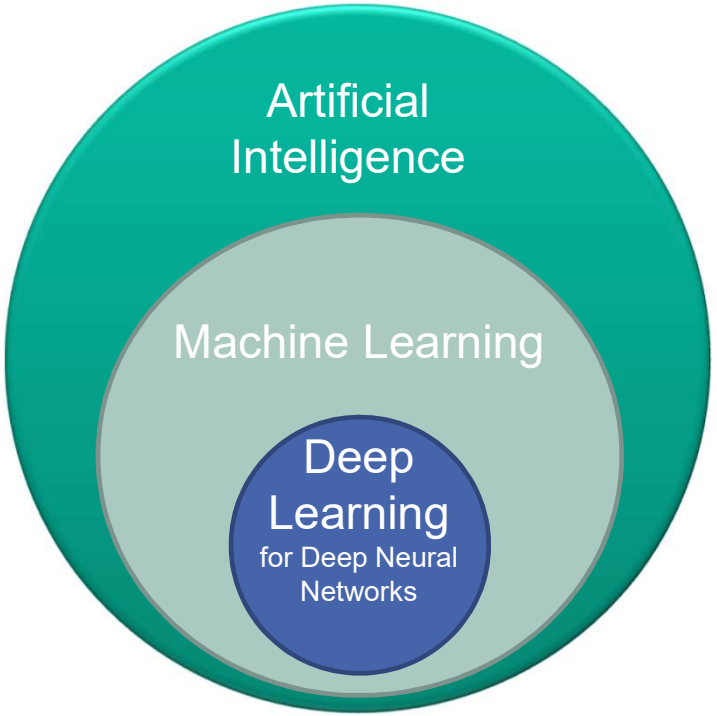
Begriffe (3)

- **Maschinelles Lernen – Machine Learning (ML):**
  - Im Begriff Maschinelles Lernen sind die statistischen, mathematischen Verfahren inbegriffen, die im Bereich der Diagnostic und Predictive Analytics eingesetzt werden.
  - Die Idee ist geprägt von der Frage, wie man einer Maschine (Computer) etwas beibringen kann:
    - Nämlich nicht, indem man ein starres Programm schreibt, sondern indem man ihm das Lernen ermöglicht.
  - Allgemeine Definition
    - „*Machine Learning is the field of study that gives computers the ability to learn without explicitly programmed.*“ [Arthur Samuel, 1959]
  - Ingenieurorientierte Definition
    - *“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”* [Tom Mitchel, 1997]
  - Bildet die technische Basis (Algorithmen) für Data Mining



Begriffe (3)

- **Deep Learning:**
  - Deep Learning bezieht sich auf einen Ausschnitt der Machine-Learning-Verfahren, nämlich v. a. auf die neuronalen Netze mit mehreren Hidden Layers und weitere Graphen-Verfahren (z. B. Multi-Class Logistic Regression).
  - Es sind also Verfahren, die besonders geeignet erscheinen, Erkenntnisse und Entscheidungen aus der Analyse von Daten zu gewinnen.



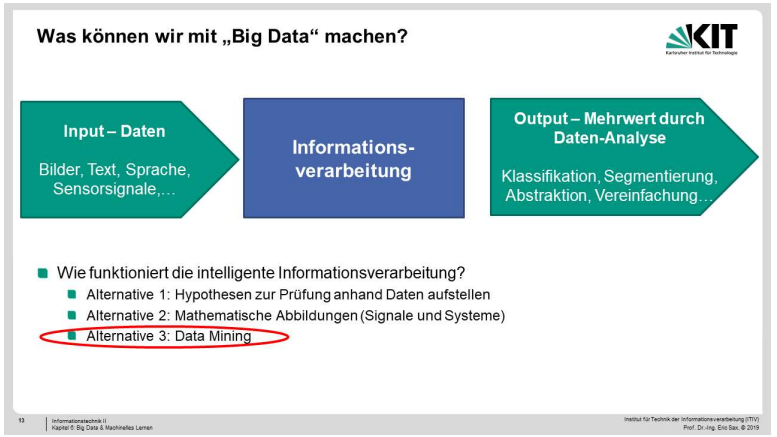
Maschinelles Lernen und Data Mining

„untrainiertes“ Modell

Input – Daten
Information
(+ evtl. Antworten)

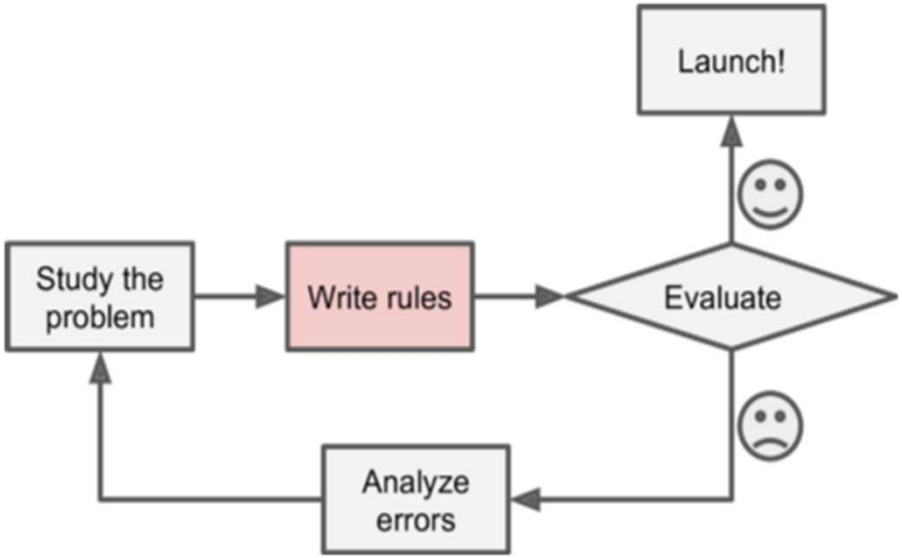
Maschinelles Lernen
- Beziehungen
- Muster
- Abhängigkeiten
- Versteckte Strukturen

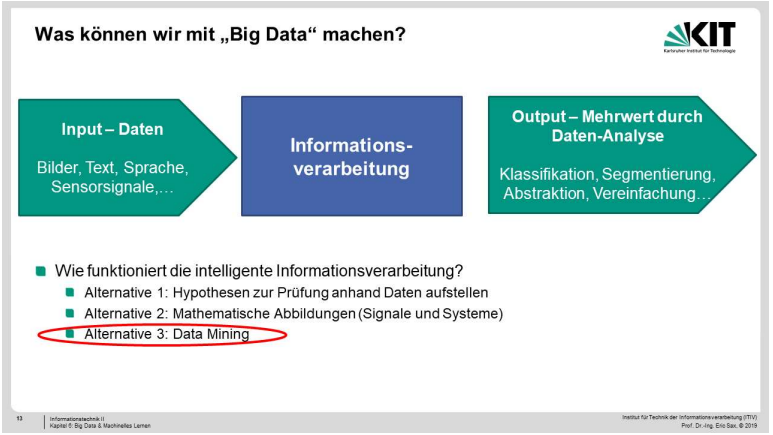
Trainiertes / Gelerntes Modell



Warum benutzen wir Maschinelles Lernen (1)

Spam Filter (traditionell)

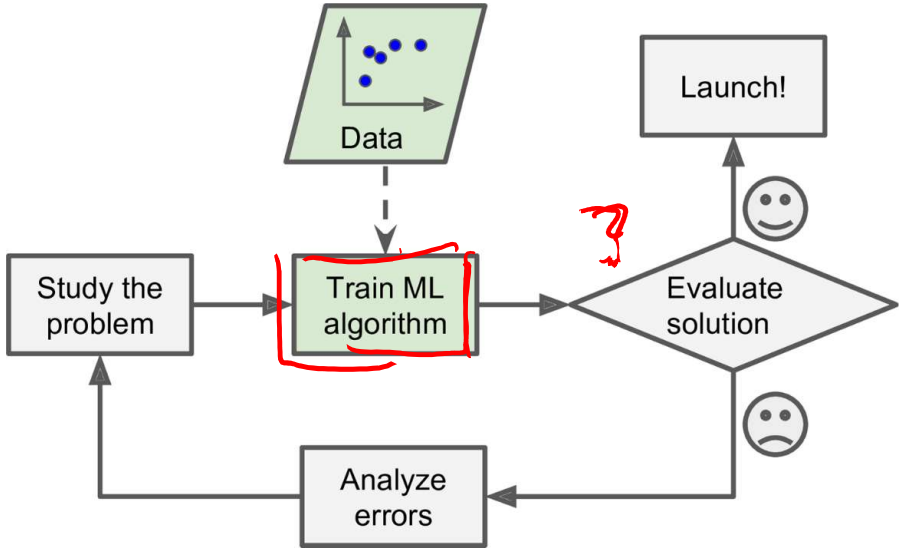


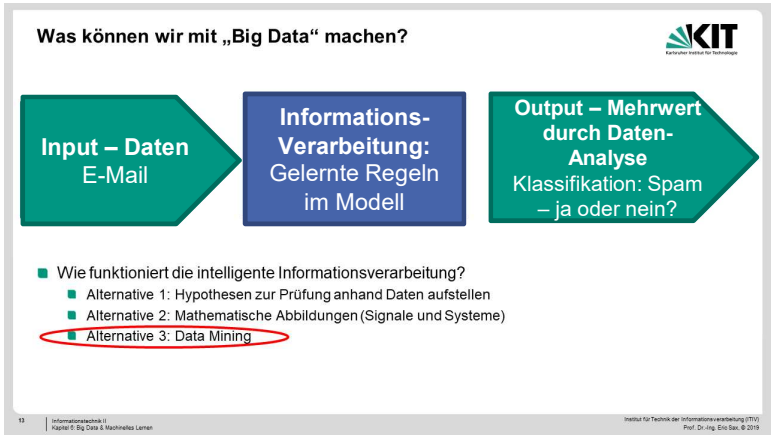


- Nachteile: „Rules“ (Programme, Funktionen, Prozeduren, …) müssen von Hand geschrieben und ständig angepasst werden.

Warum benutzen wir Maschinelles Lernen (2)

Spam Filter (modern)





- Lernt automatisch, z.B. welche Wörter „gute Prädiktoren“ für Spam sind.
- Code viel kürzer, einfacher zu warten und höchstwahrscheinlich genauer.

Was ist Lernen?

Lernen – Inferenz – Intelligenz – Deep Learning

- Lernen = sich neues Wissen und Fähigkeiten aneignen
- Üben = etwas immer wieder tun, um es zu automatisieren und dann besser zu beherrschen
- Trainieren = durch Übungen leistungsfähiger werden
- Lernen setzt eine Transferleistung voraus → Generalisierung notwendig

- Definitionen von „Lernen“
  - **Simon**: jede Veränderung eines Systems, die es ihm erlaubt, eine Aufgabe bei Wiederholung derselben Aufgabe oder einer Aufgabe derselben Art besser zu lösen.
  - **Scott**: ein Prozess, bei dem ein System eine abrufbare Repräsentation von vergangenen Interaktionen mit seiner Umwelt aufbaut.
  - **Michalski**: das Konstruieren oder Verändern von Repräsentationen von Erfahrungen.

Varianten und Kategorien von selbstlernenden Systemen

Einteilung danach …

- **… ob sie inkrementell lernen können oder nicht**
  - (Online versus Batch-Learning)
- **… ob sie einfach neue Datenpunkte mit bekannten Datenpunkten vergleichen oder stattdessen Muster in den Trainingsdaten erkennen und ein prädiktives Modell erstellen**
  - (instanzbasiertes versus modellbasiertes Lernen)
- **… ob sie unter menschlicher Aufsicht ausgebildet werden**
  - (überwachten, unüberwachten, halbautomatisch und Reinforcement Learning)

Lernen

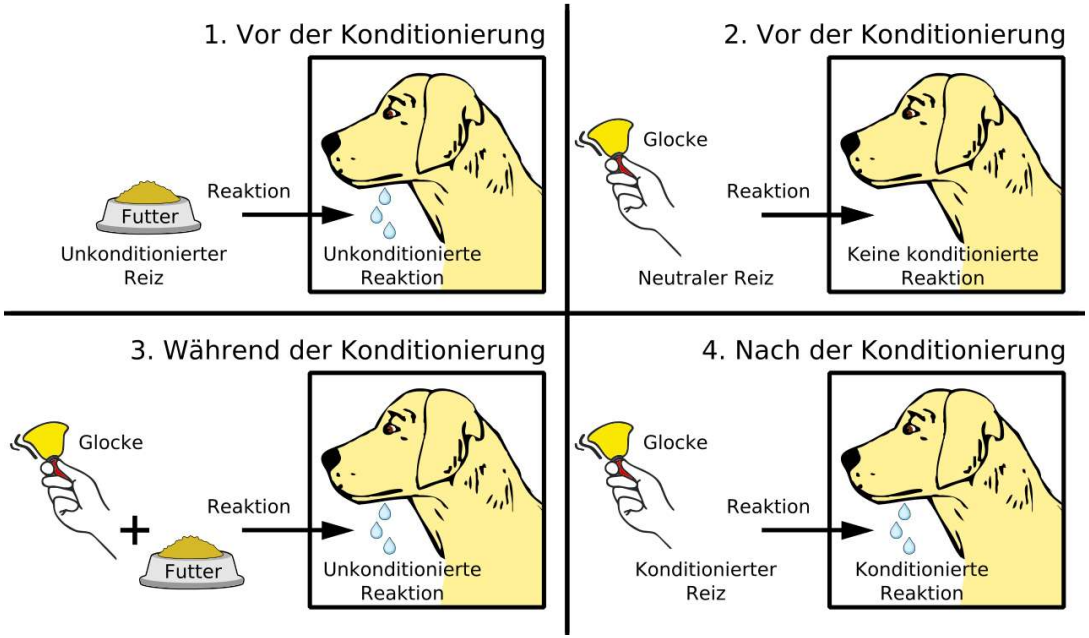
Bedingter Reflex nach Pawlow

- Pawlow hatte im Verlauf seiner Experimente zum Zusammenhang von Speichelfluss und Verdauung beobachtet, dass bei Zwingerhunden schon die Schritte des Besitzers Speichelfluss auslösten, obwohl noch gar kein Futter in Sicht war.
- Er vermutete, dass das Geräusch der Schritte, dem regelmäßig die Fütterung folgte, für die Hunde mit Fressen verbunden war.
- Dabei hat ein Hund eigentlich gar keinen Grund bei einfachen hörbaren Schritten mehr zu sabbern – es sei denn, er hat gelernt, den unbedeutenden Reiz der Schritte mit der Gabe von Futter zu verknüpfen.
- Die Theorie dieses Lernprozesses beim Hund wollte Pawlow nun beweisen.

Lernen

Bedingter Reflex nach Pawlow

- Um diese Hypothese zu prüfen, gestaltete er 1905[1] ein aussagekräftiges Experiment:
  - Auf die Darbietung von Futter, einem unbedingten Reiz, folgt Speichelfluss (unbedingte Reaktion), auf das Ertönen eines Glockentons (neutraler Reiz) nichts.
  - Wenn aber der Glockenton wiederholt in engem zeitlichem Zusammenhang mit dem Anbieten von Futter erklingt, reagieren die Hunde schließlich auf den Ton allein mit Speichelfluss.
  - Dieses Phänomen bezeichnete Pawlow als Konditionierung.



Lernen

Bedingter Reflex nach Pawlow

- Um diese Hypothese zu prüfen, gestaltete er 1905[1] ein aussagekräftiges Experiment:
  - Auf die Darbietung von Futter, einem unbedingten Reiz, folgt Speichelfluss (unbedingte Reaktion), auf das Ertönen eines Glockentons (neutraler Reiz) nichts.
  - Wenn aber der Glockenton wiederholt in engem zeitlichem Zusammenhang mit dem Anbieten von Futter erklingt, reagieren die Hunde schließlich auf den Ton allein mit Speichelfluss.
  - Dieses Phänomen bezeichnete Pawlow als Konditionierung.



- **Klassische Konditionierung** ist eine von dem russischen Physiologen Iwan Petrowitsch Pawlow begründete behavioristische Lerntheorie, die besagt, dass einer natürlichen, meist angeborenen, sogenannten *unbedingten* Reaktion durch Lernen eine neue, *bedingte* Reaktion hinzugefügt werden kann.

Bedingter Reflex nach Pawlow

- Wie kann der „bedingte Reflex“ in einem technisches System umgesetzt werden?

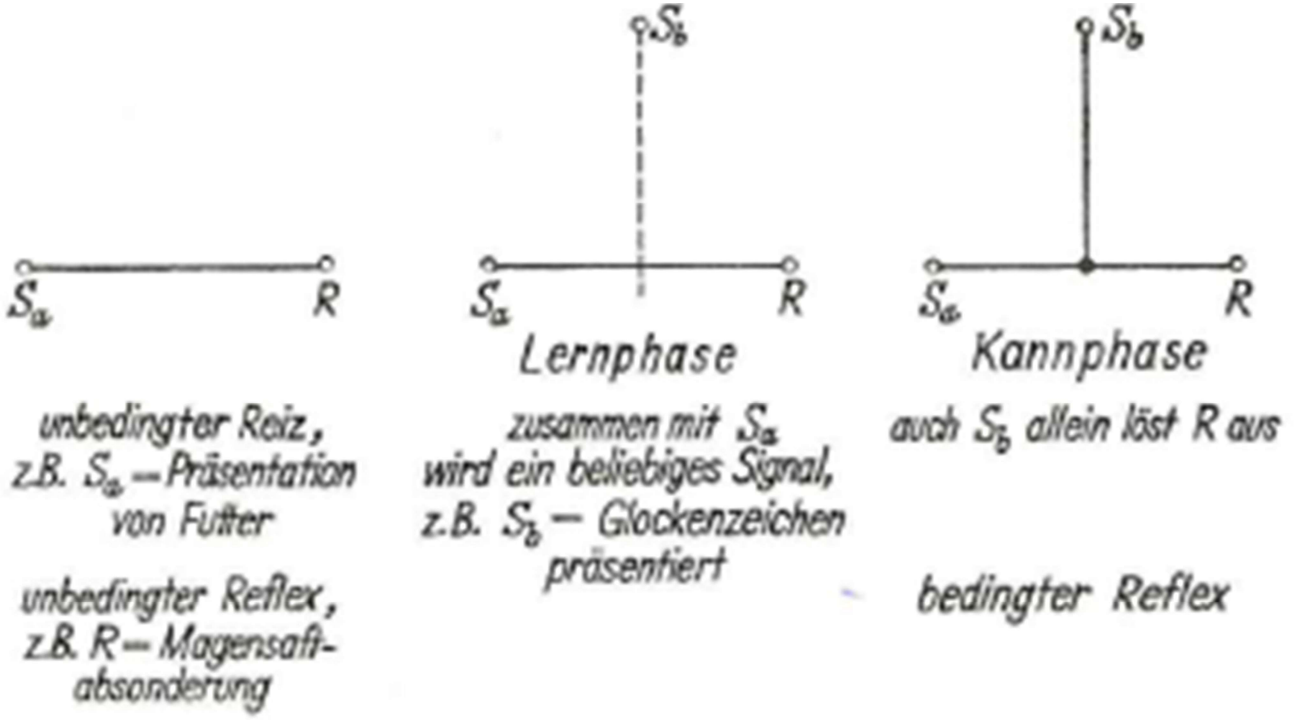


Bild 102. Bedingter Reflex
(schematische Darstellung des Versuches von J. P. Pawlow)

Bedingte Verknüpfung

- Lernphase:
  - Treten Signal e und b gleichzeitig auf, wird die Verknüpfung v vergrößert.
  - Tritt jedoch gleichzeitig mit b nicht e auf, wird die Verknüpfung verkleinert.
- Kannphase:
  - Verhalten hängt von Lernphase ab:
    - Trat mit Signal b stets Signal e auf (e ist ein Indiz für b), besteht eine maximale Verknüpfung (zwischen e und b)
    - Trat mit Signal b im Regelfall nicht Signal e auf (e ist kein Indiz für b), besteht eine minimale Verbindung (zwischen e und b)

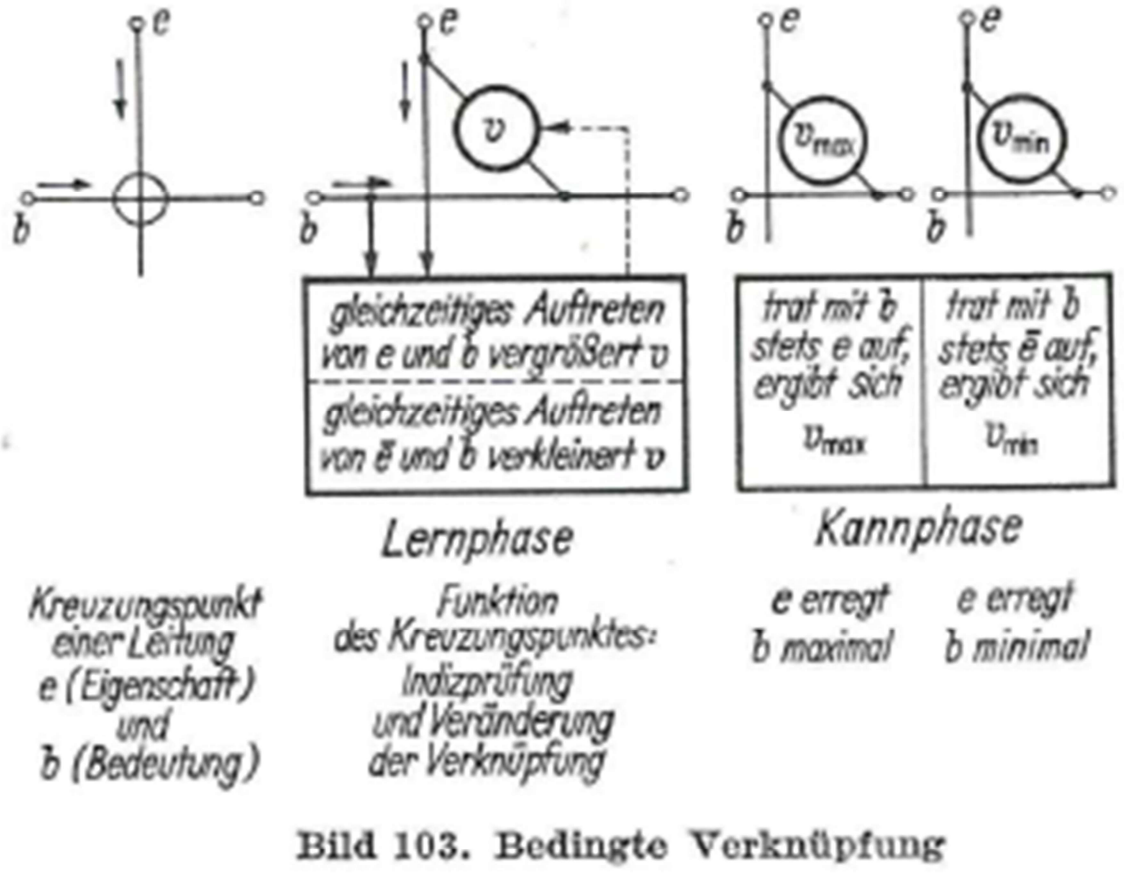


Bild 103. Bedingte Verknüpfung

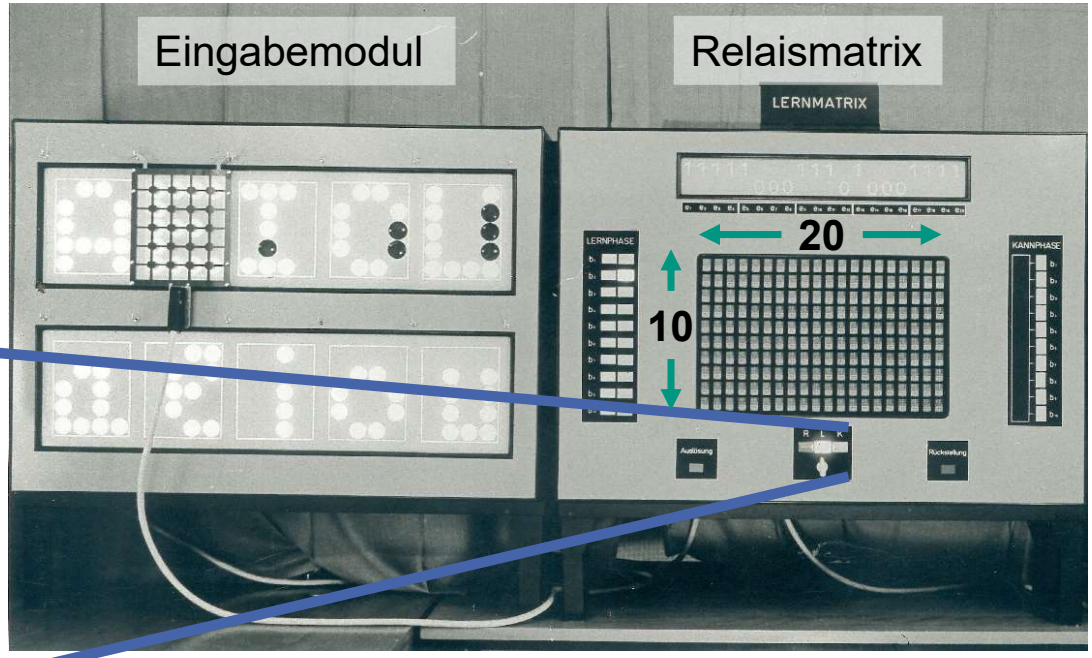
Lernmatrix für binäre Signale

nach Karl Steinbuch

- Das Demonstrationsmodell der binären Lernmatrix besteht aus zwei Modulen.
- Das Eingabemodul bietet zehn verschiedene optische Zeichen als Hell-/Dunkel-Kontrast in je einer Matrix aus 5 x 4 Punkten an
  - … die optisch über Fotoelemente abgetastet
  - … die dem zweiten Modul als 20-stelliger Merkmalsvektor angeboten werden.
- Dort realisiert eine Relais-Matrix aus 10 x 20 Relais eine binäre Verkopplung zwischen Spalten- und Zeilensignalen.
  - 20 Punkte
  - 10 optische Zeichen



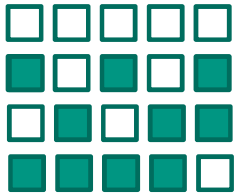
- Ein Betriebsartenschalter für Lern-, Kann- und Rückstellungsphase sowie Taster zur Auslösung der Funktionen und zur Selektion der Zeilen erlauben die manuelle Bedienung des Moduls.



Lernmatrix für binäre Signale

4 binäre Eigenschaftssätze (Pixel-Vektoren) A,B,C,D

	e_1	e_2	e_3	e_4	e_5
A	0	0	0	0	0
B	1	0	1	0	1
C	0	1	0	1	1
D	1	1	1	1	0



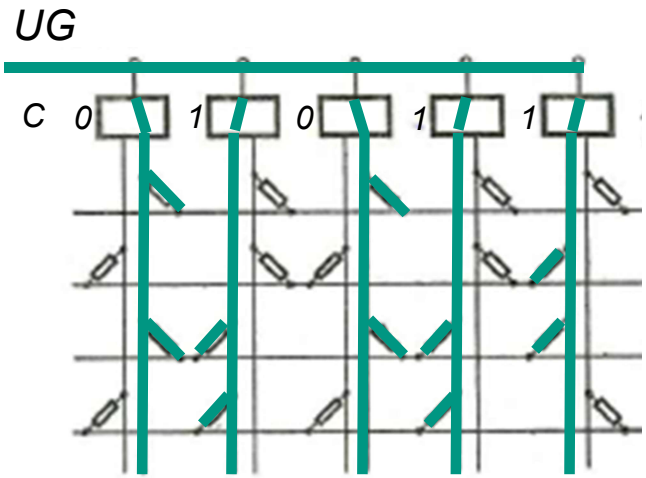


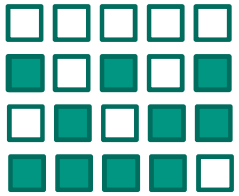
Bild 107. Lernmatrix für binäre Signale

- Eingabe erfolgt „kontradiktorisch“
  - e = 0 (Pixel aus)
    - ► Linke Spaltenleitung: 0
    - ► rechte Spaltenleitung: Versorgungsspannnug (UG)
  - e = 1 (Pixel an)
    - ► linke Spaltenleitung: UG
    - ► rechte Spaltenleitung: 0
- Lernphase:
  - Relais schaltet auf 0 oder UG (exklusiv)
    - ► Bedingte Verknüpfung haben sich gebildet
- Kannphase:
  - Ströme von Spaltenleitungen in die Zeilenleitung
    - ► Bei A voller Strom in der 1. Zeile → A erkannt
    - ► Bei B voller Strom in der 2. Zeile → B erkannt
    - ► Usw.

Lernmatrix für binäre Signale

4 binäre Eigenschaftssätze (Pixel-Vektoren) A,B,C,D

	e_1	e_2	e_3	e_4	e_5
A	0	0	0	0	0
B	1	0	1	0	1
C	0	1	0	1	1
D	1	1	1	1	0



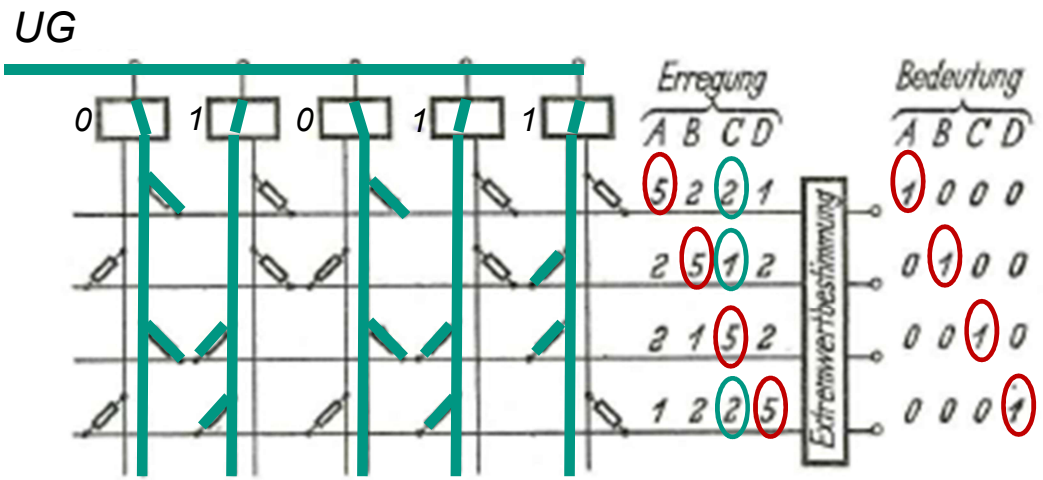
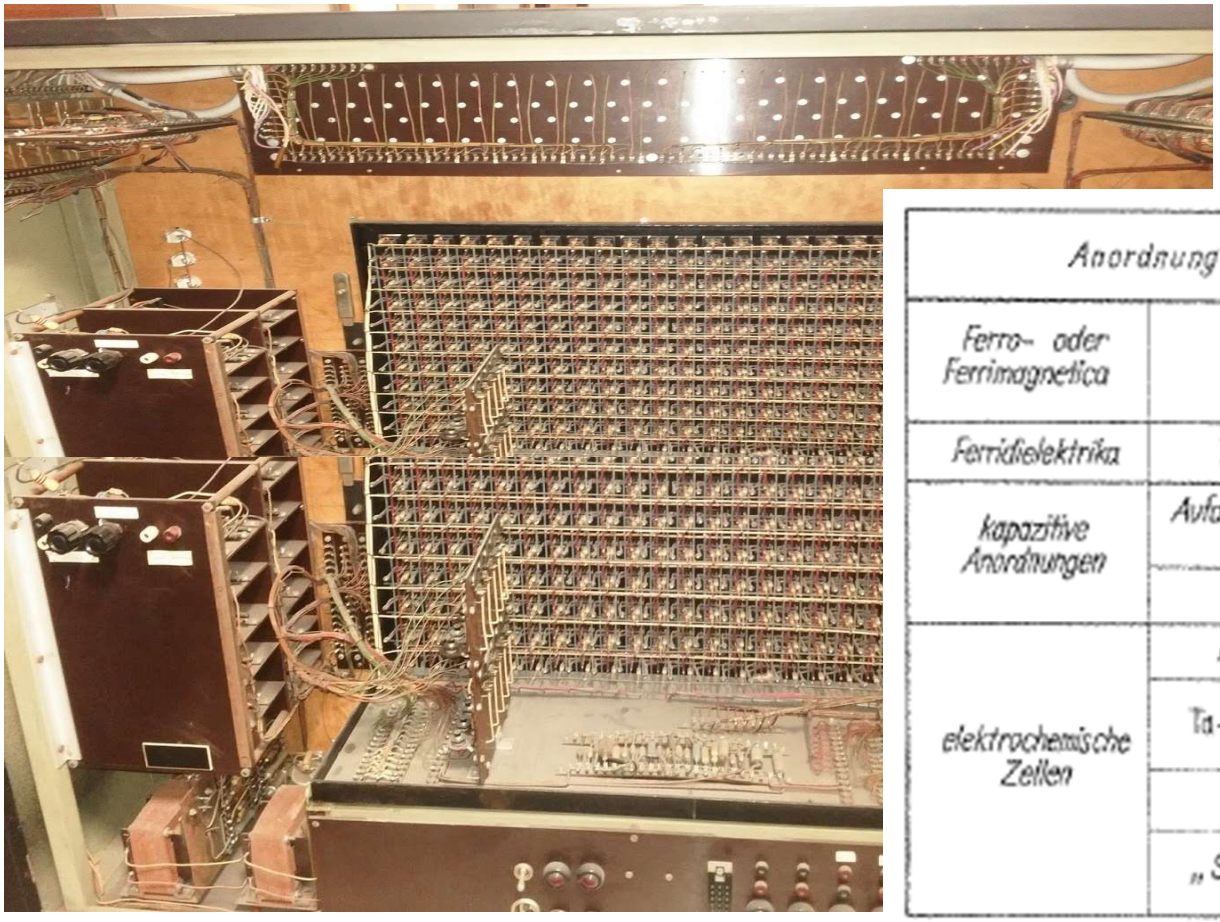


Bild 107. Lernmatrix für binäre Signale

- Eingabe erfolgt „kontradiktorisch“
  - e = 0 (Pixel aus)
    - ► Linke Spaltenleitung: 0
    - ► rechte Spaltenleitung: Versorgungsspannnug (UG)
  - e = 1 (Pixel an)
    - ► linke Spaltenleitung: UG
    - ► rechte Spaltenleitung: 0
- Lernphase:
  - Relais schaltet auf 0 oder UG (exklusiv)
    - ► Bedingte Verknüpfung haben sich gebildet
- Kannphase:
  - Ströme von Spaltenleitungen in die Zeilenleitung
    - ► Bei A voller Strom in der 1. Zeile → A erkannt
    - ► Bei B voller Strom in der 2. Zeile → B erkannt
    - ► Usw.

Lernmatrix für binäre Signale aus dem Jahre 1960

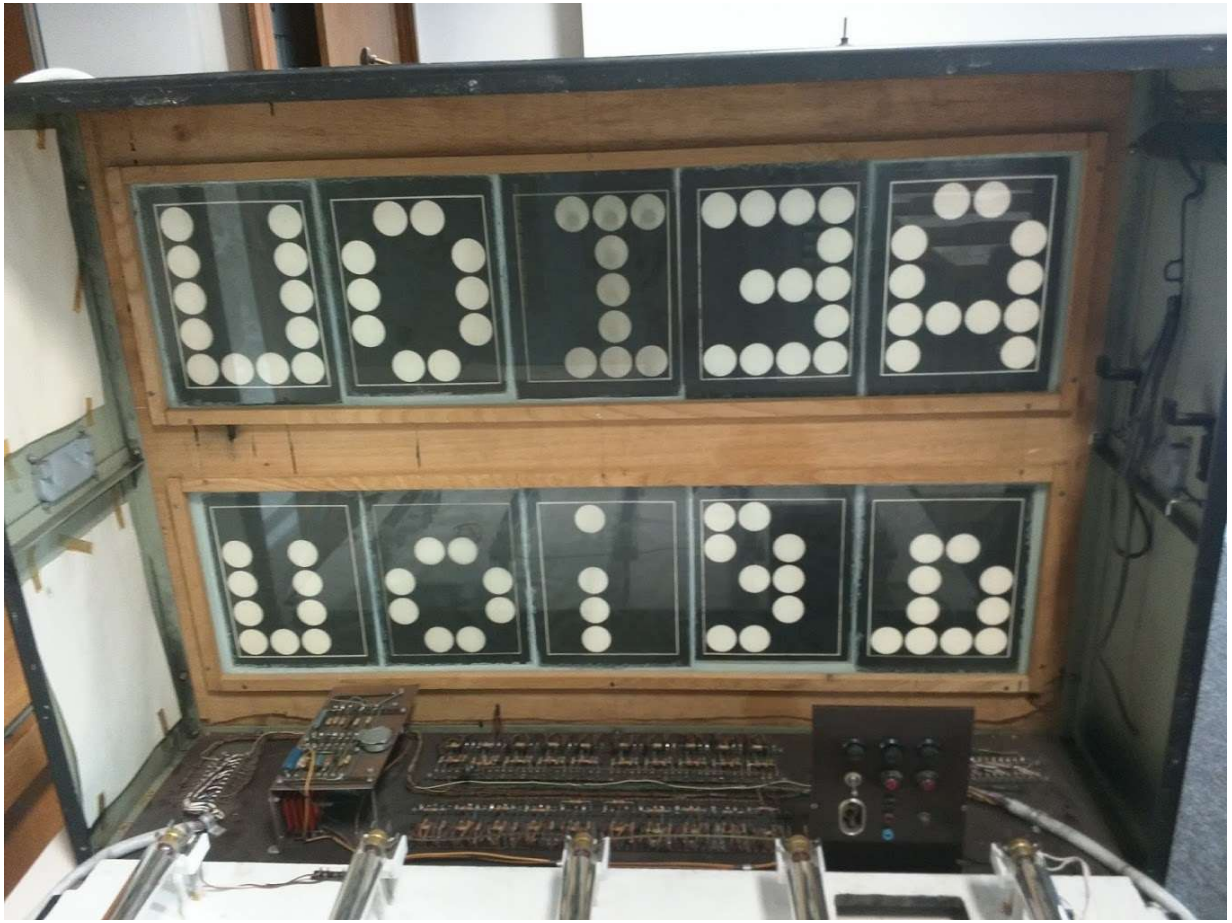
Vorstufe des Maschinellen Lernens



Anordnung		Ursache der Verknüpfungsänderung	Wirkung
Ferro- oder Ferrimagnetica	Ringkern Folienkern Transfluxor	Stromansteuerung	Änderung der remanenten Induktion und damit Kopplungsänderung
Ferridielektrika	Transpolarizer	Spannungsansteuerung	Änderung der Polarisation
kapazitive Anordnungen	Aufdampfkondensatoren Ag–SiO_2	Ausbrennen der Beläge	Kapazitätsänderung
	Metallpapier		
elektrochemische Zellen	Ag–AgBr–Ag	Bildung von Ag-Fäden	Leitwerterhöhung
	Ta–Ag-Elektrolyt	Auflösen von Ag, Bildung von Ta_2O_5	Leitwertverringerung
	„Memistor"	Cu-Ablagerung	Leitwerterhöhung
	„Solion"-Tetrode	Änderung der Jod-Ionenkonzentration	Leitwerterhöhung

Bild 106. Realisierungsmöglichkeiten für Lernmatrizen

Innenansicht der „Eingabe“ Box



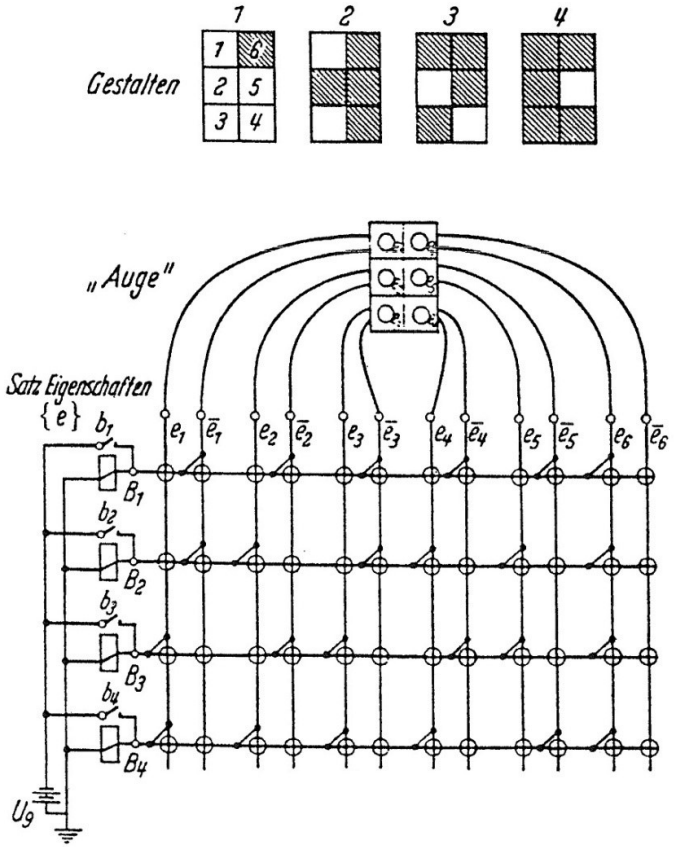
Karl Steinbuch

Lernmatrix



Die Lernmatrix: Weiterentwicklung

- neu an Steinbuchs Lernmatrix (Weiterentwicklung, analog)
  - „als Matrixelemente werden nicht nur Dioden mit der bekannten Kennlinie zugelassen, sondern auch Elemente mit ganz anderen Kennlinien“
  - d.h., dass sich die Ausgangssignale aus einer Summe von Strömen ergeben, wobei jeder Strom aus dem Eingangssignal e und dem zugehörigen Matrixelement besteht
- realisiert werden die Matrixelemente durch veränderbare Leitwerte



Prinzip der analogen Lernmatrix

Neural Network → Irrtümer

https://neil.fraser.name/writing/tank/

- **The Plan**
  - In den 1980er Jahren wollte das Pentagon Computer-Technologie einsetzen, um feindliche Panzer besser zu erkennen.
  - Der Computer würde ständig die Umgebung scannen auf mögliche Bedrohungen (z. B. einen feindlichen Panzer versteckt sich hinter einem Baum).
  - Computer für die sich wiederholende Aufgabe ohne Pause ist erst einmal eine gute Idee.
  - Computer sind aber in der Regel schlecht bei der Interpretation der Bilder.
  - Die einzige Möglichkeit zur Lösung des Problems war ein neuronales Netz zu beschäftigen.





Neural Network → Irrtümer

https://neil.fraser.name/writing/tank/

- **The Implementation**
  - Das Forscherteam hat 100 Fotos von Panzer versteckt hinter Bäumen aufgenommen und dann 100 Bilder ohne Panzer fotografiert.
  - Es nahm die Hälfte der Fotos aus jeder Gruppe und steckte sie in ein Depot zur Verwahrung, dann die andere Hälfte in ihre Mainframecomputer gescannt.
  - Ein immenses neuronales Netz wurde nun mit den Fotos gefüttert.
  - Natürlich waren seine Antworten zu Beginn völlig zufällig, da das Netzwerk nicht wusste, was los war oder was es tun soll.
  - Aber bei jedem Foto erzeugte das System eine Antwort und die Wissenschaftler quittierten, wenn es richtig oder falsch war.
  - Wenn es falsch war würde es nach dem Zufallsprinzip die Gewichtungen in seinem Netzwerk ändern, bis es die richtige Antwort gab.
  - Im Laufe der Zeit wurde es besser und besser bis schließlich jedes Foto korrekt war.
  - Es konnte auf allen Fotos richtig feststellen, ob ein Panzer hinter den Bäumen versteckt war.





Neural Network Irrtümer

https://neil.fraser.name/writing/tank/

- **Verifikation**
  - Aber die Wissenschaftler waren besorgt: hatte man tatsächlich einen Weg gefunden zu erkennen, ob ein Panzer auf dem Foto war, oder nur auswendig gelernt hatte, welche Fotos Panzer hatten und welche nicht?
  - Dies ist ein großes Problem mit neuronalen Netzen, nachdem sie trainiert worden sind,
    - Wir haben keine Ahnung wie die Antworten zustande kommen.
  - Zur Prüfung nahmen die Wissenschaftler die anderen Bilder aus dem Tresor und fütterten den Computer.
  - Diese hatte der Computer noch nie gesehen – dies wäre die große Bewährungsprobe.
  - Zu ihrer großen Erleichterung identifiziert das neuronale Netz korrekt auf jedem Foto ob Panzer oder nicht.
- **Independent testing**
  - Das Pentagon war sehr zufrieden mit diesem Ergebnis, aber immer noch ein bisschen misstrauisch.
  - Es beauftragt einen anderen Satz von Fotos (die Hälfte mit Panzern) die der Computer durch das neuronale Netz interpretierte.
    - ► Die Ergebnisse waren völlig zufällig.
  - Lange Zeit konnte niemand herausfinden, warum.
    - Niemand konnte nachvollziehen, wie die neuronalen Netze gelernt hatten.





Neural Network Irrtümer

https://neil.fraser.name/writing/tank/

- **Grey skies for the US military**
  - Schließlich bemerkte jemand, dass in den ursprünglichen Satz von 200 Fotos, alle Bilder mit Panzern an einem bewölkten Tag aufgenommen worden waren, während die Bilder ohne Panzer an einem sonnigen Tag genommen worden waren.
  - Das neuronale Netz hat die zwei Gruppen von Fotos getrennt und die naheliegendste Möglichkeit es zu tun verwendet:
    - ► nicht auf der Suche nach einem getarnten Panzer hinter Bäumen versteckt, sondern lediglich anhand der Farbe des Himmels.
  - Das Militär war nun stolzer Besitzer einer Multimillionen-Dollar-Mainframe-Computer, der sagen konnte, wenn es sonnig war.





Klassische Programmierung vs. Maschinelles Lernen

- Beim klassischen Algorithmus legen Programmierer eine bestimmte Logik fest, die sich im Laufe der Zeit nicht ändert.
  - Die Maschine verarbeitet dann die Daten entsprechend des Schemas.
  - Bei Abweichungen oder unbekannten Konstellationen kann der klassische Algorithmus die Daten nicht verarbeiten und gibt einen Exception-/Ausnahmefehler zurück.
  - Die Beziehungen zwischen Variablen sind fest definiert und ändern sich nicht.

- Beim Maschinellen Lernen kann sich die Logik im Laufe der Zeit anpassen, die Maschinen “lernen” und Künstliche Intelligenz entsteht.
  - Programme und Algorithmen verbessern sich selbst.
  - Beziehungen zwischen den Variablen sind nicht fest, sondern ändern sich im Laufe der Zeit.
  - Auch Algorithmen sind nicht fest definiert, sondern entstehen während des Lernprozesses.
  - Entsprechend geht es bei Machine Learning hauptsächlich um die Mustererkennung und die Erkennung von Beziehungen zwischen den Entitäten.

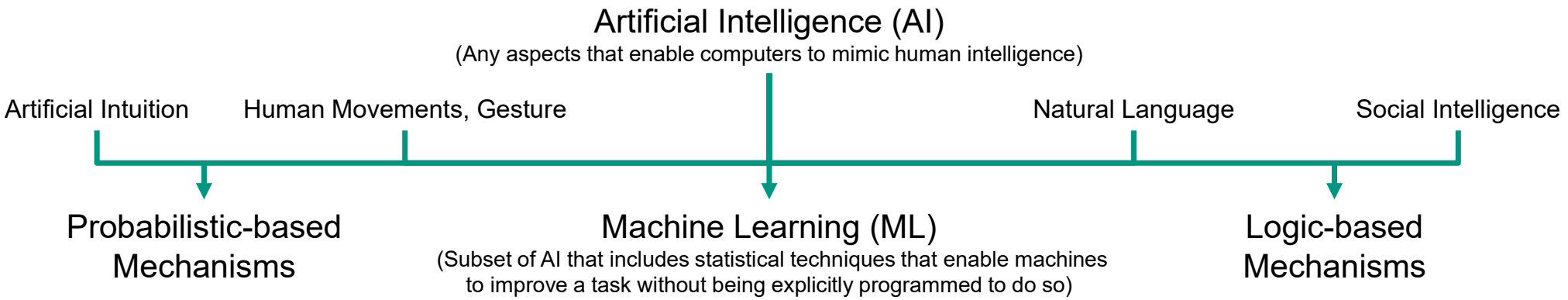
Klassische Programmierung vs. Maschinelles Lernen

- Beim klassischen Algorithmus legen Programmierer eine bestimmte Logik fest, die sich im Laufe der Zeit nicht ändert.
  - Die Maschine verarbeitet dann die Daten entsprechend des Schemas.
  - Bei Abweichungen oder unbekannten Konstellationen kann der klassische Algorithmus die Daten nicht verarbeiten und gibt einen Exception-/Ausnahmefehler zurück.
  - Die Beziehungen zwischen Variablen sind fest definiert und ändern sich nicht.

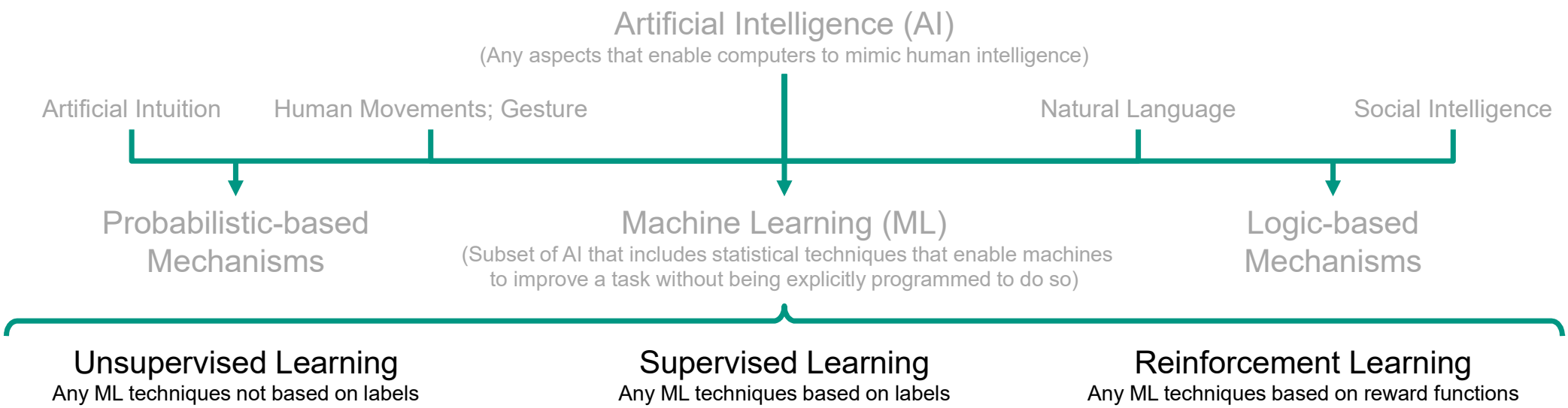
- Beim Maschinellen Lernen kann sich die Logik im Laufe der Zeit anpassen, die Maschinen “lernen” und Künstliche Intelligenz entsteht.
  - Programme und Algorithmen verbessern sich selbst.
  - Beziehungen zwischen den Variablen sind nicht fest, sondern ändern sich im Laufe der Zeit.
  - Auch Algorithmen sind nicht fest definiert, sondern entstehen während des Lernprozesses.
  - Entsprechend geht es bei Machine Learning hauptsächlich um die Mustererkennung und die Erkennung von Beziehungen zwischen den Entitäten.

- Im Grunde genommen gibt es zwei hauptsächliche Ansätze:
  - Überwachtes (supervised) Lernen und Unüberwachtes (unsupervised) Lernen.
  - Bei der Künstlichen Intelligenz (KI) werden teilweise auch beide Ansätze kombiniert.

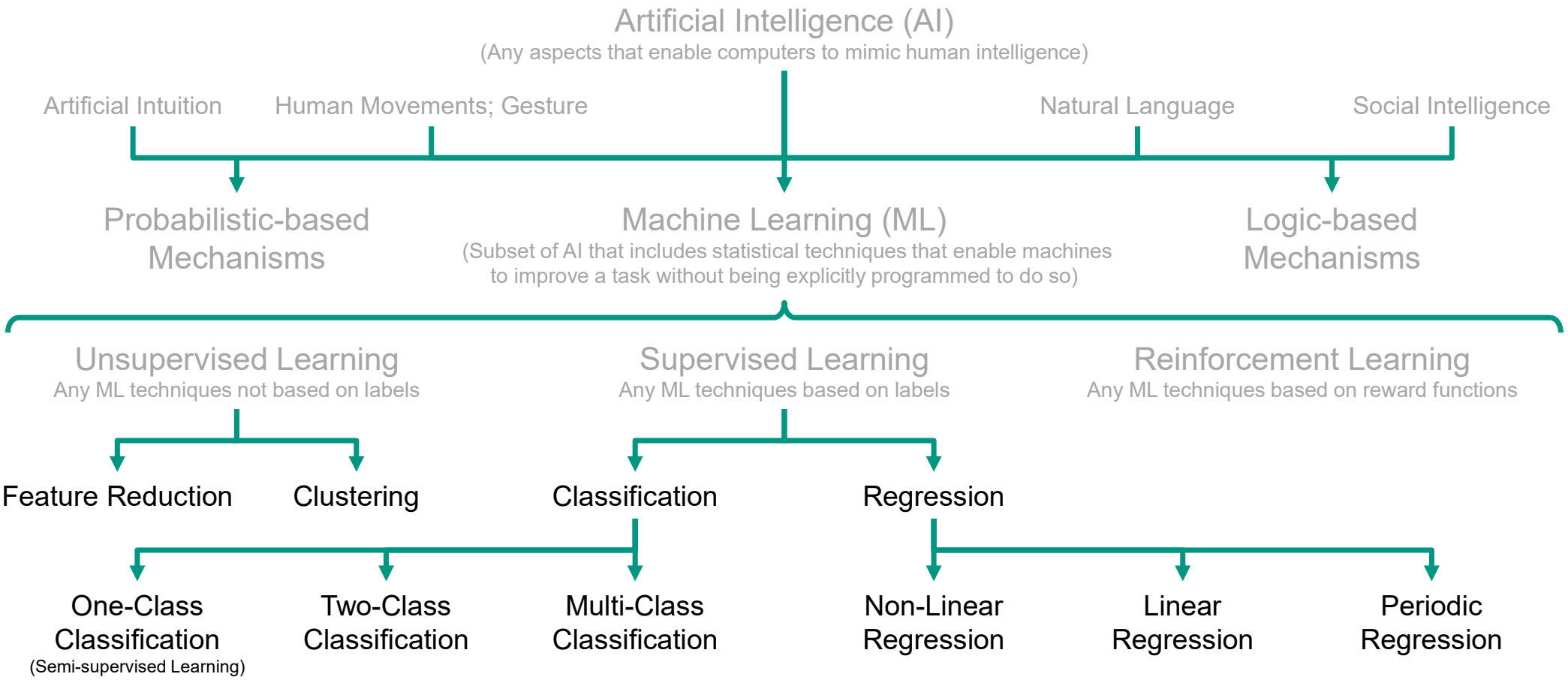
Artificial Intelligence und Machine Learning



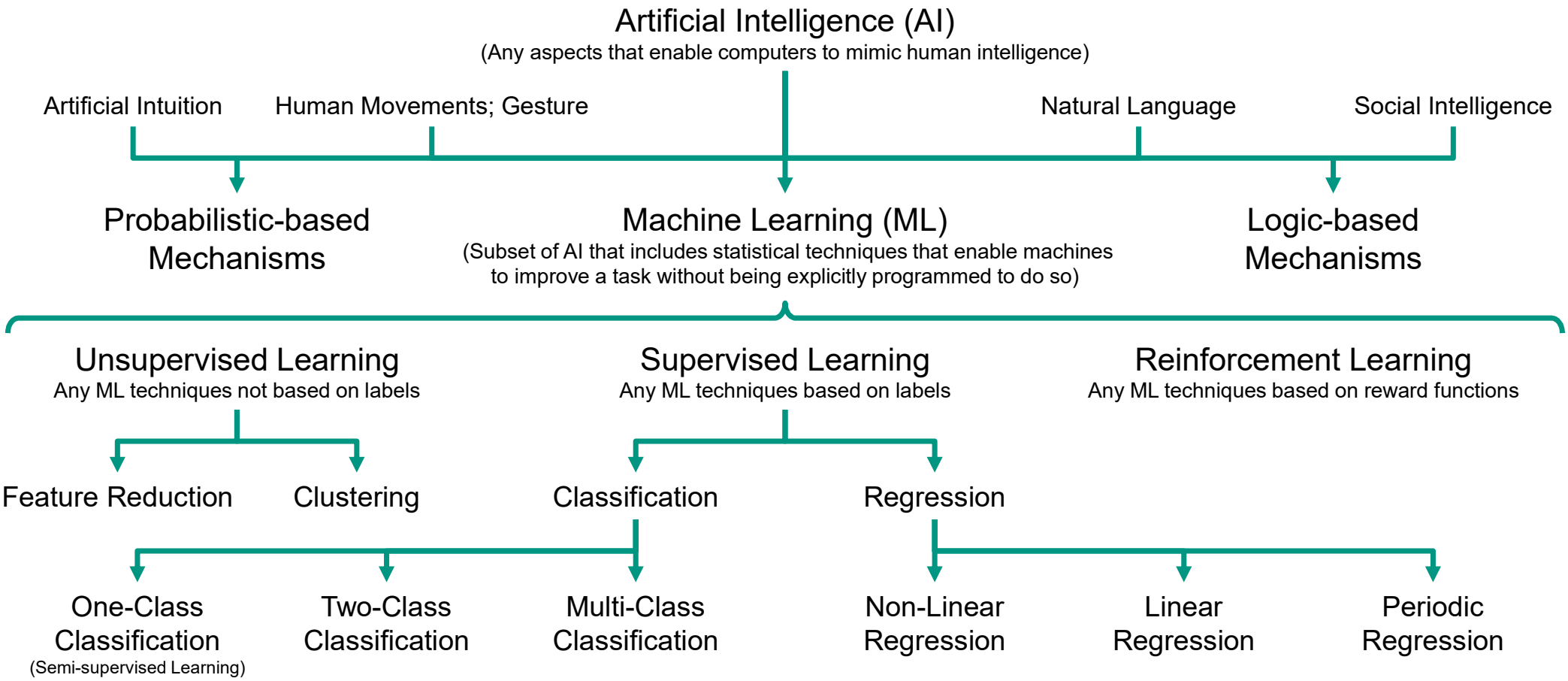
Lernmethoden



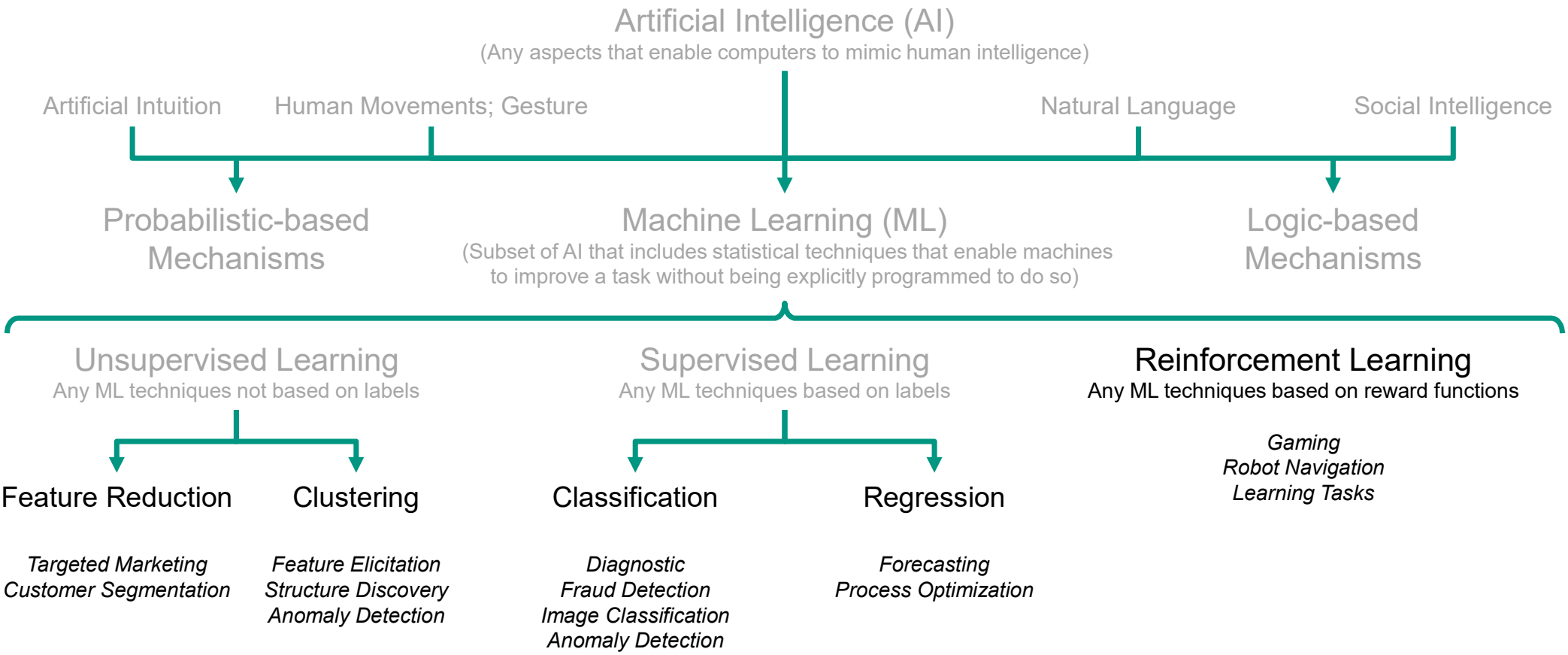
Problemklassen

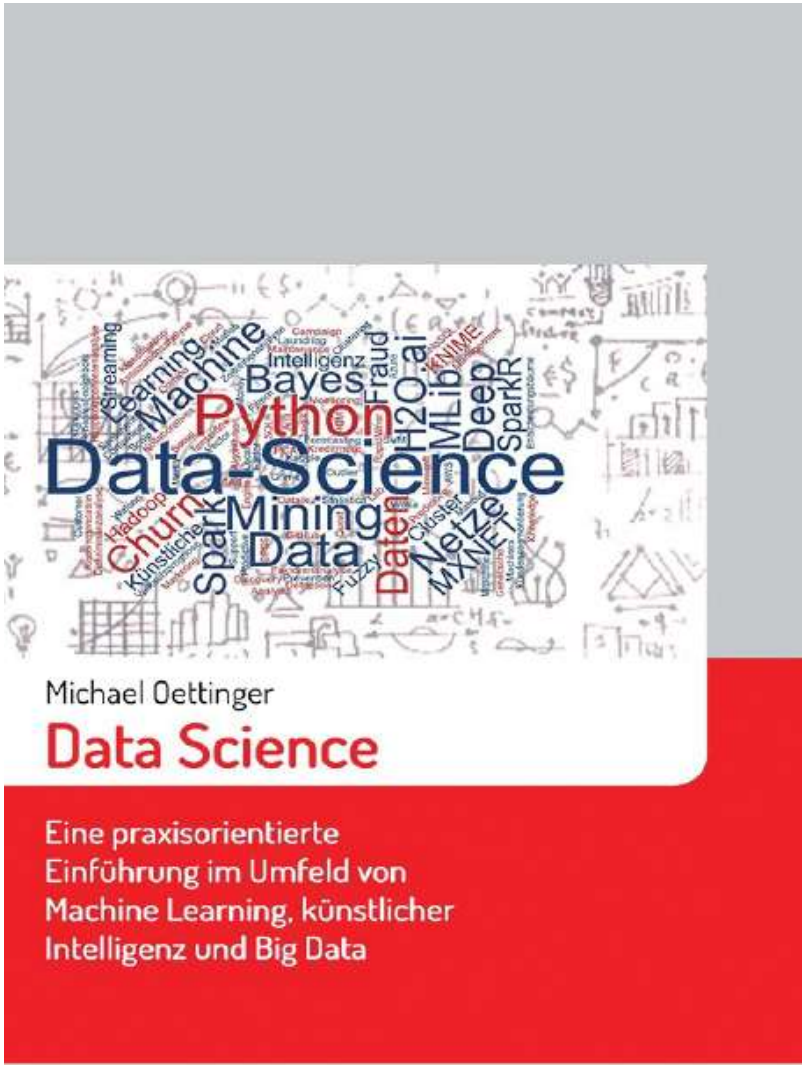


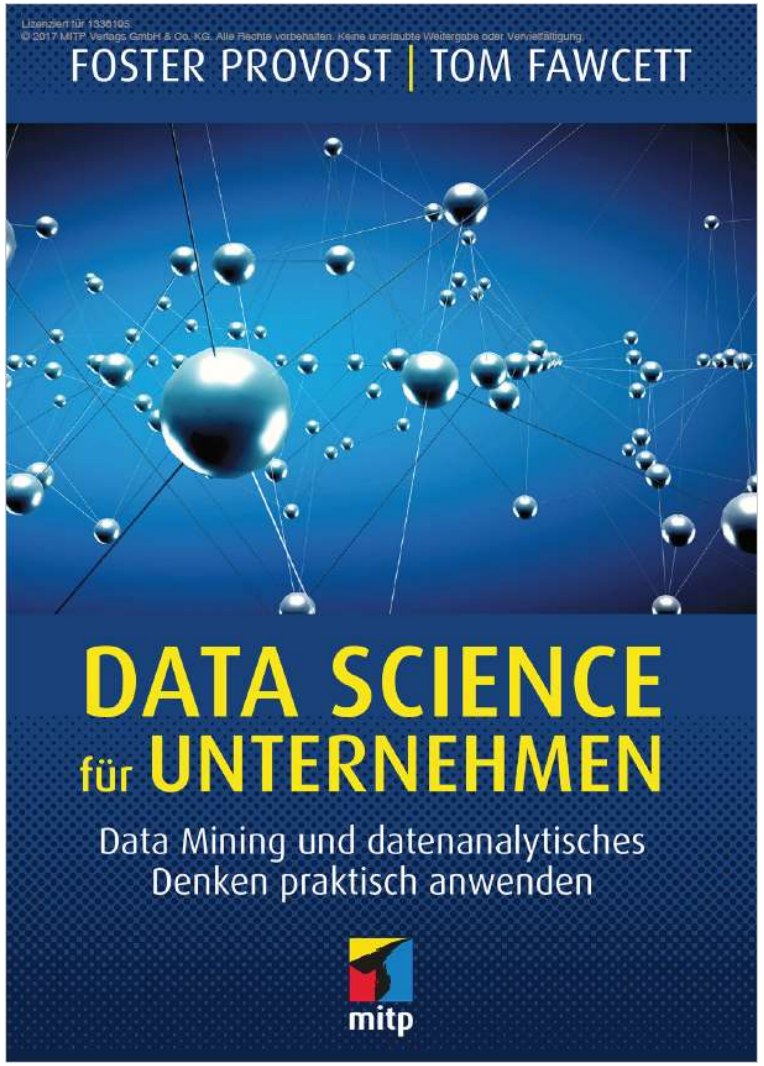
Problemklassen



Anwendungen



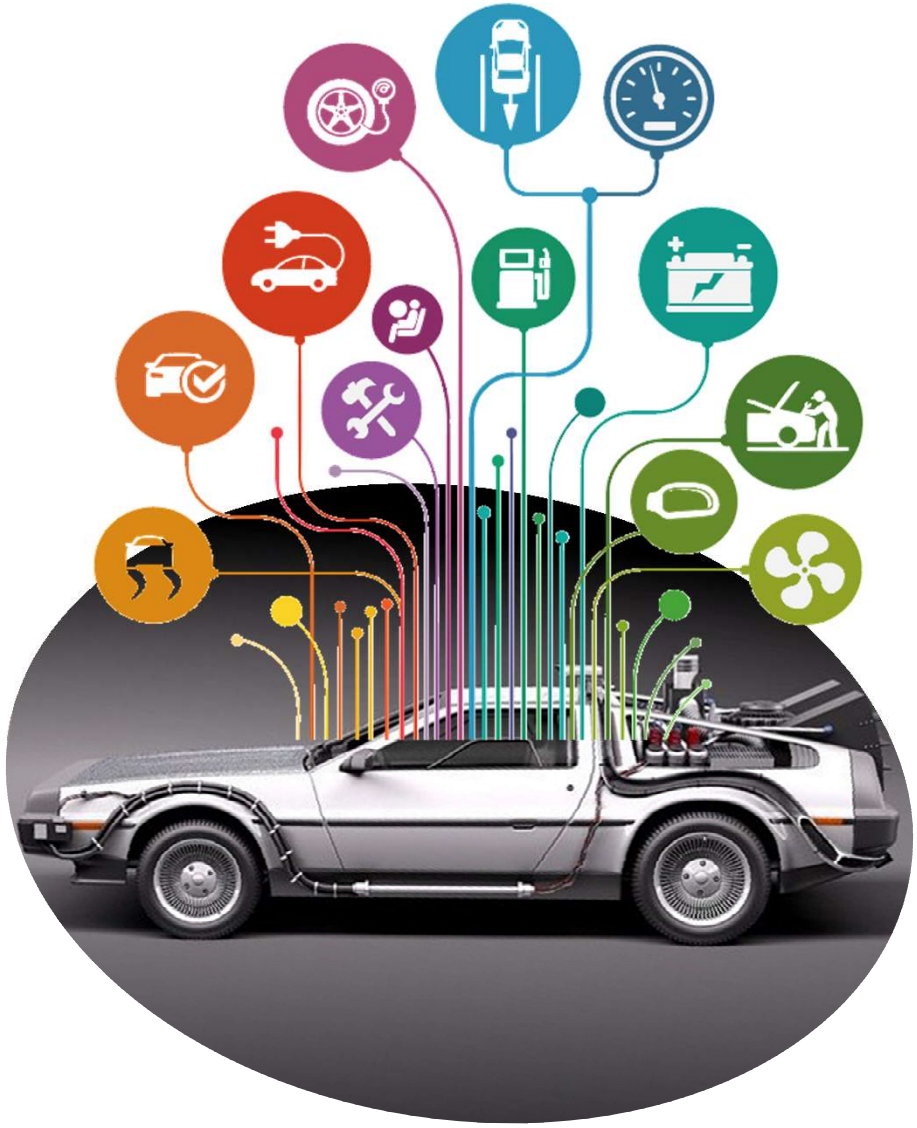

Machine
Learning
Python
Data-Science
Bayes
Intelligenz
Fraud
H2O.ai
MLlib
Deep
SparkR
KNIME
Mining
Data
Daten
Netze
Spark
Hadoop
Churn
Künstliche
Cluster
Fuzzy
Michael Oettinger
Data Science
Eine praxisorientierte Einführung im Umfeld von Machine Learning, künstlicher Intelligenz und Big Data


Lizenziert für 1336105
© 2017 MITP Verlags GmbH & Co. KG. Alle Rechte vorbehalten. Keine unerlaubte Weitergabe oder Vervielfältigung
FOSTER PROVOST | TOM FAWCETT
DATA SCIENCE
für UNTERNEHMEN
Data Mining und datenanalytisches Denken praktisch anwenden
mitp

Daten

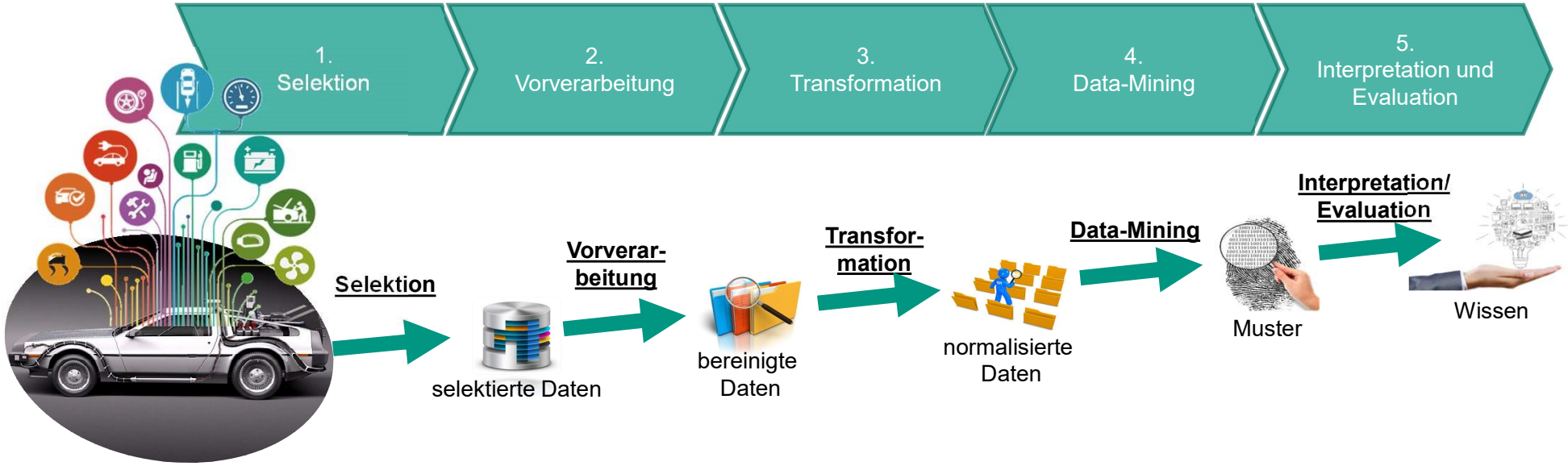

BAU DEN DELOREAN AUS
ZURÜCK IN DIE ZUKUNFT
UNIVERSAL
eaglemoss
OUTATIME

Daten



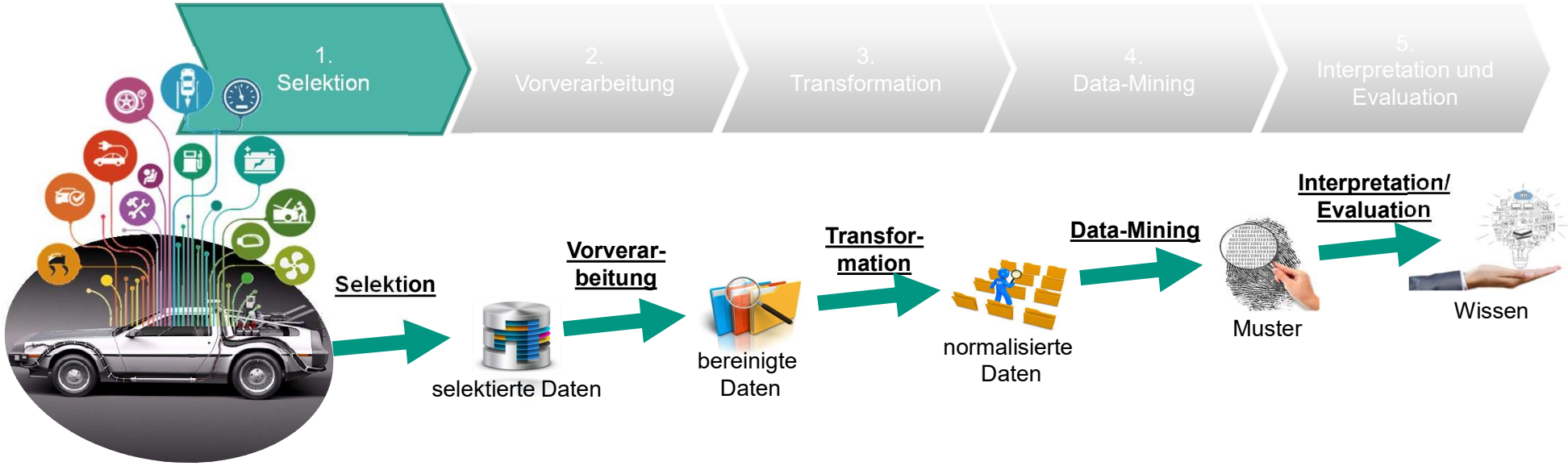
Data Science als Prozess – KDD

KDD – Knowledge Discovery in Databases



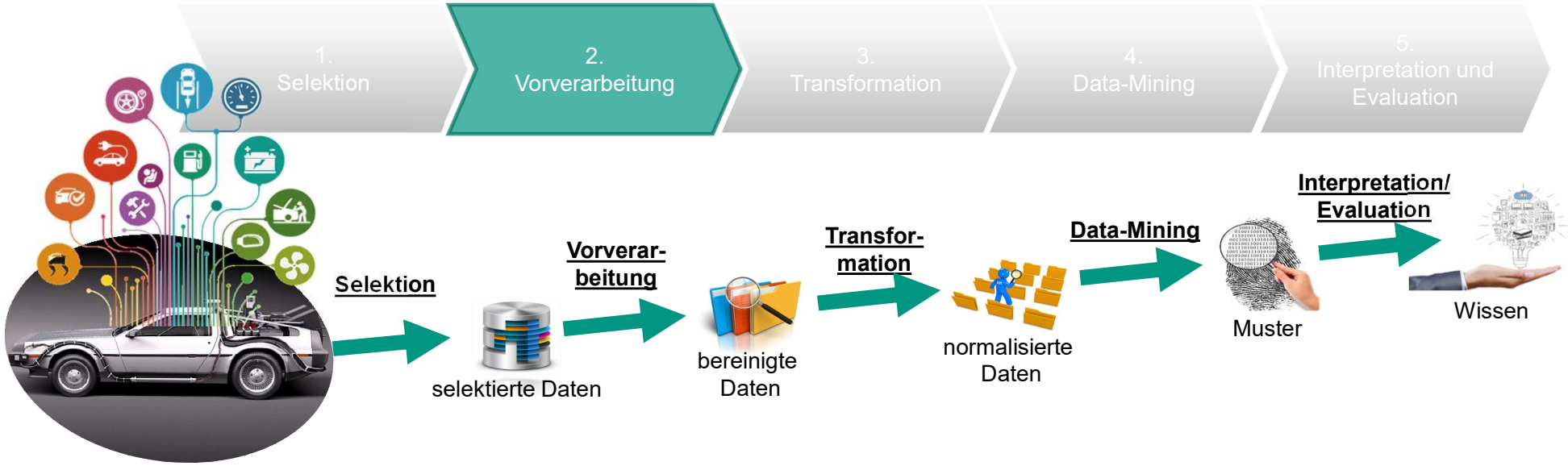
- **KDD ist der Prozess der (semi-)automatischen Extraktion von Mustern aus Datenbanken, die …**
  - … *gültig*
  - … *bisher unbekannt*
  - … und potentiell *nützlich* und *verständlich* sind

Data Science als Prozess – KDD



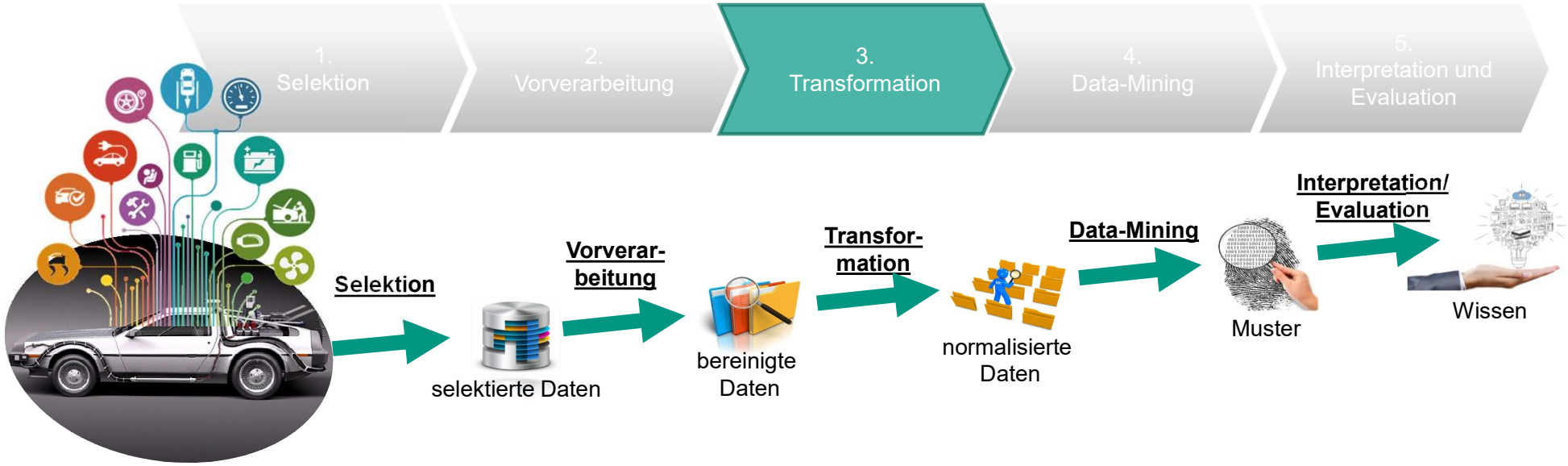
- **Erstellung eines Quelldatensatzes:**
  - Auswahl eines bestimmten Datensatzes
  - Fokussierung auf eine Teilmenge von Merkmalen oder Datenproben, bei denen die Untersuchung durchgeführt werden soll

Data Science als Prozess – KDD



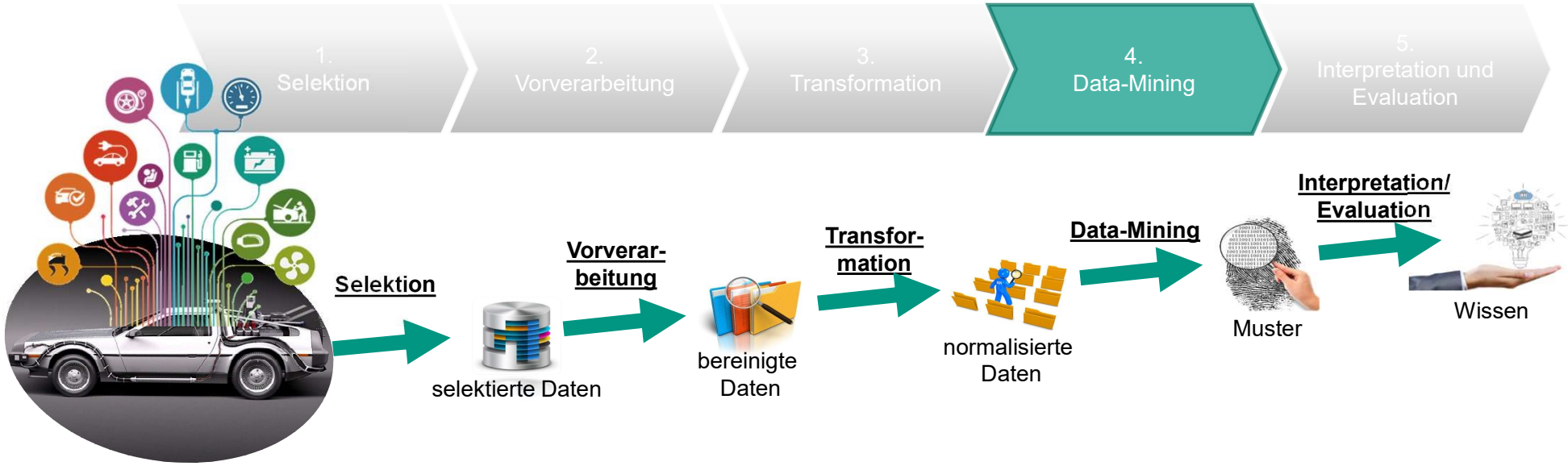
- **Vorverarbeitung:**
  - Beseitigung von Rauschen, falls erforderlich
  - Sammeln der notwendigen Informationen zur Berücksichtigung von Rauschen
  - Strategien für den Umgang mit fehlenden Datenfeldern
  - Berücksichtigung von Zeitreiheninformationen und bekannten Veränderungen

Data Science als Prozess – KDD



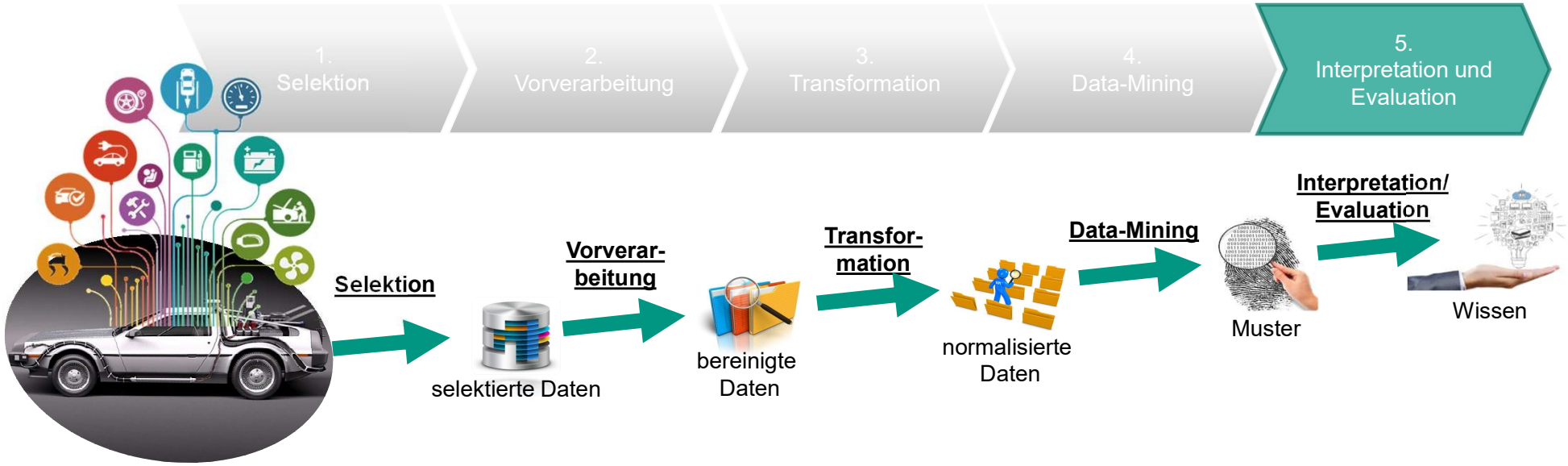
- **Datenreduktion und -projektion:**
  - Nützliche Funktionen finden, um die Daten je nach Ziel der Aufgabe darzustellen
  - Mit Dimensionalitätsreduktions- oder Transformationsmethoden die effektive Anzahl der betrachteten Merkmale reduzieren
  - Invariante Darstellungen für die Daten finden

Data Science als Prozess – KDD



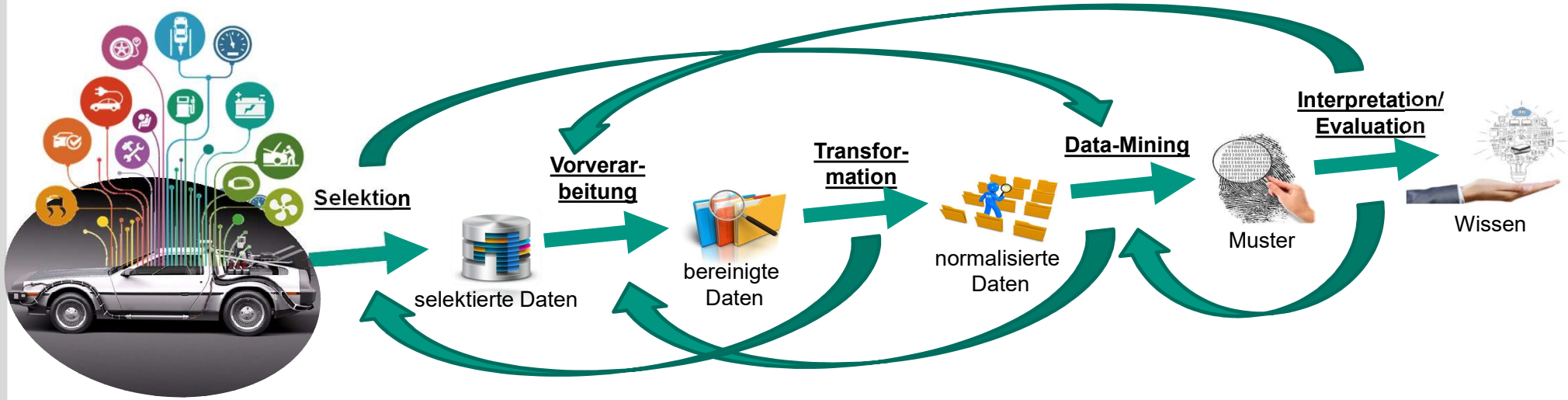
- **Data-Mining:**
  - Suchen nach Mustern von Interesse in einer bestimmten Darstellungsform
    - Klassifizierungsregeln, Entscheidungsbäume, Regression, Clustering …
  - Benutzer kann durch passende Modell- und Parameterwahl unterstützen
    - z.B. Endbenutzer mehr am Verständnis des Modells interessiert als an Genauigkeit

Data Science als Prozess – KDD



- **Interpretieren der gefundenen Muster/Modell**
  - Visualisierung der extrahierten Muster und Modelle
  - Visualisierung der Daten anhand der extrahierten Modelle
- **Evaluation der gefundenen Muster**
  - Oft: Metrik für Sicherheit / Güte des Ergebnisses definiert
  - Problem: „Neuheit“, „Verständlichkeit“, „Interessantheit“ sind subjektiv!

Data Science in der Realität



- **KDD-Prozess liefert Beschreibung der Basis-Aufgaben**
  - Für konsequente Umsetzung in der Praxis „Implementierung“ und genauere Vorgabe der Schritte nötig
  - Zyklischer Charakter von Data-Mining-Projekten nicht präzise beschrieben

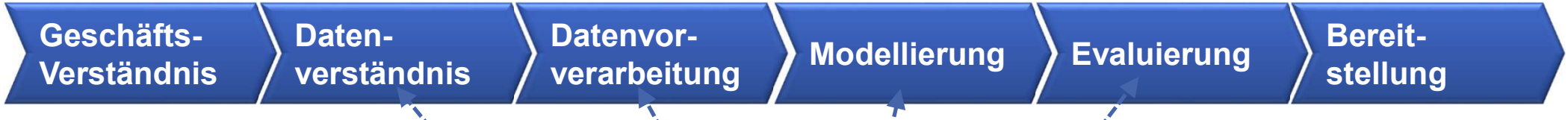
Führte zur Entwicklung von

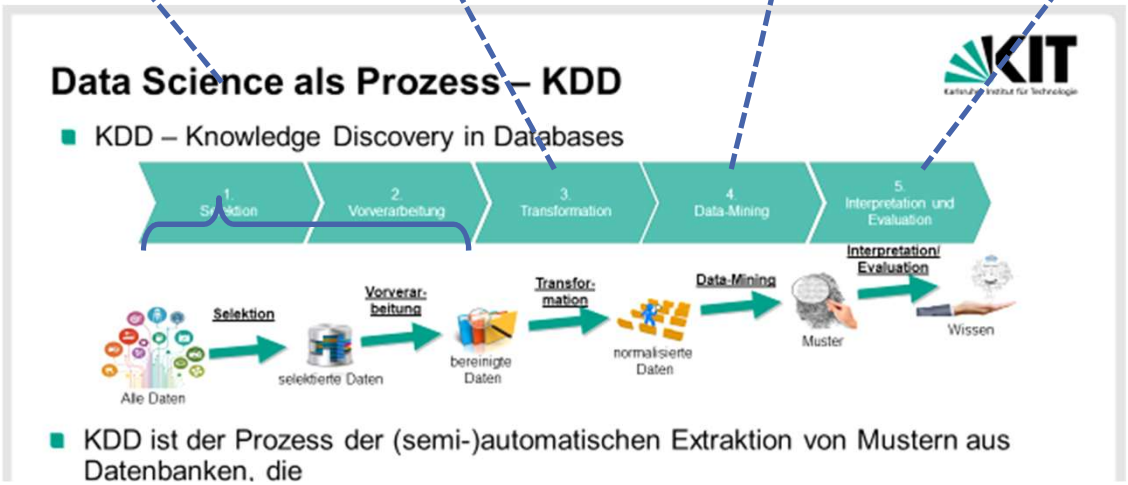
CRISP-DM
CROSS INDUSTRY STANDARD PROCESS FOR DATA MINING

Data Science als Prozess – CRISP-DM

- **Der komplette Data Science Prozess**
- **Standard für das Vorgehen in einem Projekt „Cross Industry Standard Process for Data Mining“ (CRISP-DM) Zusammengesetzt aus sechs Phasen:**

CRISP-DM
CROSS INDUSTRY STANDARD PROCESS FOR DATA MINING





Data Science als Prozess – CRISP-DM

- Der komplette Data Science Prozess
- Standard für das Vorgehen in einem Projekt
  „Cross Industry Standard Process for Data Mining“ (CRISP-DM)
  Zusammengesetzt aus sechs Phasen:

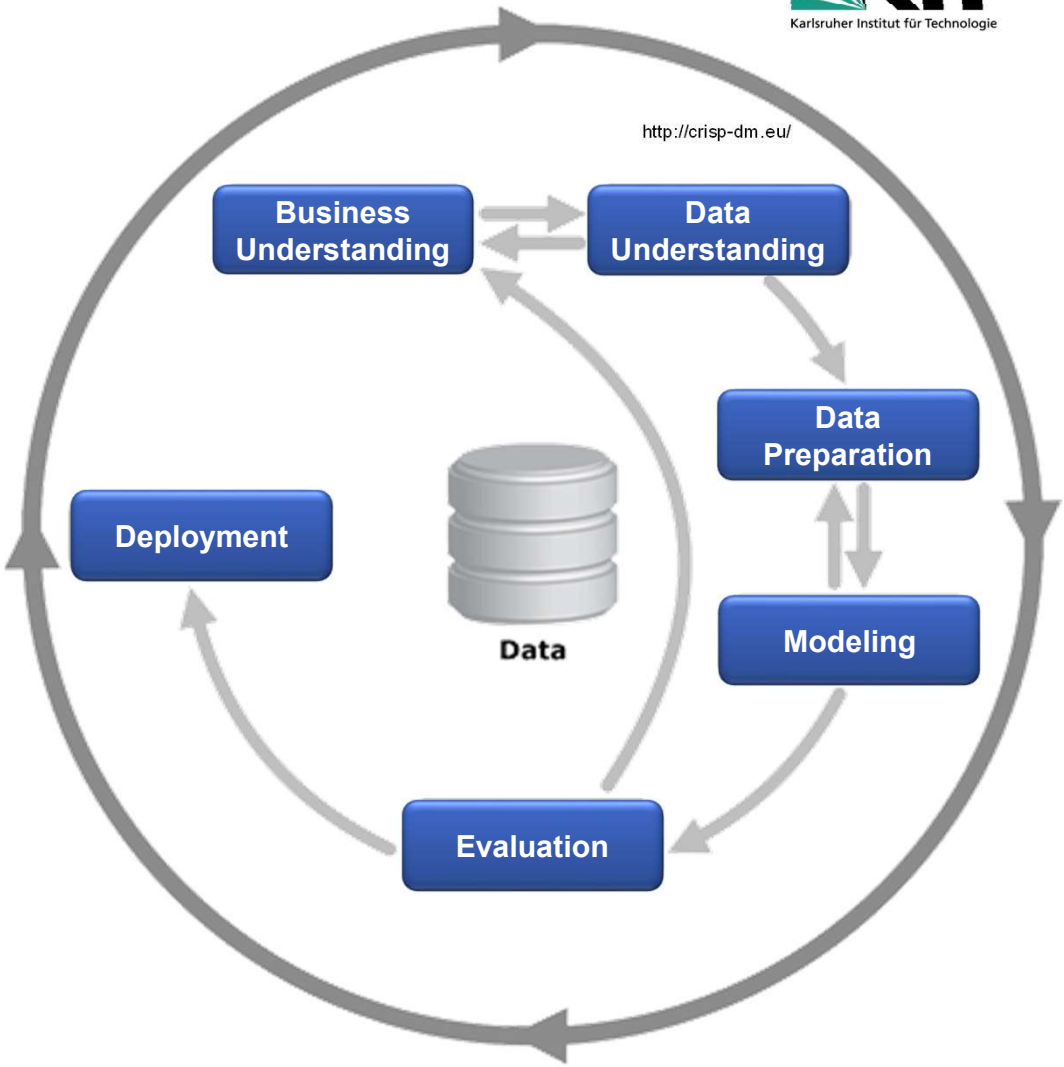


- Dieses branchenübergreifende Prozess-Modell wurde ab 1996 im Rahmen eines EU-Förderprojekts entwickelt, und zwar von so namhaften Teilnehmern wie DaimlerChrysler (damals noch Daimler-Benz) und **SPSS** (damals ISL). Es ist nicht eigentumsrechtlich geschützt.
- Standard für das Vorgehen in einem Projekt in sechs Phasen

CRISP-DM im Überblick

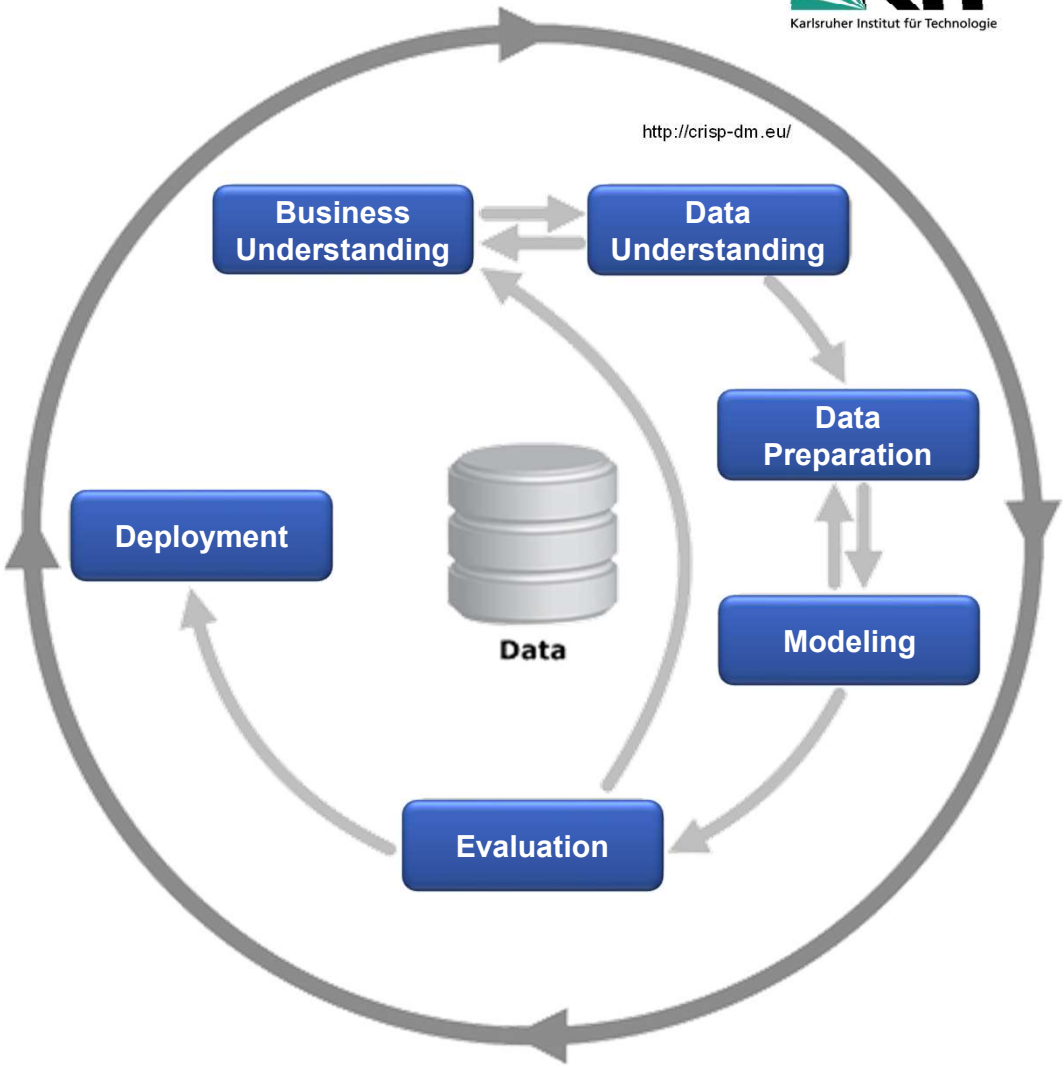
- Im KDD ist die Ablaufreihenfolge in der Praxis nicht strikt, sondern kann in der Praxis zwischen den einzelnen Phasen hin und her springen
  - Dem trägt CRISP Rechnung!
- Es werden sechs Phasen in Data Mining-Projekten unterschieden, wobei man sich diese Phasen nicht als einmaligen, sequentiellen Ablauf vorstellen sollte. Häufig muss zwischen verschiedenen Phasen hin- und her gewechselt werden.

  Kommt durch die Pfeile zum Ausdruck



Typische Aufwände

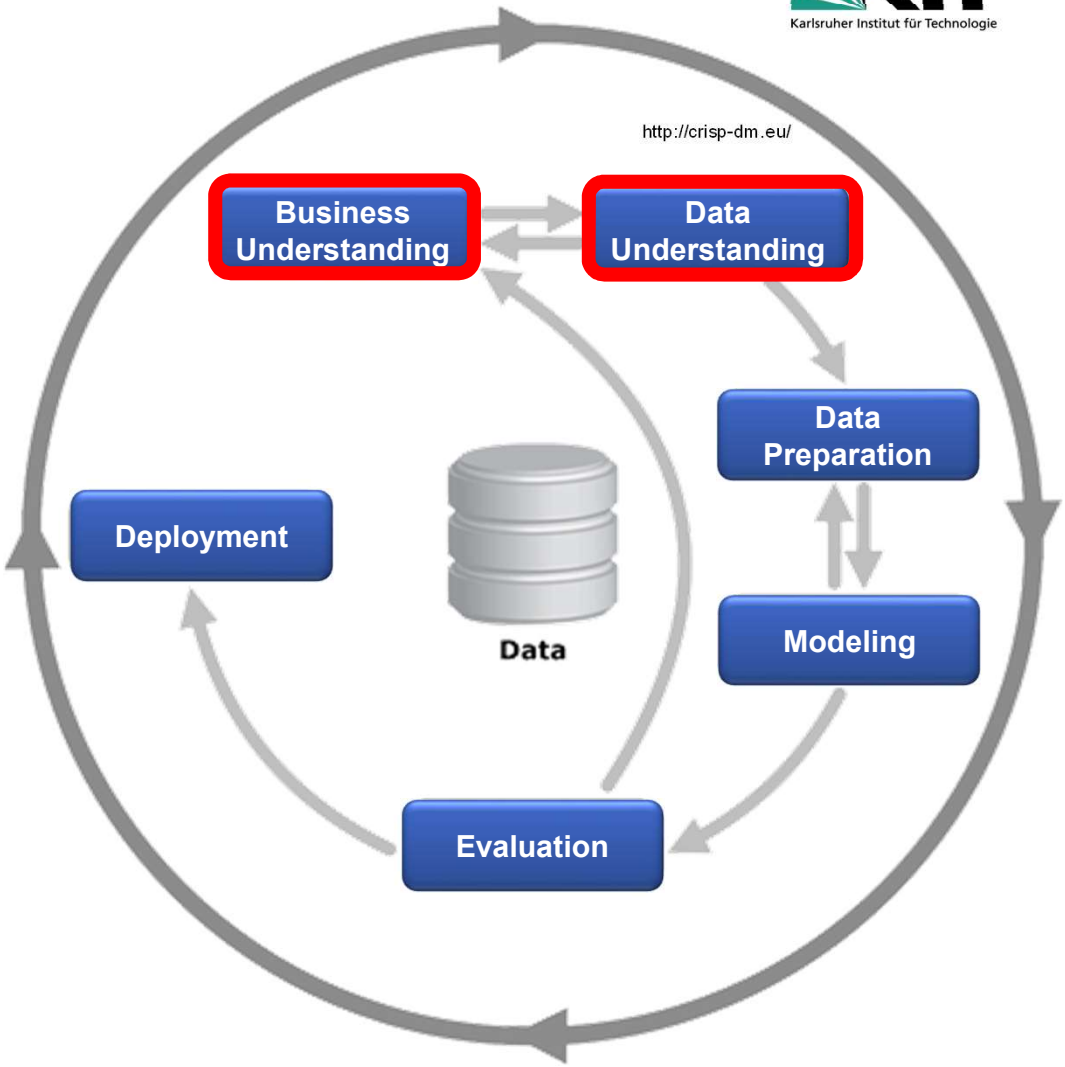
- 20-30% Data Understanding
- 50-70% Data Preparation
- 10-20% Modeling and Evaluation
- 5-10% Deployment



CRISP-DM im Überblick

Phasen in CRISP-DM (1 & 2)

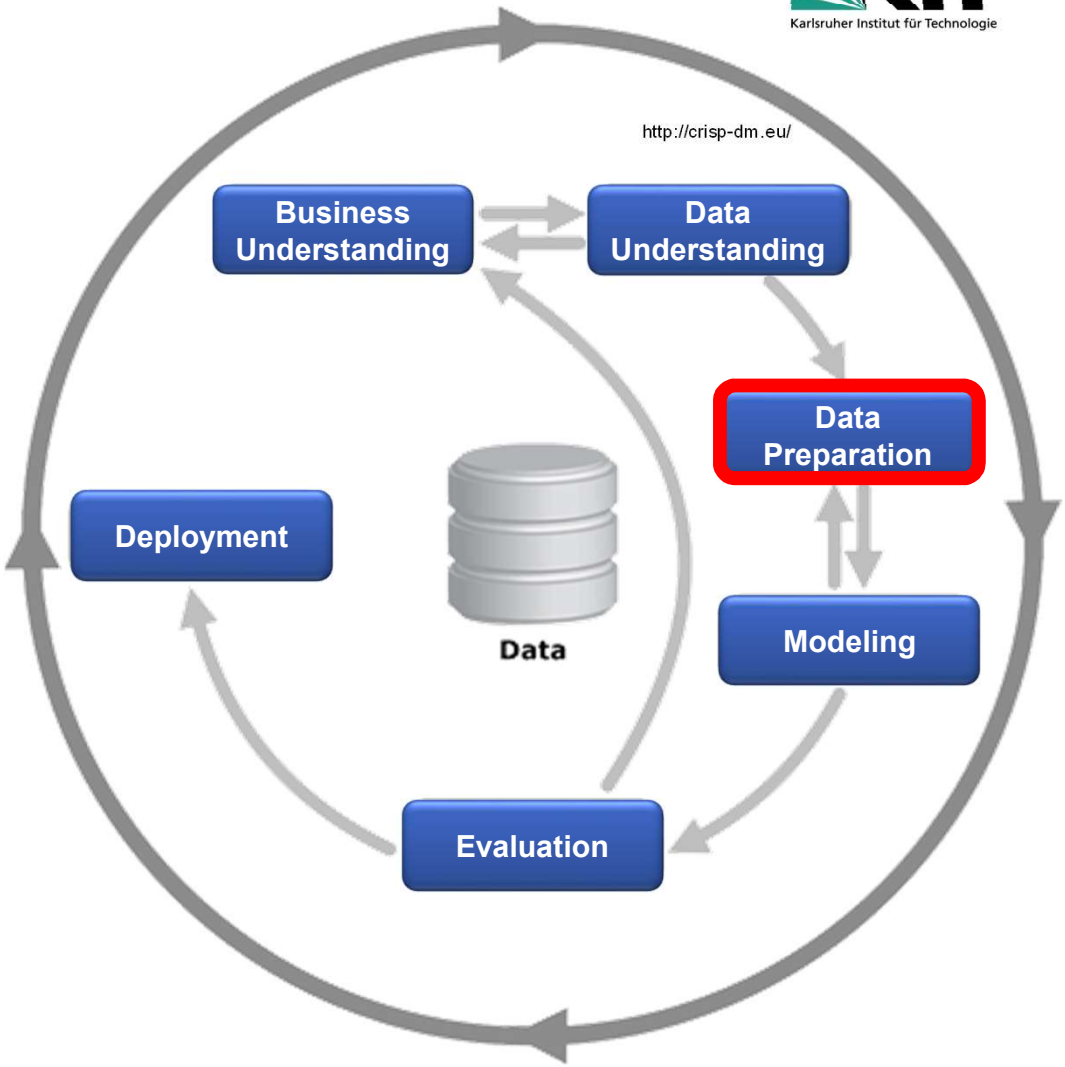
- Business Understanding (Geschäftsverständnis)
  - Festlegung der Ziele und Anforderungen
  - Ableitung der konkreten Aufgabenstellung und der groben Vorgehensweise
  - Aufstellung von Erfolgskriterien
- Data Understanding (Datenverständnis)
  - Datensammlung bzw. erste Sichtung der zur Verfügung stehenden Daten
  - Ermittlung möglicher Probleme mit Datenqualität
  - Outlier entdecken (→ Kapitel „Anomalie-Erkennung“)



CRISP-DM im Überblick

Phasen in CRISP-DM (3)

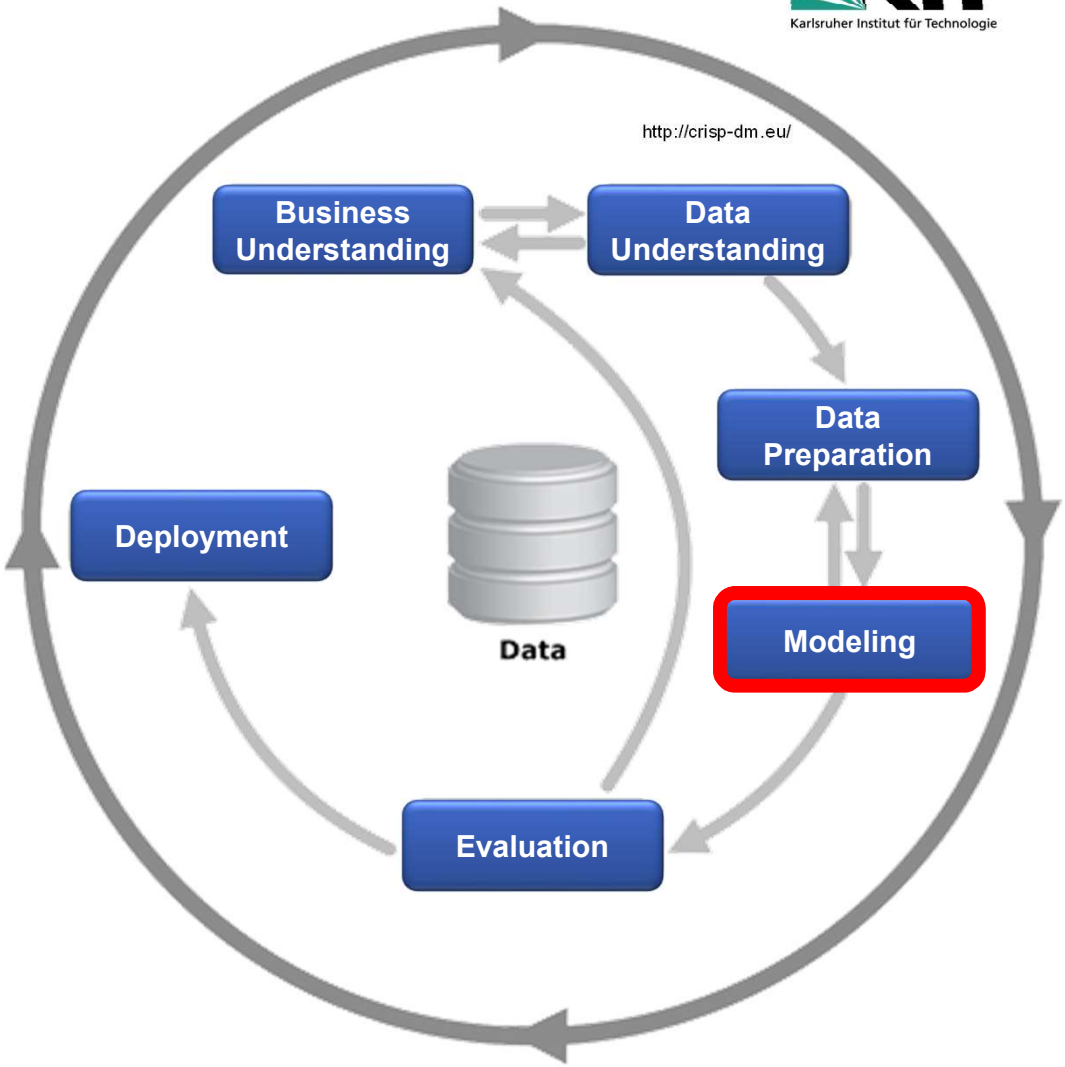
- Data Preparation (Datenvorverarbeitung)
  - Benötigt häufig bis 70-80% des Aufwands von Data Mining
  - Beurteilung
  - Konsolidierung und Säuberung
    - Tabellenlinks, fehlende Werte, …
  - Datenauswahl
    - Ausreißer
    - Sampling
    - Variablen
  - Transformationen – Erzeugung neuer Variablen



CRISP-DM im Überblick

Phasen in CRISP-DM (4)

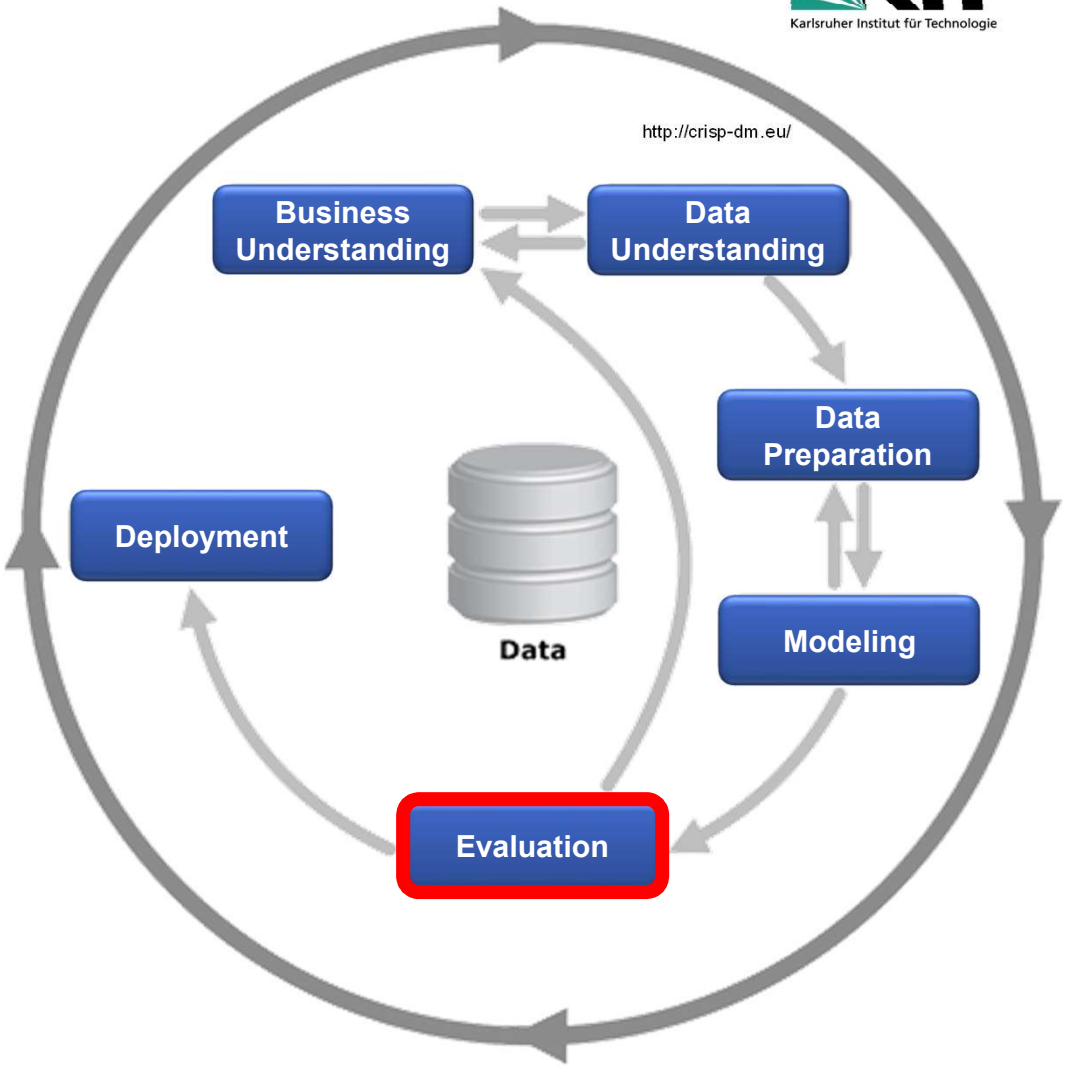
- Modeling (Modellierung)
  - Auswahl der Modellierungstechnik wird auf Basis des erstellten Data Mining Ziels und der Daten getroffen
  - Modellbildung ist ein iterativer Prozess – Unterscheidung zwischen überwachten und unüberwachten Lernen
    - Modelle können beschreibend oder vorhersagend sein
  - Vielzahl von Verfahren einsetzbar
    - Logistische Regression
    - KNN (K-nearest-neighbor)
    - K-means und weitere Cluster-Algorithmen
    - Neuronale Netze
    - Support Vector Machines
    - …



CRISP-DM im Überblick

Phasen in CRISP-DM (5)

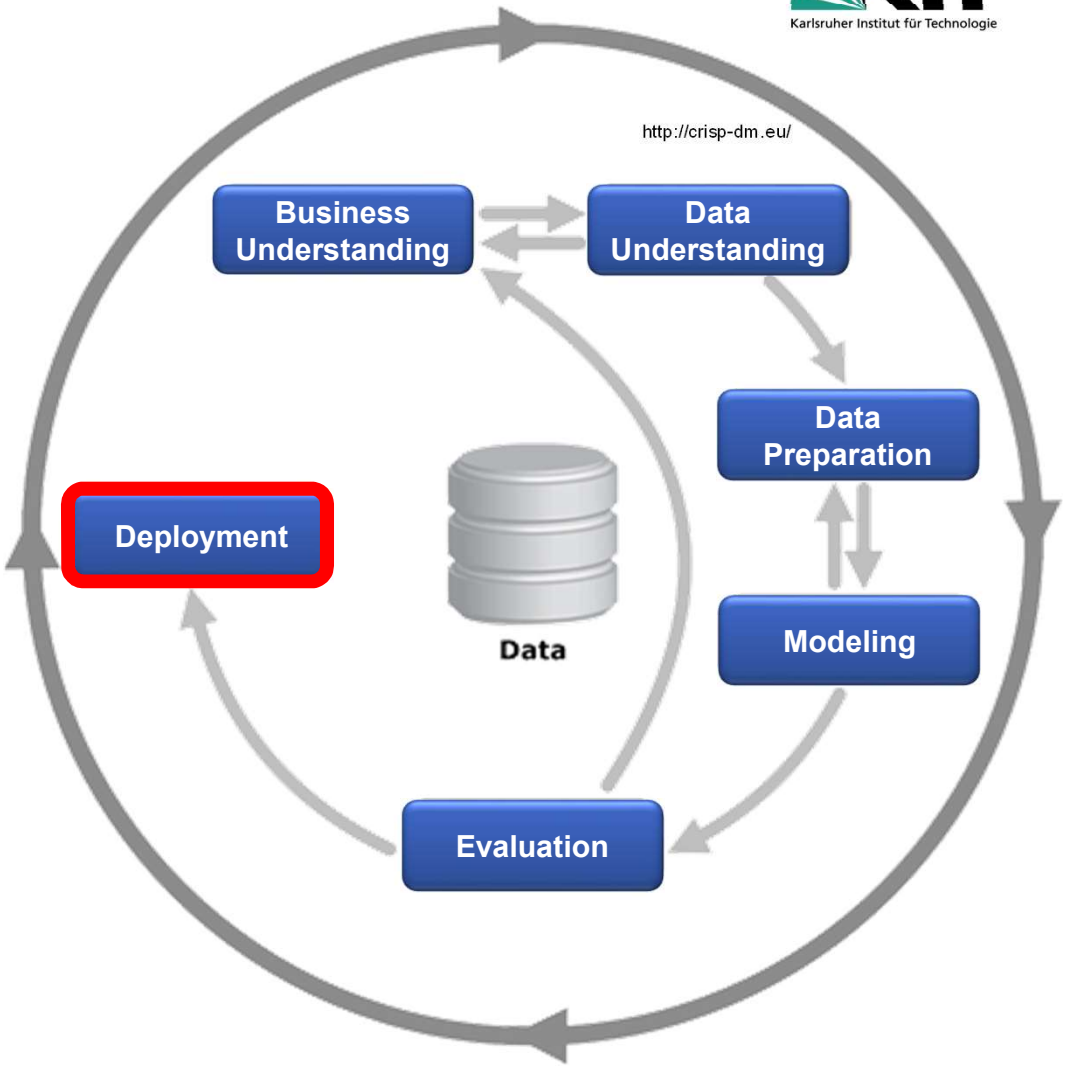
- Modell Evaluation (Evaluierung)
  - Auswahl und Evaluation des Modells
  - Evaluationsmethoden und –kriterien hängen vom jeweilig verwendeten Modelltyp ab
    - Fehlermatrix für Klassifikationsmodelle
    - Mittlerer quadratischer Fehler bei Regressionsmodellen
  - Interpretation des Modells:
    - Wichtig oder unwichtig
    - Einfach oder schwer/kompliziert
  - hängt jeweils vom verwendeten Modelltyp und gesetzten Zielen ab



CRISP-DM im Überblick

Phasen in CRISP-DM (6)

- Deployment (Bereitstellung)
  - Bestimmung wie die Ergebnisse genutzt werden sollen und können
  - Wer wird sie nutzen
  - Wie oft werden sie genutzt
  - Anwendung der Ergebnisse durch
    - Vorhersage auf Datenbanken
    - Anwendung als (Geschäfts-)regeln
    - Interaktive Vorhersagen
    - Aufsetzen Präventivmaßnahmen (Versicherung)
    - Identifizierung auffälliger Transaktionen

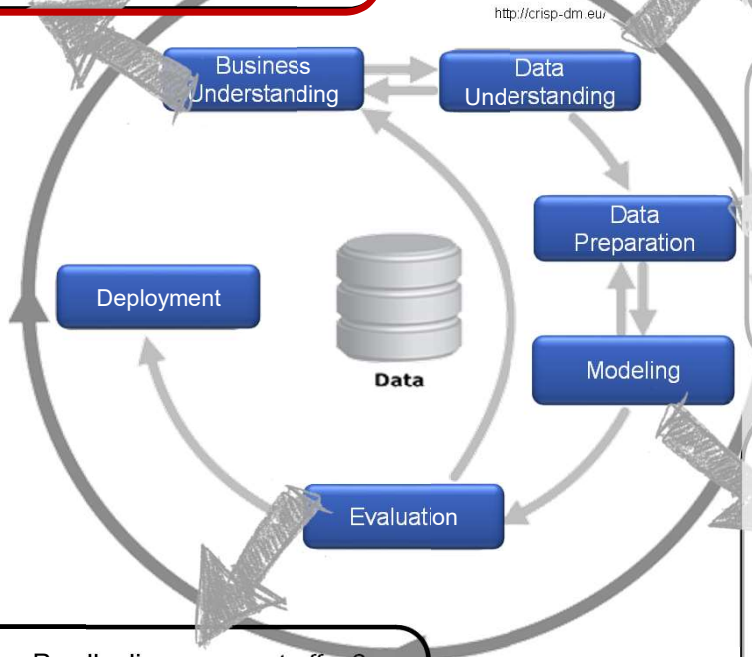


CRISP-DM im Detail: Business Understanding

- Zielsetzung
- Rand-bedingungen

Analytic Aproach

- Regression
- Klassifikation
- Clustering
- Recommendation
- Outlier



Ergebnisse

Evaluate Results

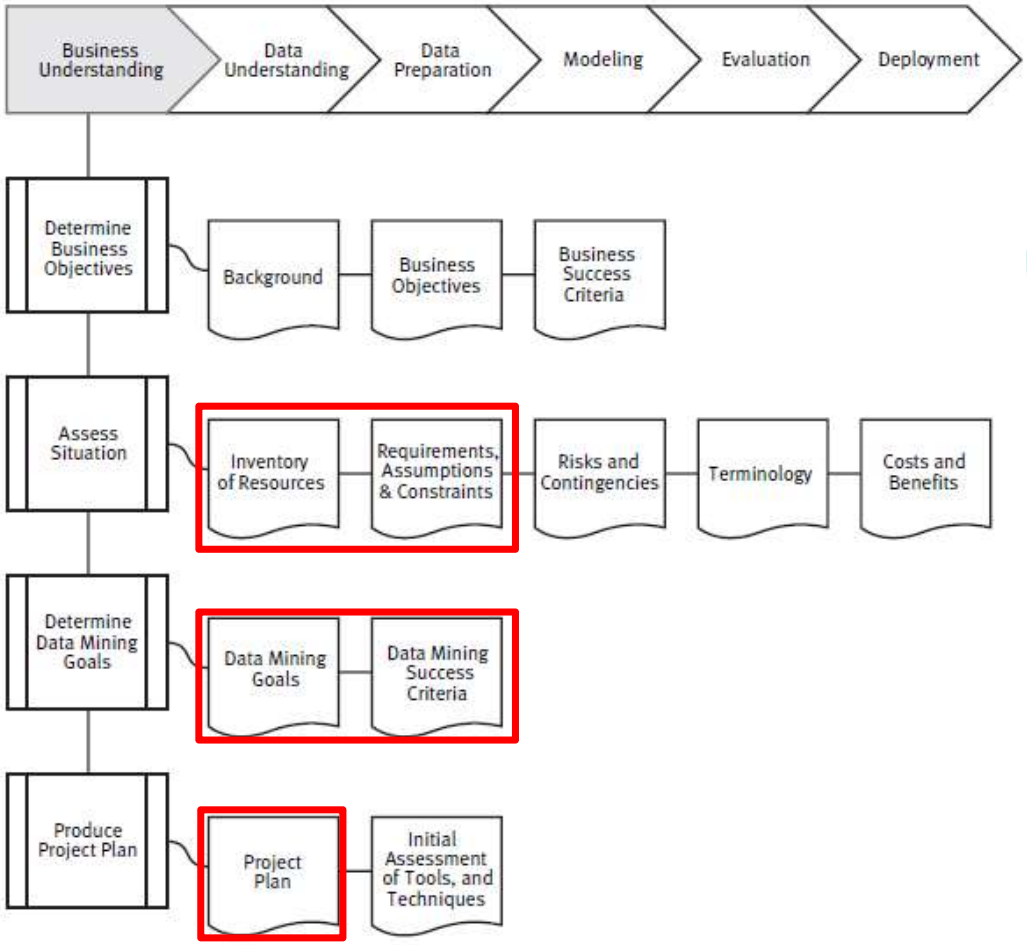
- Randbedingungen getroffen? HW Constraints
- Summarize assessment results

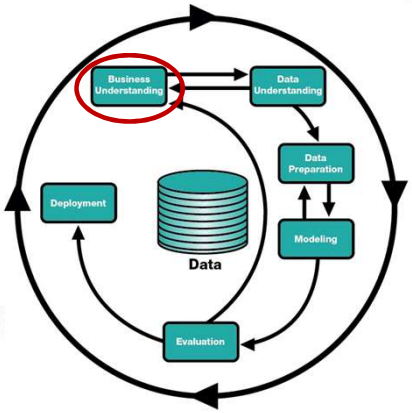
Review of process

- Summarize process and highlight activites

- Festlegung der betriebswirtschaftlichen Ziele
- Situationsbewertung
- Bestimmung der Data Mining Ziele
- Erstellung eines Projektplanes

CRISP-DM im Detail: Business Understanding





- Festlegung der betriebswirtschaftlichen Ziele
- Situationsbewertung
- Bestimmung der Data Mining Ziele
- Erstellung eines Projektplanes

Business Understanding

Determine Business Objectives
Background
Business Objectives
Business Success Criteria

Assess Situation
Inventory of Resources
Requirements, Assumptions, and Constraints
Risks and Contingencies
Terminology
Costs and Benefits

Determine Data Mining Goals
Data Mining Goals
Data Mining Success Criteria

Produce Project Plan
Project Plan
Initial Assessment of Tools and Techniques

- Festlegung der Geschäftsziele
  - Hintergrundwissen:
    Informationen über betriebswirtschaftliche Situation um Ressourcen zu identifizieren
    - Im Bereich der Organisation
    - Im Bereich des betroffenen Geschäftsbereich
  - Geschäftsziele:
    - Primärziele des Kunden aus einem betriebswirtschaftlichen Standpunkt Umsetzungsschritte festhalten
  - Betriebswirtschaftliche Erfolgsziele
    jedes Erfolgskriterium muss mit einem geschäftsziel korrelieren
- Hintergrundwissen:
  Informationen über betriebswirtschaftliche Situation um Ressourcen zu identifizieren
  - Im Bereich der Organisation
  - Im Bereich des betroffenen Geschäftsbereich
- Geschäftsziele:
  - Primärziele des Kunden aus einem betriebswirtschaftlichen Standpunkt Umsetzungsschritte festhalten
- Betriebswirtschaftliche Erfolgsziele
  jedes Erfolgskriterium muss mit einem geschäftsziel korrelieren

Business Understanding

Situationsbewertung

Business Understanding

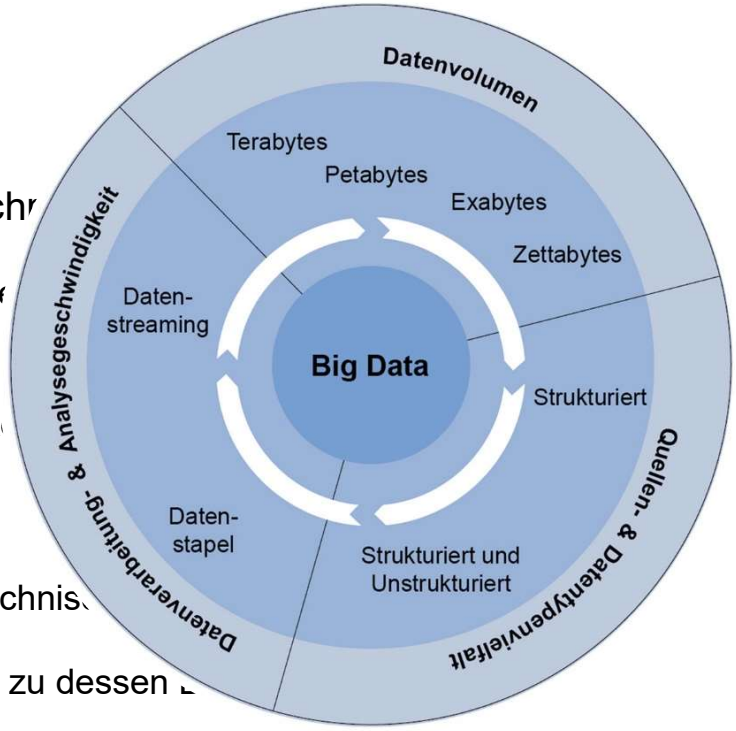
Determine Business Objectives
Background
Business Objectives
Business Success Criteria

Assess Situation
Inventory of Resources
Requirements, Assumptions, and Constraints
Risks and Contingencies
Terminology
Costs and Benefits

Determine Data Mining Goals
Data Mining Goals
Data Mining Success Criteria

Produce Project Plan
Project Plan
Initial Assessment of Tools and Techniques

- Ressourcen-Inventar: Faktensuche zu allen Ressourcen, Beschränkungen und Annahmen
  - Personalverfügung
  - Daten- und Wissensquellen
  - Hardware Ressourcen bestimmen
    verfügbare und benötigte Hardware
  - Software Ressourcen bestimmen
- Anforderungen, Annahmen und Randbedingungen/Einschr
  - Anforderungen festhalten
    Vollständigkeit, Exaktheit, Anwendbarkeit, Wartungsfähigk
  - Annahmen fixieren
  - Einschränkungen identifizieren
    Zugriffsrechte, technische Zugriffsfähigkeit, Zugang zu Dat
    Budget
- Risiken
  - Identifizierung der Risiken
    aus Wirtschaft, organisatorische Risiken, finanzielle und technis
  - Entwicklung von Plänen für möglichen Eintritt von Risiken
    Bedingungen zum Eintritt der Risiken festhalten und Pläne zu dessen
- Kosten und Nutzen
  - Kosten für Datenbeschaffung, Entwicklung und Implementierung aufstellen



Business Understanding

Bestimmung der Data Mining Ziele



Während betriebswirtschaftlicher Erfolg im Sinne der wirtschaftlichen Terminologie Ziele beschreibt, definiert ein Data Mining Erfolg ein Ziel im technischen Sinne

- Data Mining Ziele
  „Übersetzung“ der wirtschaftlichen Ziele in Data Mining Ziele
  Data Mining Ziel spezifizieren durch z.B. Klassifikation, Vorhersage, Beschreibung oder Ähnliches
- Data Mining Erfolgskriterien
  Definition der Kriterien für einen erfolgreichen Projektausgang im technischen Sinn wie z.B. Genauigkeit von Vorhersagen
  Kriterien zur Modellbewertung

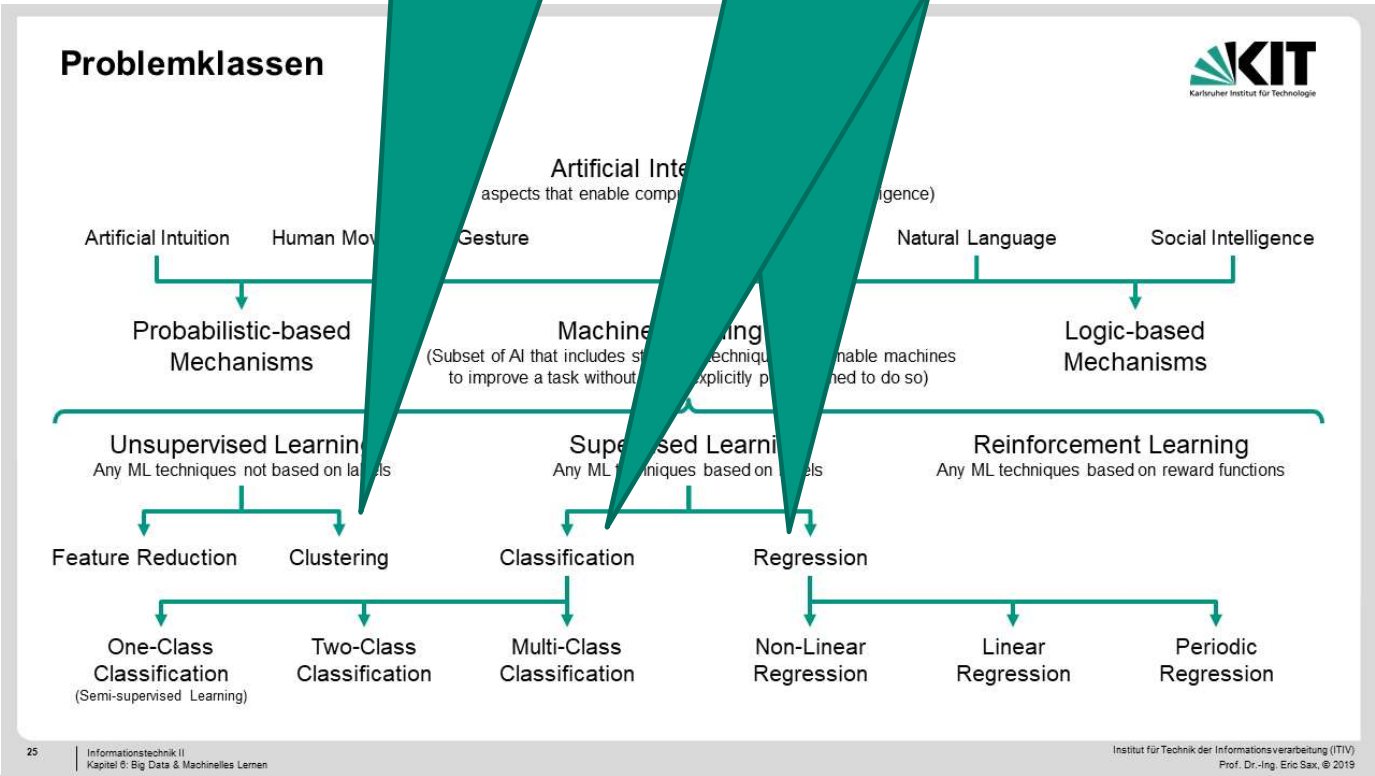
Business Understanding

Bestimmung der Data Mining Ziele

- Randbedingung und Zielsetzung geben den folgenden Verlauf des Prozesses vor
- Später auszuwählender Data Mining Algorithmus hängt von vorhandenen Daten ab
  - Welche Informationen enthalten sie bereits?
  - Welche Struktur weisen die Daten vor?
  - Welche Informationen kann man aus ihnen gewinnen?

Trainieren ohne Vorwissen über die Daten
Suche nach Struktur in den Daten, unbekannte Ziele

Trainieren mit Vorwissen über die Daten
Vorhersage von Werten, bekannte Ziele

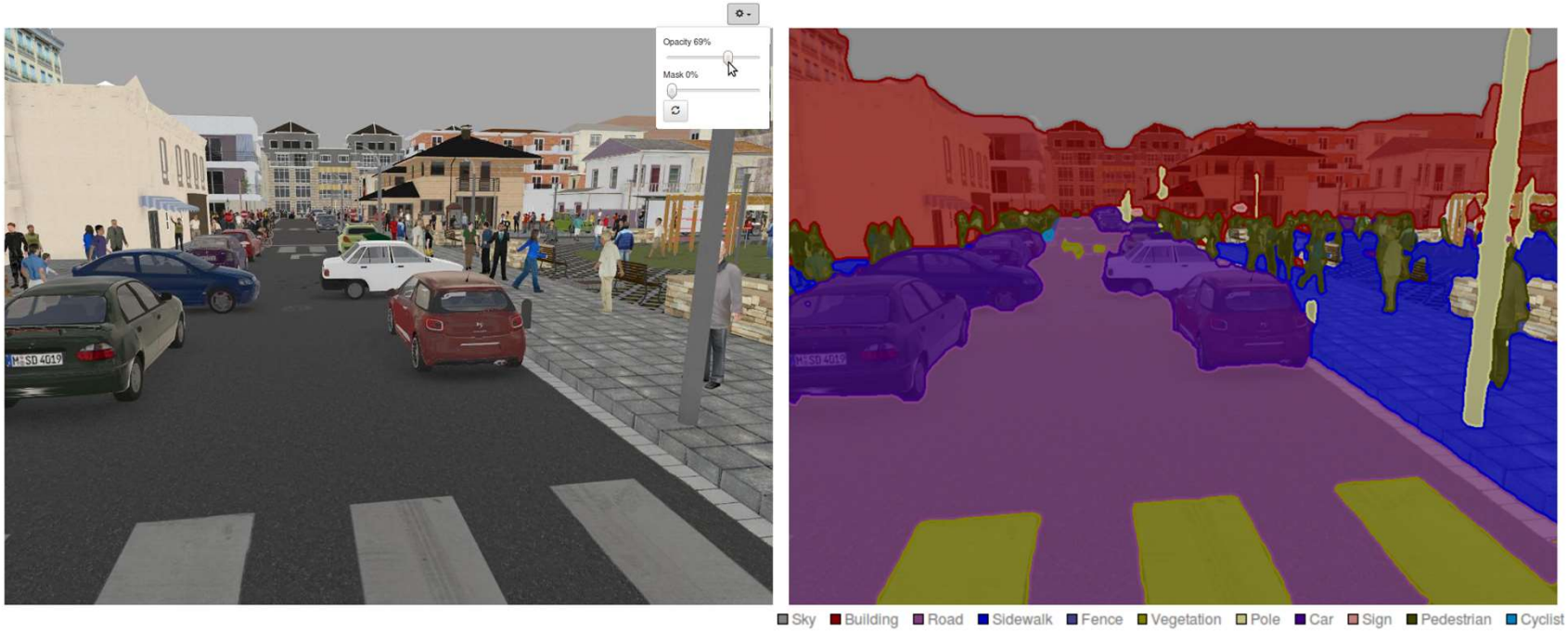


Regression
Business Understanding



- **Eine typische Aufgabe ist es, einen numerischen Zielwert, wie z.B. den Preis eines gebrauchten Autos oder Aktie vorherzusagen.**
- **Dabei werden Merkmale (Kilometerstand, Alter, Marke, etc.) als Prädiktoren bezeichnet.**

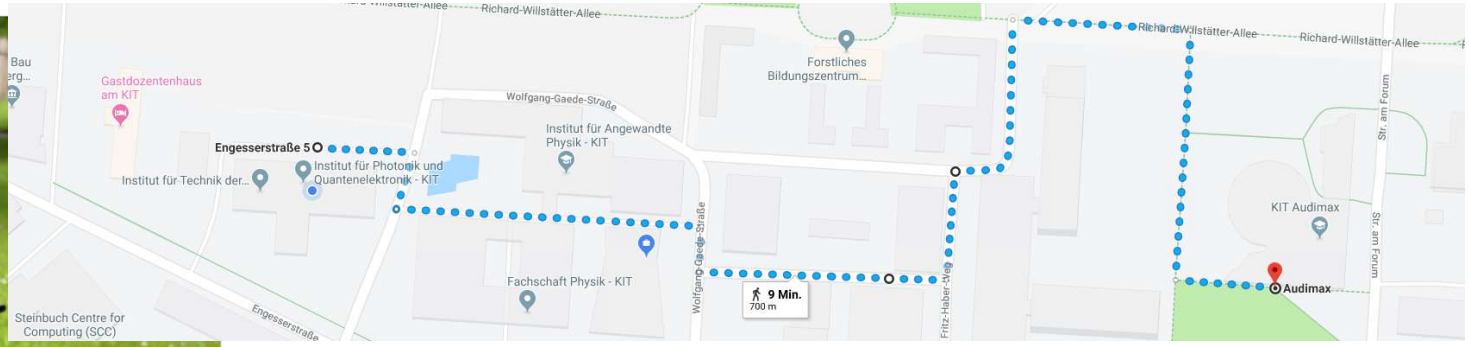
Klassifikation
Business Understanding



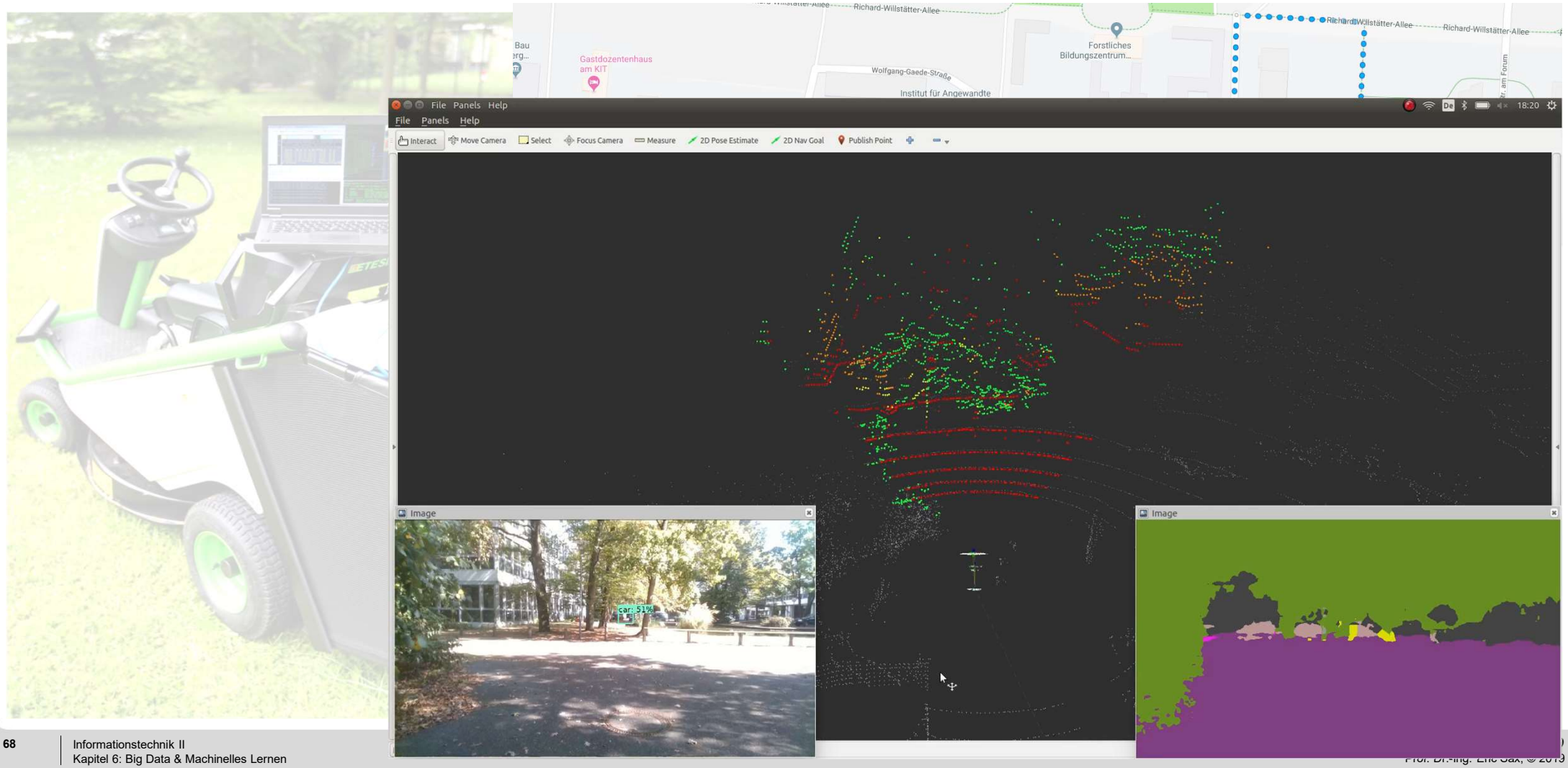
- **Einteilung von Objekten in bekannte(!) Klassen wie z.B. Himmel, Auto oder Vegetation**

Aufzeichnung einer Fahrt am KIT - Kartenerstellung





Aufzeichnung einer Fahrt am KIT - Kartenerstellung



Aufzeichnung einer Fahrt am KIT - Kartenerstellung





Clustering
Business Understanding



- **Zusammenführen von Objekten in eine homogene Gruppe. z.B. Shopping Verhalten von Amazon Kunden**

- Business Understanding
- Fragestellungen
- Fragestellung/Kategorien des ML



Veranstaltungsplan Informationstechnik 2 SS2019

Vorlesung/Übung:	Mi, 8:00 - 9:30 Uhr	Gebäude 30.22, Gaede-Hörsaal
Vorlesung/Übung:	Fr, 9:45 - 11:15 Uhr	Gebäude 30.22, Gaede-Hörsaal

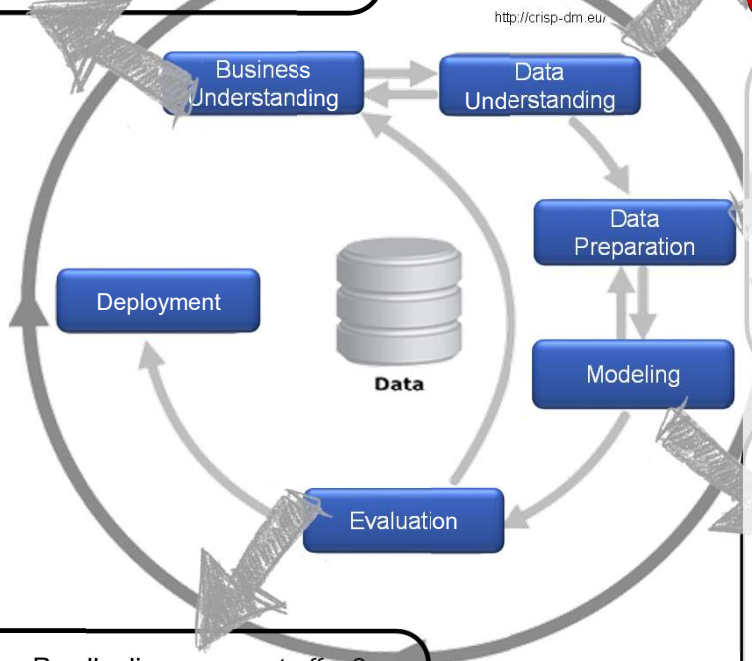
Woche	Datum Mi.	Vorlesung	Datum Fr.	Übung	Inhalt Mi.	Inhalt Fr.
1.	24. Apr		26. Apr	VL		VL0: Einleitung und Motivation
2.	01. Mai	frei	03. Mai	VL	Tag der Arbeit	VL1: Algorithmische Grundlagen
3.	08. Mai	ÜB	10. Mai	VL	ÜB1: Organisatorisches; Besprechung Aufg. 1;	VL2: Sortieralgorithmen
4.	15. Mai		17. Mai	VL		VL3: Algorithmen auf Graphen
5.	22. Mai	VL	24. Mai	frei	VL4: Optimierungsalgorithmen	frei
6.	29. Mai	ÜB	31. Mai	VL	ÜB2: Besprechung Aufg. 2;	VL5: Big Data Einführung
7.	05. Jun	ÜB	07. Jun	VL	ÜB3: Besprechung Aufg. 3;	VL6: Big Data - Prozesse
8.	12. Jun	frei	14. Jun	frei	Pfingsten	Pfingsten
9.	19. Jun	ÜB	21. Jun	VL	ÜB4: Besprechung Aufg. 4;	VL7: Big Data - Modelling 1
10.	26. Jun	ÜB	28. Jun	VL	ÜB5: Besprechung Aufg. 5;	VL8: Big Data - Modellin 2
11.	03. Jul		05. Jul	VL		VL9: Big Data - Evaluation
12.	10. Jul	ÜB	12. Jul	VL	ÜB6: Besprechung Aufg. 6;	VL10: Big Data - Infrastruktur
13.	17. Jul	ÜB	19. Jul	VL	ÜB7: Besprechung Aufg. 7;	VL11: Anwendungen Institut
14.	24. Jul	ÜB	26. Jul	VL	ÜB8: Besprechung Aufg. 8;	VL12: Cyber Security

CRISP-DM im Detail

- Zielsetzung
- Rand-bedingungen

Analytic Aproach

- Regression
- Klassifikation
- Clustering
- Recommendation
- Outlier



Daten
- Struktur
- Statistik
- Visualisierung

Collect Initial Data

- Feature
- Type
- Struktur

Describe & Explore Data

- Statistik: Mean,Var, Median
- Visualisierung: Plots

- Datenformat, Menge, Anzahl Merkmale, Dauer
- Statistische Eigenschaften, Korrelationen, Histogramme, Scatterplot, Box-Plot
- Fehlende Werte? Starkes Rauschen?

Ergebnisse

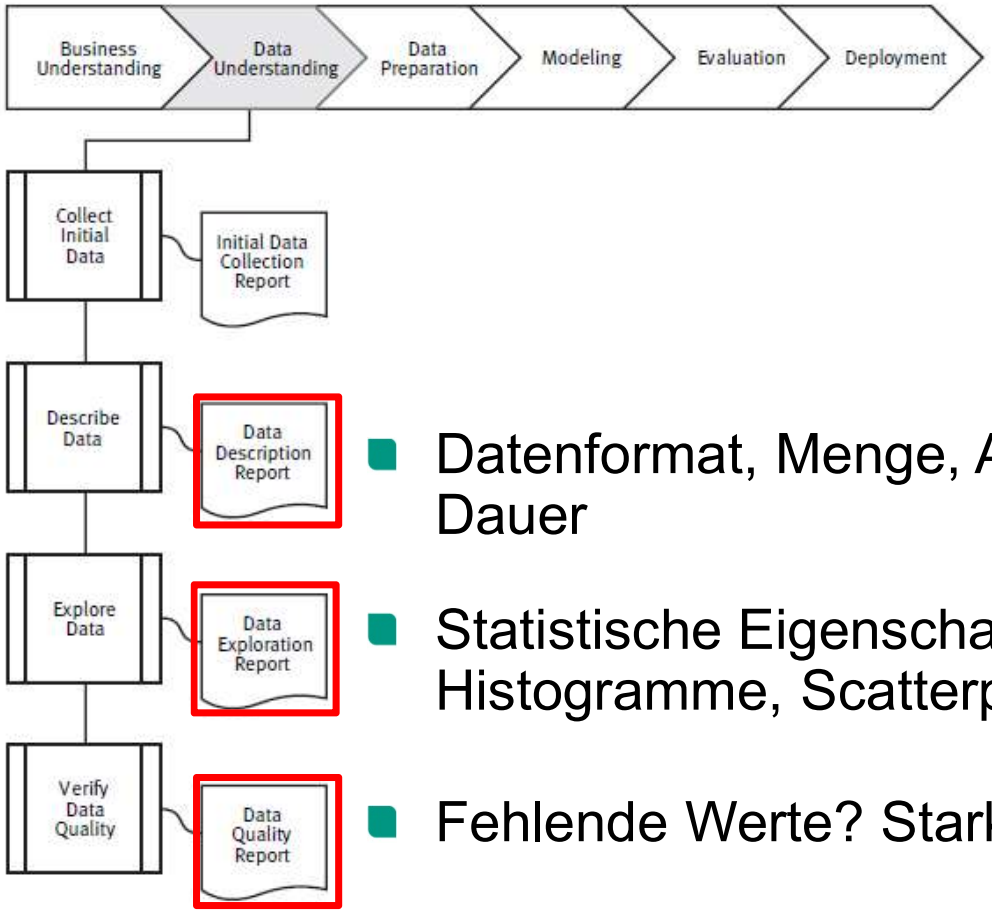
Evaluate Results

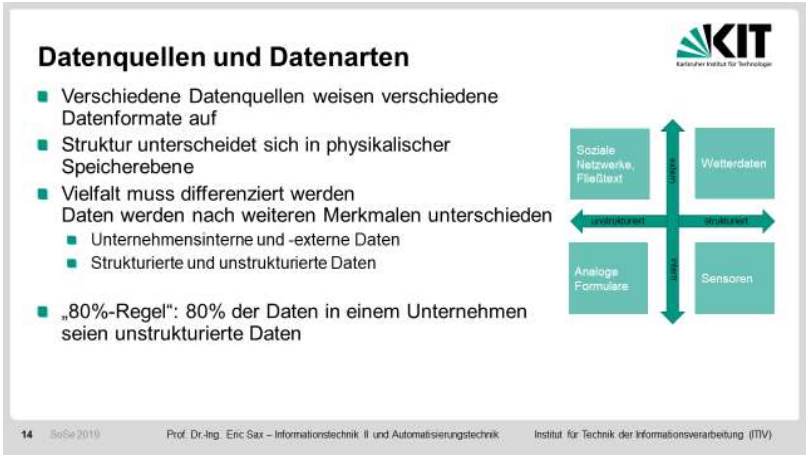
- Randbedingungen getroffen? HW Constraints
- Summarize assessment results

Review of process

- Summarize process and highlight activites

CRISP-DM im Detail: Data Understanding





- Datenformat, Menge, Anzahl Merkmale, Dauer
- Statistische Eigenschaften, Korrelationen, Histogramme, Scatterplot, Box-Plot
- Fehlende Werte? Starkes Rauschen?

Business Understanding	Data Understanding
Determine Business Objectives *Background* *Business Objectives* *Business Success Criteria*	**Collect Initial Data** *Initial Data Collection Report*
Assess Situation *Inventory of Resources* *Requirements, Assumptions, and Constraints* *Risks and Contingencies* *Terminology* *Costs and Benefits*	**Describe Data** *Data Description Report*
Determine Data Mining Goals *Data Mining Goals* *Data Mining Success Criteria*	**Explore Data** *Data Exploration Report*
Produce Project Plan *Project Plan* *Initial Assessment of Tools and Techniques*	**Verify Data Quality** *Data Quality Report*

- Festlegung der Geschäftsziele

CRISP-DM im Detail: Data Understanding

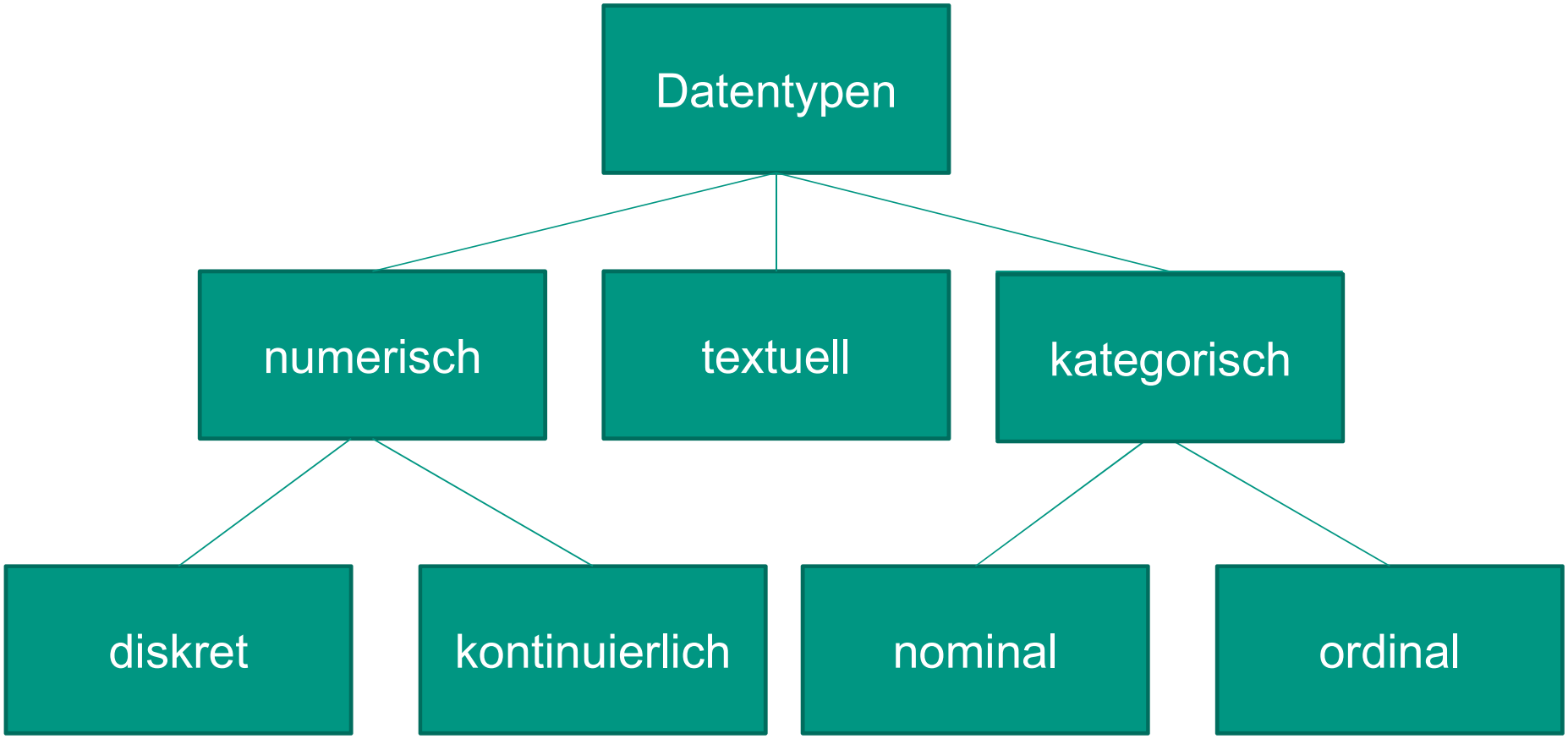
Merkmale (Features)

- **Merkmale sind die Beobachtungen, die verwendet werden um Vorhersagen zu treffen.**
  - Für die Bild-Klassifikationen sind die Pixel die relevanten Merkmale
  - Für Sprache-Erkennung, der Pitch (Tonhöhe) und die Lautstärke
  - Für autonome Fahrzeuge die Kameradaten, der Radar oder Lidar und die GPS-Daten
- **Das Extrahieren der relevanten Merkmale ist entscheidend für das Bilden eines Modells**
  - Zeit kann zum Beispiel irrelevant bei der Klassifikation von Bildern sein.
  - Zeit kann relevant bei der Erklärung von Wetterphenomänen oder -vorhersagen sein
- **Merkmalsarten bei Robotik**
  - Pixels (RGB data)
  - Depth data (sonar, laser rangefinders)
  - Movement (encoder values)
  - Orientation or Acceleration (Gyroscope, Accelerometer, Compass)

Folgende Folien neu nach Email Stang vom 17.6.2019

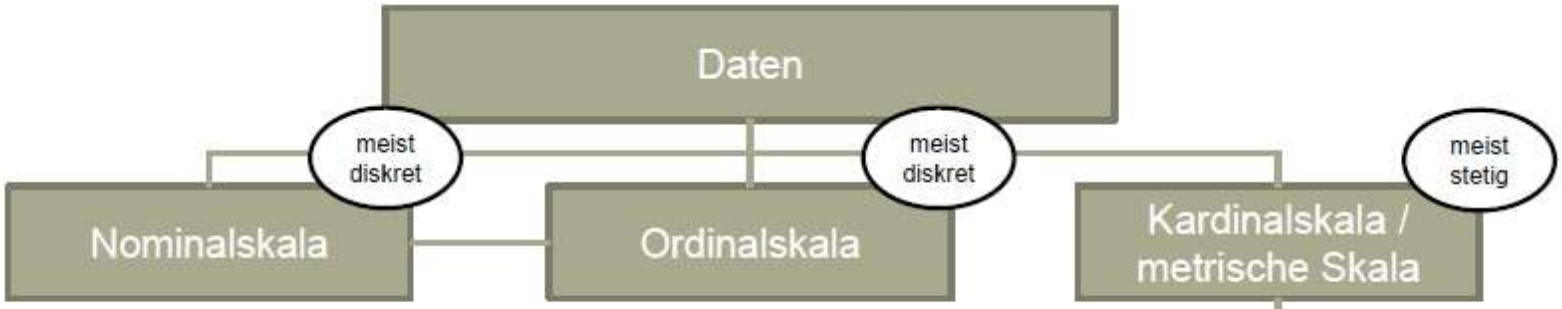
CRISP-DM im Detail: Data Understanding

Datentypen



CRISP-DM im Detail: Data Understanding

Datentypen



- keine Rangordnung

Beispiele

- Geschlecht
- Studiengang
- Familienstand
- Telefonnummer

- Rangordnung
- Keine interpretierbaren Abstände

Beispiele

- Schulnoten
- Steuerklassen
- Präferenzrankings

Intervallskala (kein natürlicher Nullpunkt)

Verhältnisskala (natürlicher Nullpunkt)

- Rangordnung
- Interpretierbare Abstände

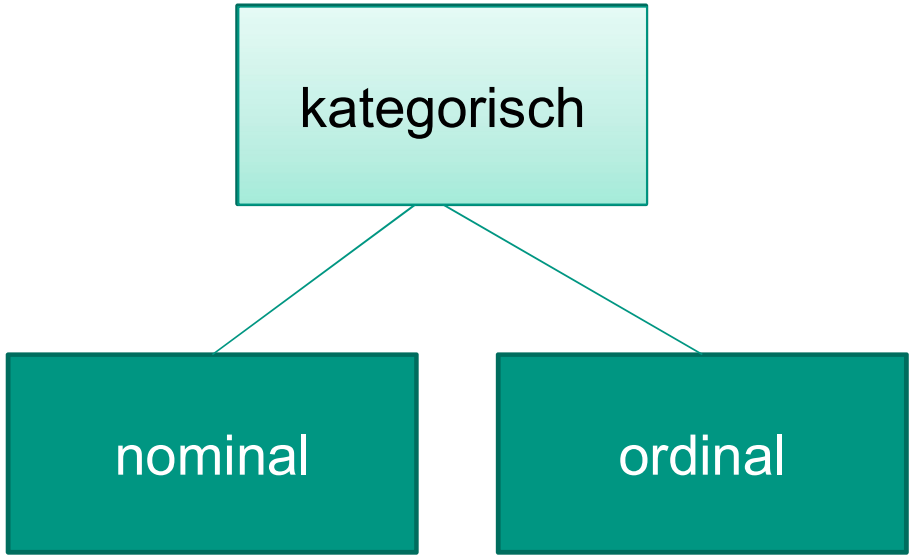
Beispiele

- Preis in EUR
- Abstand in cm

Kategorische Daten (1)

Datentypen

- **Kategorische Daten stellen Unterscheidungs-Merkmale dar.**
  - Beispiel 1: Geschlecht, Sprache usw. für eine Person
  - Beispiel 2: 1 für Frauen und 0 für Männer
    - ► Kategorische Daten können auch numerische Werte annehmen.
    - ► Diese Zahlen haben keine mathematische Bedeutung.



Nominalskalenniveau (1)

Datentypen

- **Bei nominalskalierten Daten handelt es sich um Daten, die in keinerlei natürliche Reihenfolge gebracht werden können**
- **Feststellbar ist hier lediglich, ob zwei statistische Einheiten im Hinblick auf ein nominalskaliertes Merkmal die gleichen Ausprägungen aufweisen.**
- **d.h. beide befragten Personen sind blond oder besitzen unterschiedliche Haarfarben**
- **Da es sich beim Nominalskalennivau um dasjenige Skalenniveau mit dem geringsten Informationsgehalt handelt, lassen sich mit nominalskalierten Daten nur wenige Berechnungen anstellen.**
- **Beispiele: Geschlecht, Kontonummer, Haarfarbe, Telefonnummer, Geschmacksrichtung…**

Ordinalskalenniveau (2)

Datentypen

- **ordinalskalierte Daten können in eine natürliche Reihenfolge gebracht werden**
  - Aber: die Abstände zwischen den einzelnen Werten sind nicht quantifizierbar und somit kann mit ordinalskalierte Daten **nicht** normal gerechnet werden (Additionen, Subtraktionen)
- **Klassisches Beispiel:**
  - Prüfungsnoten besitzen eine natürliche Reihenfolge (eine 1 ist besser als eine 2, eine 2 ist besser als eine 3 usw.) als auch unterschiedliche Abstände zwischen den einzelnen Werten auf.
  - der Notenbereich der 1 umfasst den Bereich von 92% bis 100% der maximal erreichbaren Punkte
  - der Notenbereich der 5 dagegen den Bereich von 0% bis 50%
  - Aus diesem Grund sind Rechenoperationen wie etwa das Addieren oder das Subtrahieren von Noten nicht sinnvoll: Zwei “2er” ergeben keinen “4er” – und wenn man von einem “2er” einen “1er” abzieht, erhält man auch keinen “3er”.
- **Beispiele: Prüfungsnoten, Präferenzrangfolgen, Zufriedenheit (z.B. auf einer Skala von 1 bis 5)**

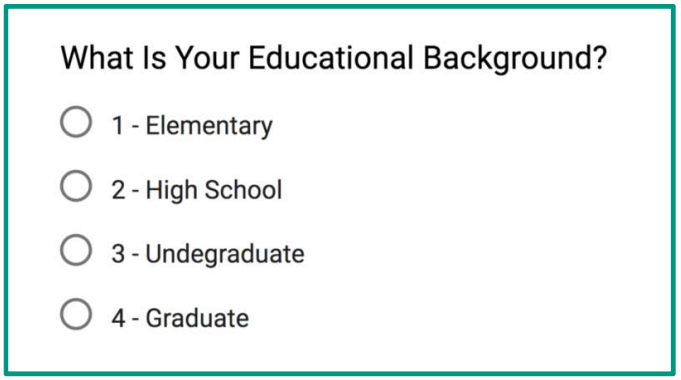
Kategorische Daten (2)

Datentypen

- Nominal
  - stellen diskrete Einheiten dar und dienen zur Kennzeichnung von Variablen, die keinen quantitativen Wert haben
  - Label
  - Ungeordnet, keine Reihenfolge

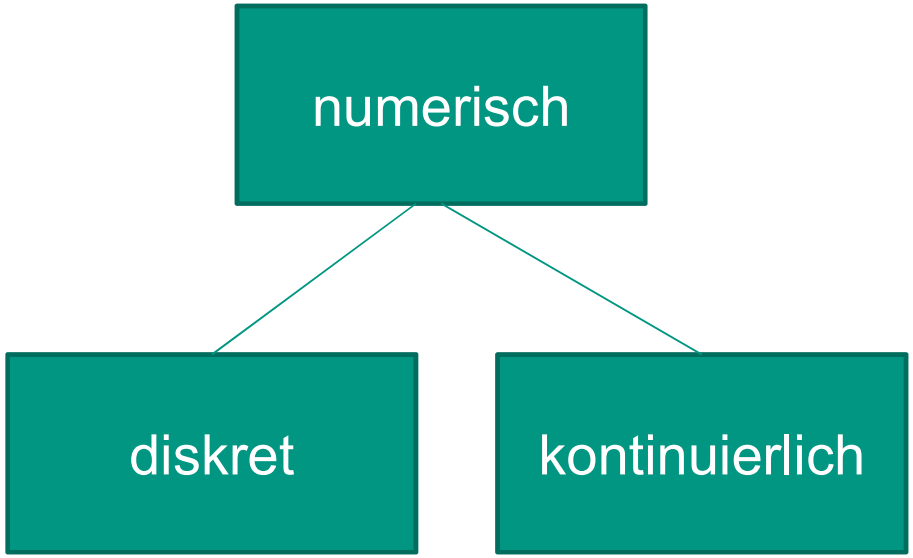


- Ordinal
  - stellen diskrete und geordnete Einheiten dar
  - wird normalerweise verwendet, um nicht-numerische Merkmale wie Glück, Kundenzufriedenheit zu messen.



Numerische Daten
Datentypen

- Daten haben eine messbare Bedeutung



- Diskrete Daten
  - Diese Daten können nur bestimmte numerische Werte haben.
  - Sie entstehen nahezu ausnahmslos durch Zählungen, Beispiele:
    - Anzahl Kinder
    - Anzahl der Besuche beim Hausarzt im Jahr
    - usw.
- Kontinuierliche Daten
  - Diese werden gesammelt durch Messungen., Beispiele:
    - Körpergroße
    - Gewicht
    - Blutdruck
  - Diese Daten können alle möglichen Werte annehmen und sind nur eingeschränkt durch die Genauigkeit der Messung.

Metrisches Skalenniveau (Kardinalskala) (3)

Datentypen

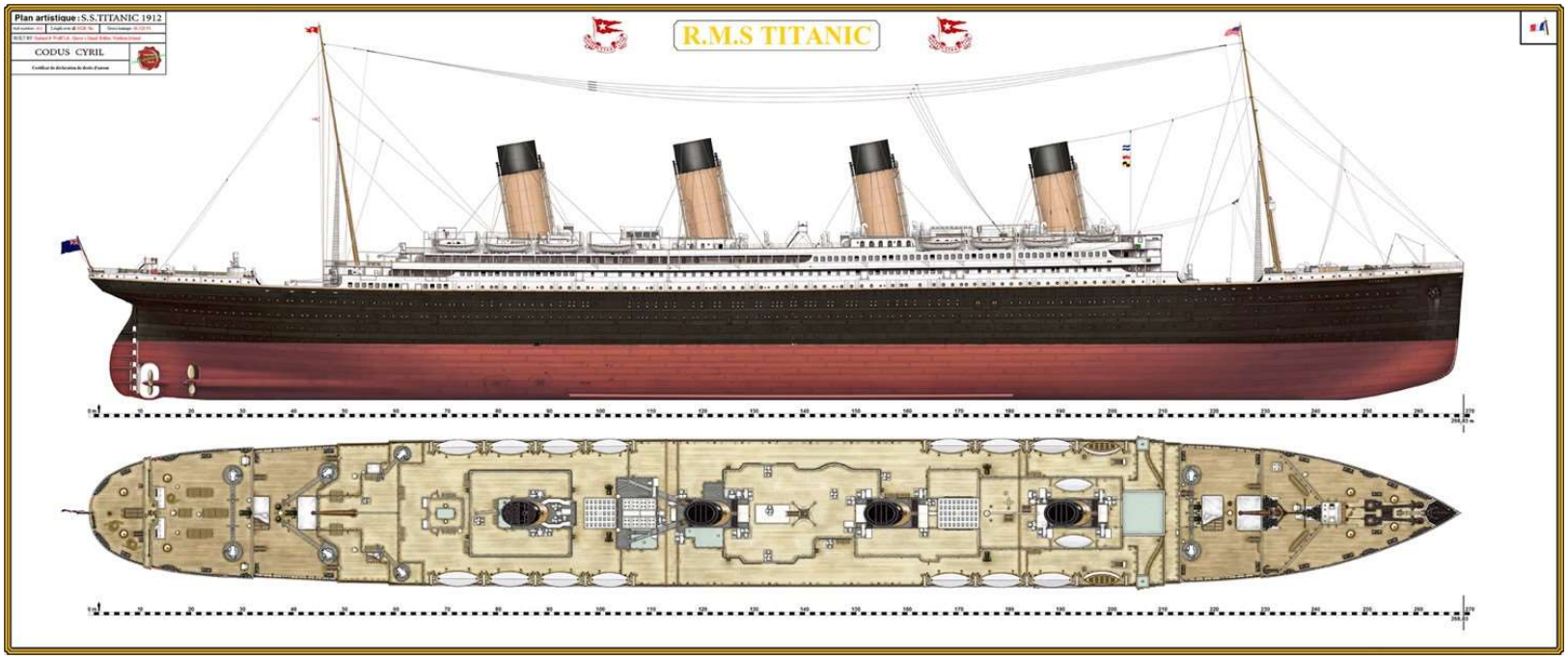
- **Metrisch skalierte Daten verfügen über eine natürliche Reihenfolge sowie auch über quantifizierbare Abstände**
- **Mit metrisch skalierte Daten kann “normal” gerechnet werden.**
- **Weitere Unterscheidung der metrischen Skala**
  - Intervallskala (ohne natürlichen Nullpunkt)
    - – z.B. Temperatur in Celsius
  - Verhältnisskala (mit natürlichem Nullpunkt)
    - – z.B. Temperatur in Kelvin
- **Beispiele: Zeitdauer in Sekunden, Wassertiefe in cm, Preis in Euro und Cent, Streckenlänge in mm…**



Merkmale (Features)

Beispiel: Titanic-Datensatz

- **1 + 11 Features pro Person**
  1. ID
  2. Survival
  3. Pclass
  4. Name
  5. Sex
  6. Age
  7. SibSp
  8. Parch
  9. Ticket
  10. Fare
  11. Cabin
  12. Embarked



Merkmale (Features)

Beispiel: Titanic-Datensatz

- **11 Features pro Person**

1. ID

2. **SURVIVED**	Überlebensvariable mit 0=Nein und 1=Ja
3. **PCLASS**	Klasse auf dem Schiff 1= Erste, 2=Zweite, 3= Dritte
4. **NAME**	Nachname, Title, Vorname(n)
5. **SEX**	Geschlecht „male“=männlich, „female“=weiblich
6. **AGE**	Alter in Jahren
7. **SIBSP**	Anzahl der Geschwister bzw. Partner, die mit an Bord waren
8. **PARCH**	Anzahl der Eltern bzw. Kinder, die mit an Bord waren
9. **TICKET**	ID Nummer des Tickets
10. **FARE**	Preis des Tickets
11. **CABIN**	Kabinennummer
12. **EMBARKED**	Einstiegshafen C= Cherbourg, Q= Queenstown, S=Southampton

Merkmale (Features)

Beispiel: Titanic-Datensatz

- **11 Features (Name, Alter, Geschlecht, Ticketklasse, Überlebt,…)**
  1. ID
  2. Survival
  3. Pclass
  4. Name
  5. Sex
  6. Age
  7. SibSp
  8. Parch
  9. Ticket
  10. Fare
  11. Cabin
  12. Embarked

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S

Merkmale (Features)

Beispiel: Titanic-Datensatz

- **11 Features (Name, Alter, Geschlecht, Ticketklasse, Überlebt,…)**
  1. ID
  2. Survival
  3. Pclass
  4. Name
  5. Sex
  6. Age
  7. SibSp
  8. Parch
  9. Ticket
  10. Fare
  11. Cabin
  12. Embarked

Feature, Merkmale, Attribute

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	umings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2.	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male				373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male				330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1			NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1		

Ordinal

Ordinal

Label

Nominal

Nominal

Numerisch Kardinal

Numerisch Kardinal

Fehlende Werte

Titanic Datensatz Datentypen

Kategorisch

- Nominal: Cabin
- Nominal Embarked (Port of Embarkation)

```
C(Cherbourg)
Q(Queenstown)
S(Southampton)
```

- Nominal: Variablen mit nur 2 Kategorien (dichotomous)
  - Sex

```
Female
Male
```

  - Survived

```
0
1
```

- Ordinal (Variablen die sortiert werden können)
  - Pclass (Socio-economic status)

```
1(Upper)
2(Middle)
3(Lower)
```

Titanic Datensatz Datentypen

Numerisch und Text

- **Diskret**
  - Passenger ID
  - SibSp
  - Parch
- **Kontinuierlich**
  - Age
  - Fare
- **Text**
  - Ticket (Ticket Nummer des Passagiers)
  - Name

Hier könnten wir auch streiten ;-)

CRISP-DM im Detail: Data Understanding

Arten von Daten

- STRUCTURED DATA
  - lists
  - n × p tables, arrays
  - hierarchies (e.g., organization chart)
  - networks (e.g., travel routes, hypertext = links)
- Generic data-interchange formats:
  - XML,
  - JSON

- UNSTRUCTURED DATA
  - text
  - images
  - video
  - Sound

- Often can be made structured by, e.g., parsing language, segmenting images, etc.

Strukturierte Daten

- **CSV, comma separated values:**

```
sepal_length,sepal_width,petal_length,petal_width,species
5.1,3.5,1.4,0.2,setosa
4.9,3,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
```

- **hierarchies, e.g., Newick tree format**

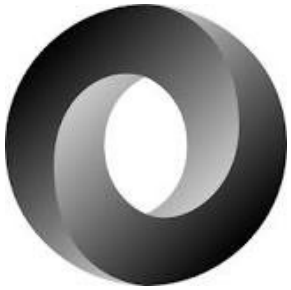
```
(A,B,(C,D)E)F;
```

- **networks, e.g., GraphViz (DOT)**

```
digraph graphname {
    a -> b -> c;
    b -> d;
}
```

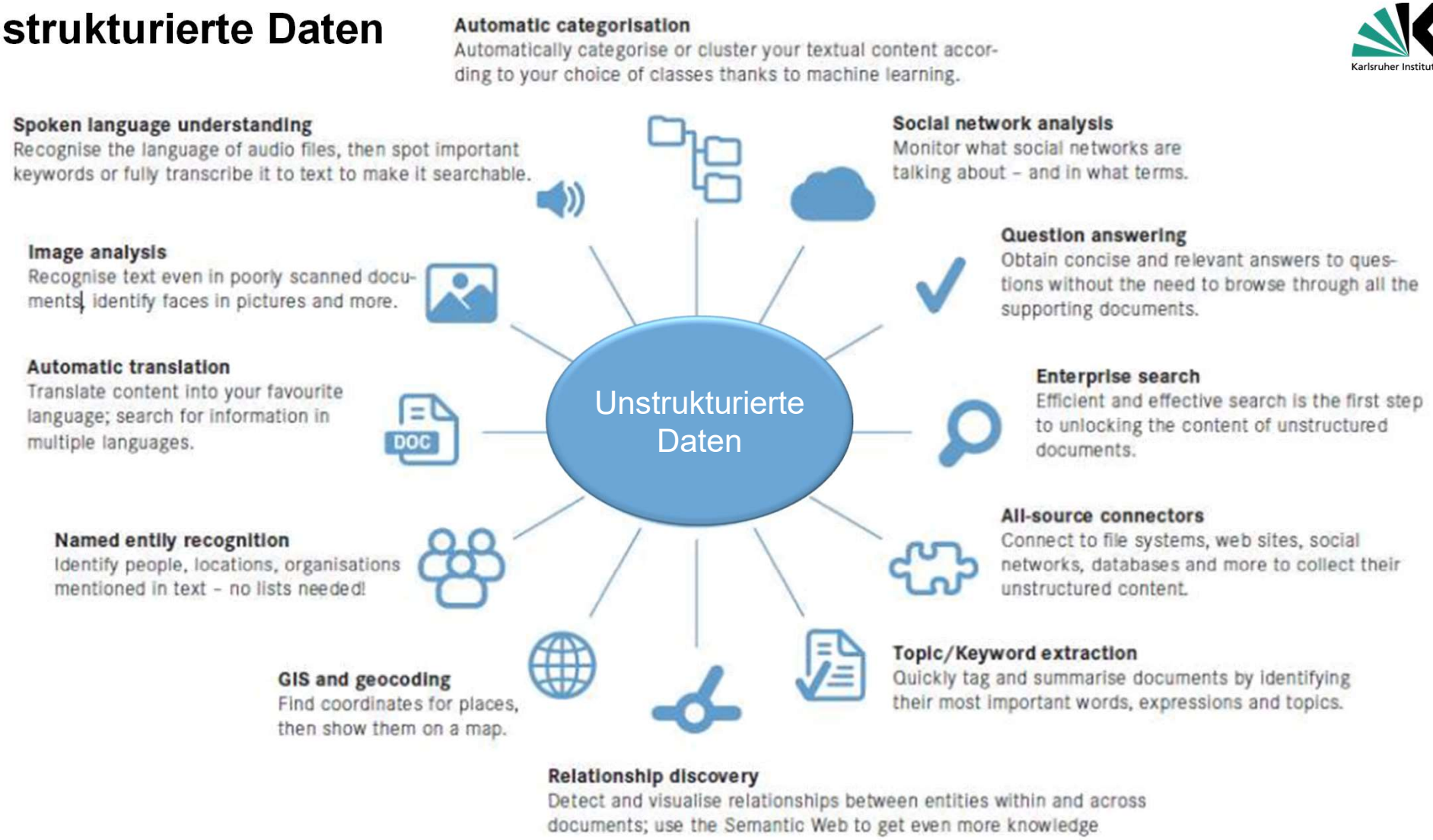
Strukturierte Daten

- **JSON**
  - Die JavaScript Object Notation, kurz **JSON** [ˈdʒeɪsən], ist ein kompaktes Datenformat in einer einfach lesbaren Textform zum Zweck des Datenaustauschs zwischen Anwendungen.



```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

Unstrukturierte Daten



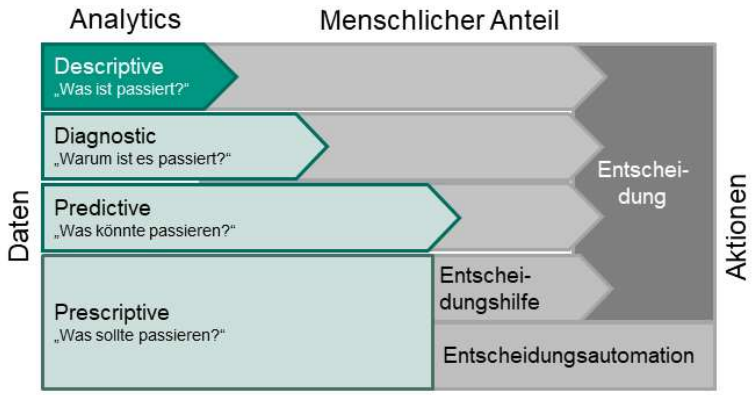
Deskriptive Statistik

- Count.
- Mean.
- Standard Deviation.
- Minimum Value.
- 25th Percentile.
- 50th Percentile (Median)
- 75th Percentile.
- Ascombe Quartet
- Maximum Value

Prozess

Analyseformen auf Daten

- Es geht darum Daten zu analysieren, um daraus Erkenntnisse zu gewinnen.
- Erkenntnisse dienen als Grundlage einer Entscheidung und lösen Handlungen aus.



- Descriptive Analytics
  - Die Blickrichtung ist auf die Vergangenheit gerichtet und beschreibt, was passiert ist.
  - Der Anteil, den ein Mensch noch beitragen muss, um zu einer Entscheidung und einer Handlung zu kommen, ist groß.
  - Beispiel:
    - Eine Bank analysiert die Daten von Kunden, die ihre Kreditraten nicht bezahlt haben.
    - Sie gruppiert sie vielleicht oder ordnet sie geografischen oder sozioökonomischen Faktoren zu.

3 | Informationstechnik II
Kapitel 6: Big Data & Machinelles Lernen

Institut für Technik der Informationsverarbeitung (ITIV)
Prof. Dr.-Ing. Eric Sax, © 2019

- Class Distribution (Classification Only)

Deskriptive Statistik

Mittelwert, Median, Min, Max

x	13	18	13	14	13	16	21	13	10

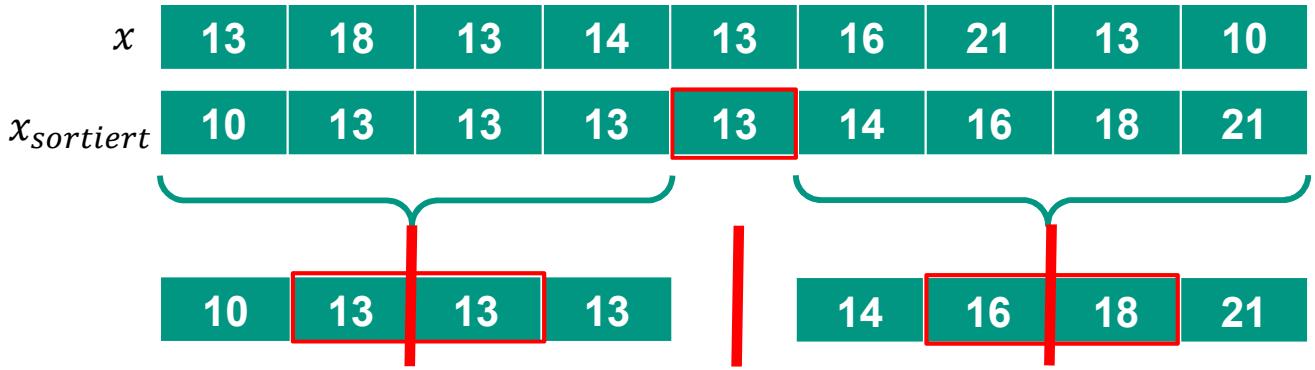
- Mittelwert: $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \ldots + x_n}{n} = 14{,}55$

$x_{sortiert}$	10	13	13	13	13	14	16	18	21

- Median: $\begin{cases} x_{\frac{n+1}{2}} & n \text{ ungerade} \\ \frac{1}{2}\left(x_{\frac{n}{2}} + x_{\frac{n}{2}+1}\right) & n \text{ gerade.} \end{cases}$ = 13

- Min, Max: 10, 21

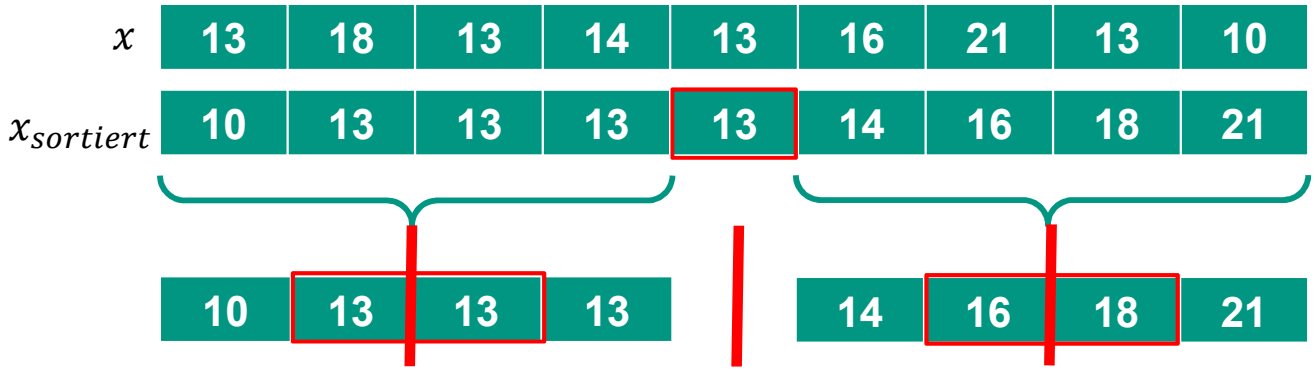
Interquartilsabstand (IQR)



$$untere\ Quartil\ Q_1 = \frac{13+13}{2} = 13 \quad obere\ Quartil\ Q_3 = \frac{16+18}{2} = 17$$

- Ein empirisches (p-)Quantil, auch kurz einfach nur Quantil genannt ist eine Kennzahl einer Stichprobe.
  - Ein empirisches p-Quantil teilt die Stichprobe so, dass ein Anteil der Stichprobe von p kleiner als das empirische p-Quantil ist und ein Anteil von 1-p der Stichprobe größer als das empirische p-Quantil ist.
    - Ist beispielsweise eine Stichprobe von Schuhgrößen gegeben, so ist das empirische 0,35-Quartil diejenige Schuhgröße, bei der 35% der Schuhgrößen in der Stichprobe kleiner als sind und 65 % größer als sind.
  - Dabei heißt das 0,25-Quartil das untere Quartil und das 0,75-Quartil das obere Quartil.
  - Zwischen oberem und unterem Quartil liegt die Hälfte der Stichprobe, unterhalb des unteren Quartils und oberhalb des oberen Quartils jeweils ein Viertel der Stichprobe.
  - Auf Basis der Quartile wird der Interquartilsabstand definiert → ein Streuungsmaß.

Interquartilsabstand (IQR)



$$untere\ Quartil\ Q_1 = \frac{13+13}{2} = 13 \quad obere\ Quartil\ Q_3 = \frac{16+18}{2} = 17 \qquad IQR = Q_3 - Q_1 = 4$$

- IQR gibt an, wie weit das obere und das untere Quartil auseinanderliegen und damit auch, wie breit der Bereich ist, in dem die mittleren 50 % der Stichprobe liegen.

Varianz und Standardabweichung

- **Varianz (μ = Mittelwert)**

x	13	18	13	14	13	16	21	13	10
$(x-\mu)$	-1,55	3,45	-1,55	-0,55	-1,55	1,45	6,45	-1,55	-4,55
$(x-\mu)^2$	2,4025	11,9025	2,4025	0,3025	2,4025	2,1025	41,6025	2,4025	20,7025

$$\sum = 86{,}225 \qquad \sigma^2 = \frac{1}{N}\sum_{i=1}^{n}(x_i - \mu)^2 = 9{,}580$$

► Die Varianz ist eine Maß für die Streuung der Wahrscheinlichkeitsdichte um ihren Schwerpunkt.

Varianz und Standardabweichung

Varianz (μ = Mittelwert)

x	13	18	13	14	13	16	21	13	10
$(x-\mu)$	-1,55	3,45	-1,55	-0,55	-1,55	1,45	6,45	-1,55	-4,55
$(x-\mu)^2$	2,4025	11,9025	2,4025	0,3025	2,4025	2,1025	41,6025	2,4025	20,7025

$$\sum = 86{,}225$$

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{n}(x_i - \mu)^2 = 9{,}580$$

- Die Varianz ist eine Maß für die (quadratische) Streuung der Wahrscheinlichkeitsdichte um ihren Schwerpunkt.

Standardabweichung (Streuung)

- Streuung der Werte einer Zufallsvariablen um ihren Erwartungswert oder einer Stichprobe um den ihr eigenen Mittelwert bezeichnet man als Standardabweichung.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(x_i - \mu)^2} = 3{,}095$$

- Die Standardabweichung ist die Größe ohne Quadrat, die somit vergleichbar mit einer Zufallsvariablen wird.

Kovarianz

- **Die Kovarianz ist ein Maß für den Zusammenhang zweier Zufallsvariablen mit gemeinsamer Wahrscheinlichkeitsverteilung.**
  - Der Wert dieser Kenngröße macht tendenzielle Aussagen darüber, ob hohe Werte der einen Variablen eher mit hohen oder eher mit niedrigen Werten der anderen Variablen einhergehen.
  - Die Kovarianz ist ein Maß für die Assoziation zwischen zwei Zufallsvariablen.

Die Kovarianz zwischen zwei Zufallsvariablen x und y wird wie folgt berechnet:

$$\mathrm{Cov}(x,y) = \frac{\sum_{i=1}^{N} (x_i - \bar{x})\,(y_i - \bar{y})}{N-1}$$

- x und y sind die Zufallsvariablen
- $\bar{x}$ und $\bar{y}$ sind die Mittelwerte der Datenreihen der Zufallsvariablen x und y
- N ist die Größe der Stichprobe (daher die Anzahl an Elementen in der Datenreihe von x oder y)

Korrelation

- **Der Korrelation ist eng verwandt mit der Kovarianz.**
  - Die Kovarianz entspricht der Korrelation, wenn die Kovarianz durch das Produkt der Standardabweichungen beider Zufallsvariablen geteilt wird.

$$\rho_{X,Y} = \operatorname{corr}(X,Y) = \frac{\operatorname{cov}(X,Y)}{\sigma_X \cdot \sigma_Y} = \frac{E\left[(X-\bar{X})\cdot(Y-\bar{Y})\right]}{\sigma_X \cdot \sigma_Y} = \frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\bar{X})^2 \cdot \sum_{i=1}^{n}(Y_i-\bar{Y})^2}}$$

- X und Y sind zwei Messreihen bzw. Zufallsvariablen
- $\operatorname{cov}(X, Y)$ ist die Kovarianz von X und Y
- σ ist die Standardabweichung (da sich $\frac{1}{n}$ und $\frac{1}{n-1}$ aus dem Bruch herauskürzen, macht es keinen Unterschied, ob man die Standardabweichung der Stichprobe oder Grundgesamtheit verwendet)
- $E(x)$ ist der Erwartungswert der Variable x

Chancen mit Datenvisualisierung

Anscombe Quartett → Statistik

I		II		III		IV	
x	y	x	y	x	y	x	y
10	8,04	10	9,14	10	7,46	8	6,58
8	6,95	8	8,14	8	6,77	8	5,76
13	7,58	13	8,74	13	12,74	8	7,71
9	8,81	9	8,77	9	7,11	8	8,84
11	8,33	11	9,26	11	7,81	8	8,47
14	9,96	14	8,1	14	8,84	8	7,04
6	7,24	6	6,13	6	6,08	8	5,25
4	4,26	4	3,1	4	5,39	19	12,5
12	10,84	12	9,13	12	8,15	8	5,56
7	4,82	7	7,26	7	6,42	8	7,91
5	5,68	5	4,74	5	5,73	8	6,89

Chancen mit Datenvisualisierung

Anscombe Quartett → Statistik

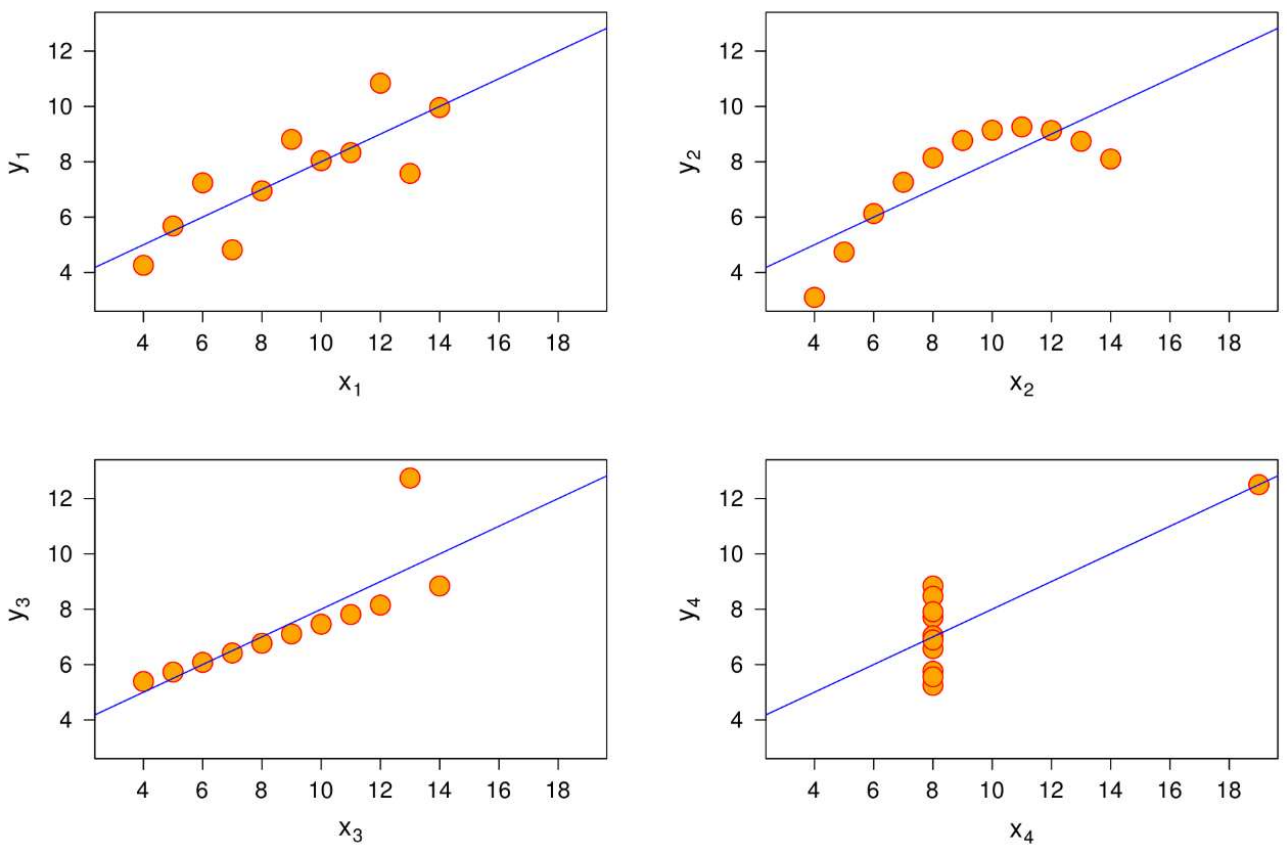
	I		II		III		IV	
	x	y	x	y	x	y	x	y
	10	8,04	10	9,14	10	7,46	8	6,58
	8	6,95	8	8,14	8	6,77	8	5,76
	13	7,58	13	8,74	13	12,74	8	7,71
	9	8,81	9	8,77	9	7,11	8	8,84
	11	8,33	11	9,26	11	7,81	8	8,47
	14	9,96	14	8,1	14	8,84	8	7,04
	6	7,24	6	6,13	6	6,08	8	5,25
	4	4,26	4	3,1	4	5,39	19	12,5
	12	10,84	12	9,13	12	8,15	8	5,56
	7	4,82	7	7,26	7	6,42	8	7,91
	5	5,68	5	4,74	5	5,73	8	6,89
SUM	99,00	82,51	99,00	82,51	99,00	82,50	99,00	82,51
AVG	9,00	7,50	9,00	7,50	9,00	7,50	9,00	7,50
STDEV	3,32	2,03	3,32	2,03	3,32	2,03	3,32	2,03



Anscombe Quartett → Visualisierung

- Datensatz I scheint saubere und gut passende lineare Modelle zu haben.
- Der Datensatz II ist nicht normalverteilt.
- Im Datensatz III ist die Verteilung linear, aber die berechneten Werte werden durch einen Ausreißer verfälscht.
- Datensatz IV zeigt, dass ein Ausreißer ausreicht, um eine hohe Korrelation zu erzeugen.

- Dieses Quartett betont die Bedeutung der Visualisierung in der Datenanalyse.
  - Der Blick auf die Daten zeigt viel von der Struktur und erzeugt ein klares Bild des Datensatzes.



Merkmale (Features)

Beispiel: Titanic-Datensatz

- **11 Features (Name, Alter, Geschlecht, Ticketklasse, Überlebt,…)**
  1. ID
  2. Survival
  3. Pclass
  4. Name
  5. Sex
  6. Age
  7. SibSp
  8. Parch
  9. Ticket
  10. Fare
  11. Cabin
  12. Embarked

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S

Merkmale (Features)

Beispiel: Titanic-Datensatz

- **11 Features (Name, Alter, Geschlecht, Ticketklasse, Überlebt,…)**

1. ID
2. Survival
3. Pclass
4. Name
5. Sex
6. Age
7. SibSp
8. Parch
9. Ticket
10. Fare
11. Cabin
12. Embarked

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
count	888.000000	888.000000	888.000000	711.000000	888.000000	888.000000	888.000000
mean	0.381757	2.313063	0.647523	29.675345	0.524775	0.381757	30.582164
std	0.486091	0.834007	0.478011	14.552495	1.104186	0.806949	41.176366
min	0.000000	1.000000	0.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	0.000000	20.000000	0.000000	0.000000	7.895800
50%	0.000000	3.000000	1.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	1.000000	38.000000	1.000000	0.000000	30.771850
max	1.000000	3.000000	1.000000	80.000000	8.000000	6.000000	263.000000

	Sex	Survived	Pclass	Age	SibSp	Parch	Fare
0	0	0.741214	2.162939	27.888462	0.696486	0.651757	42.985091
1	1	0.186087	2.394783	30.705477	0.431304	0.234783	23.830658

Merkmale (Features)

Beispiel: Titanic-Datensatz

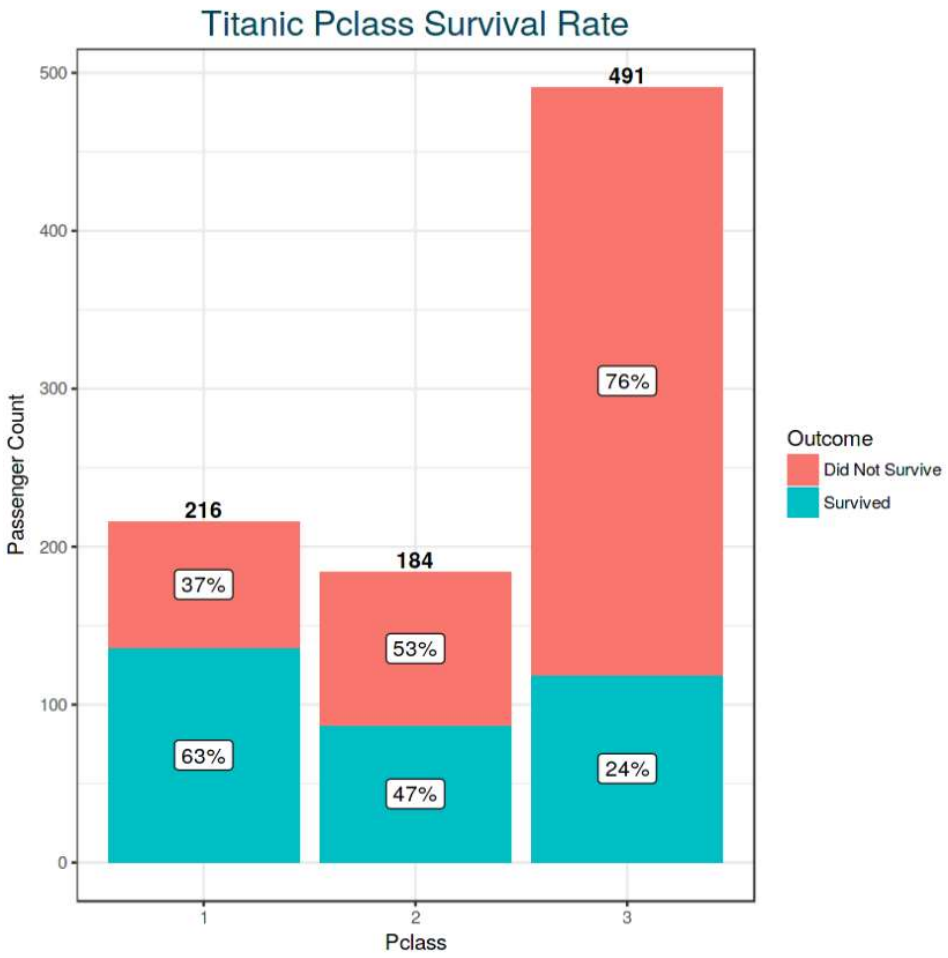
- **11 Features (Name, Alter, Geschlecht, Ticketklasse, Überlebt,…)**

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
Anzahl der Datenpunkte	888.000000	888.000000	888.000000	711.000000	888.000000	888.000000	888.000000
Mittelwert	0.381757	2.313063	0.647523	29.675345	0.524775	0.381757	30.582164
Standardabweichung	0.486091	0.834007	0.478011	14.552495	1.104186	0.806949	41.176366
Min. Wert	0.000000	1.000000	0.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	0.000000	20.000000	0.000000	0.000000	7.895800
50%	0.000000	3.000000	1.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	1.000000	38.000000	1.000000	0.000000	30.771850
Max. Wert	1.000000	3.000000	1.000000	80.000000	8.000000	6.000000	263.000000

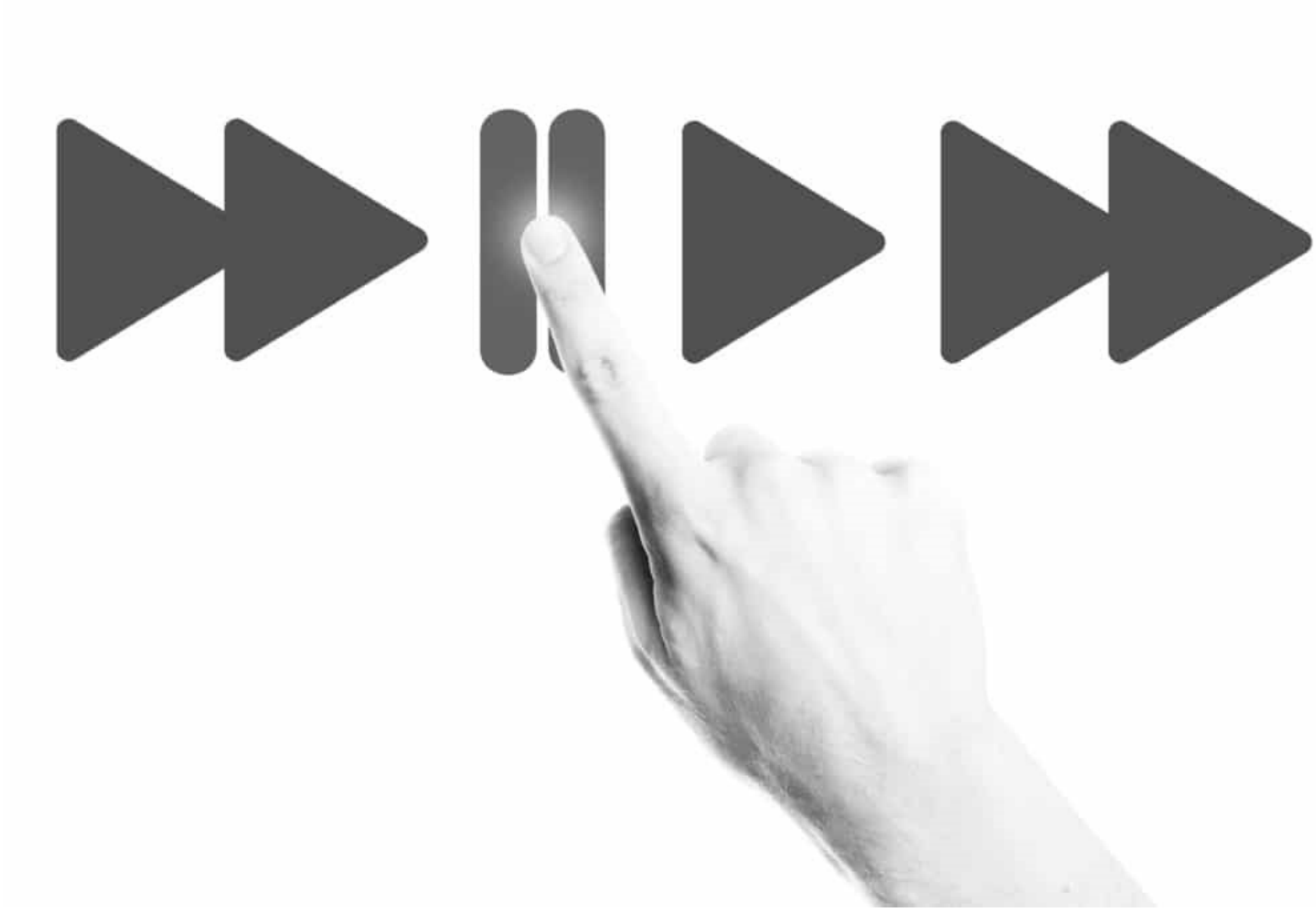
	Sex	Survived	Pclass	Age	SibSp	Parch	Fare
female	0	0.741214	2.162939	27.888462	0.696486	0.651757	42.985091
male	1	0.186087	2.394783	30.705477	0.431304	0.234783	23.830658

Visualisierung des Titanic Datasets

Histogramm

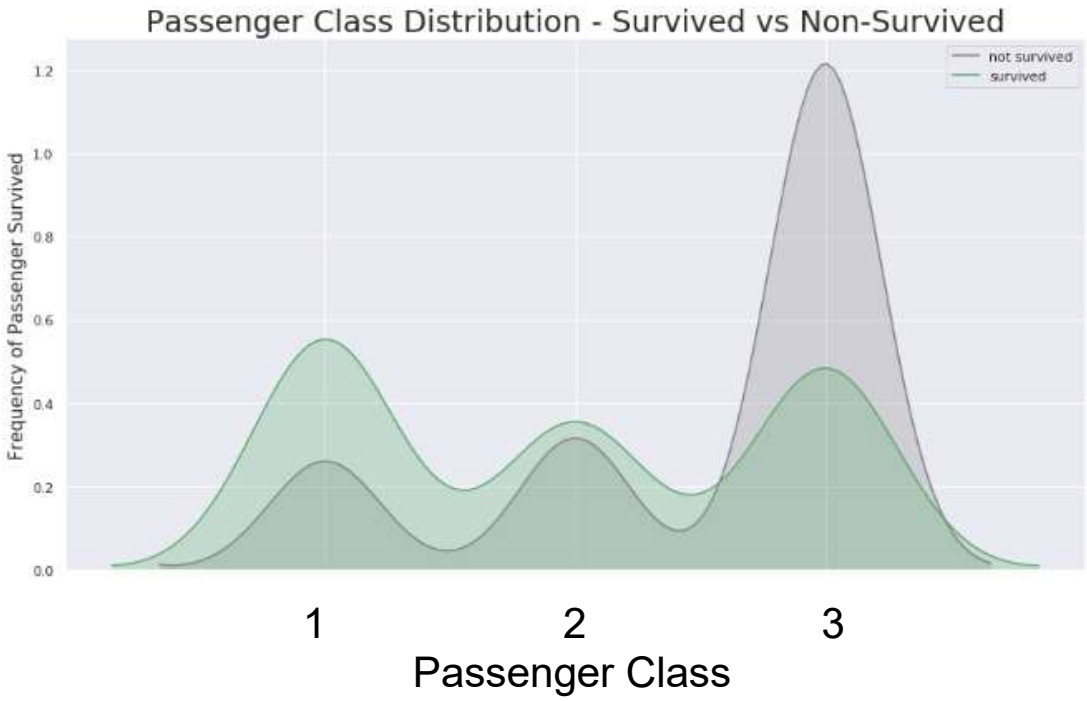


- Histogramme eignen sich hervorragend, um Vergleiche zwischen zwei oder mehr Dingen darzustellen
- Können Veränderungen im Laufe der Zeit darstellen
- Kleine Änderungen können aber nur schwer erkannt werden



Visualisierung des Titanic Datasets

Häufigkeitsverteilung

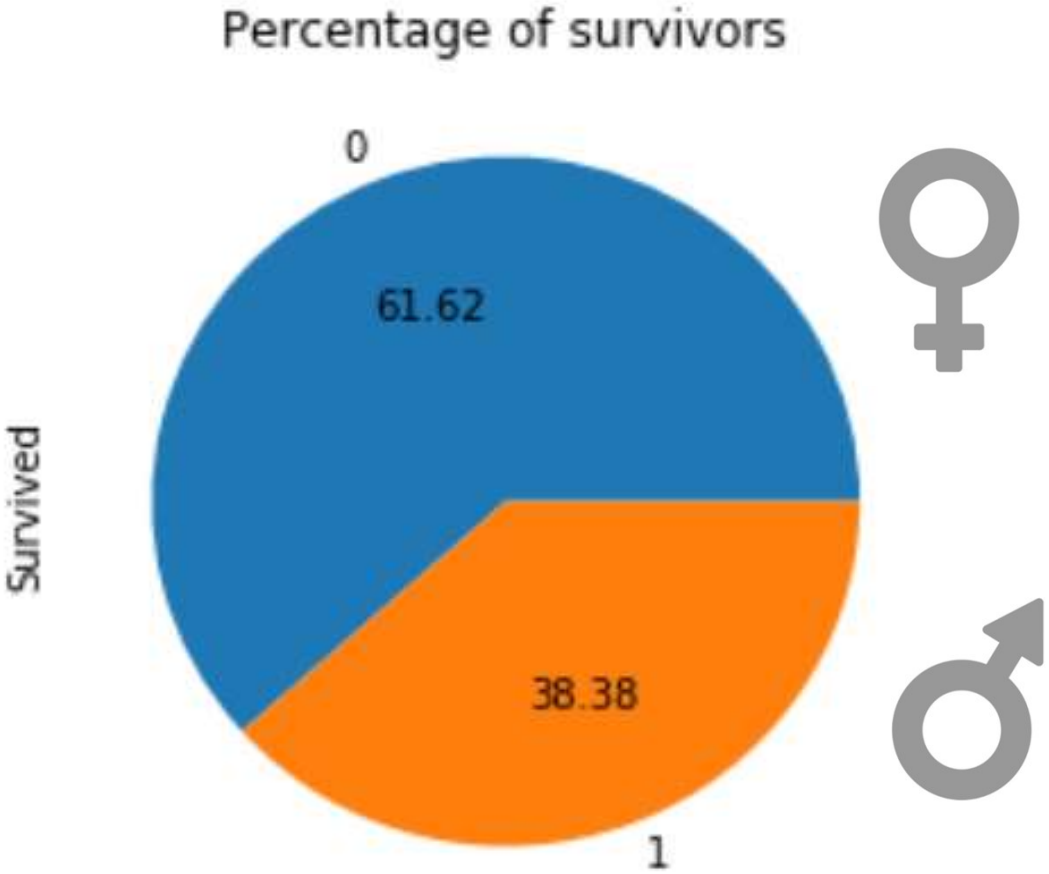


- Häufigkeitsverteilungen lassen sich grafisch darstellen.
- Die Verwendung von Diagrammen hat den Vorteil, dass sich die Form einer Verteilung auf einen einzigen Blick erschließt.
- Besonderheiten fallen bei einer Verteilung auf, die beim bloßen Durchsehen der tabellarischen Notation vielleicht übersehen worden wären.

Not survived
Survived

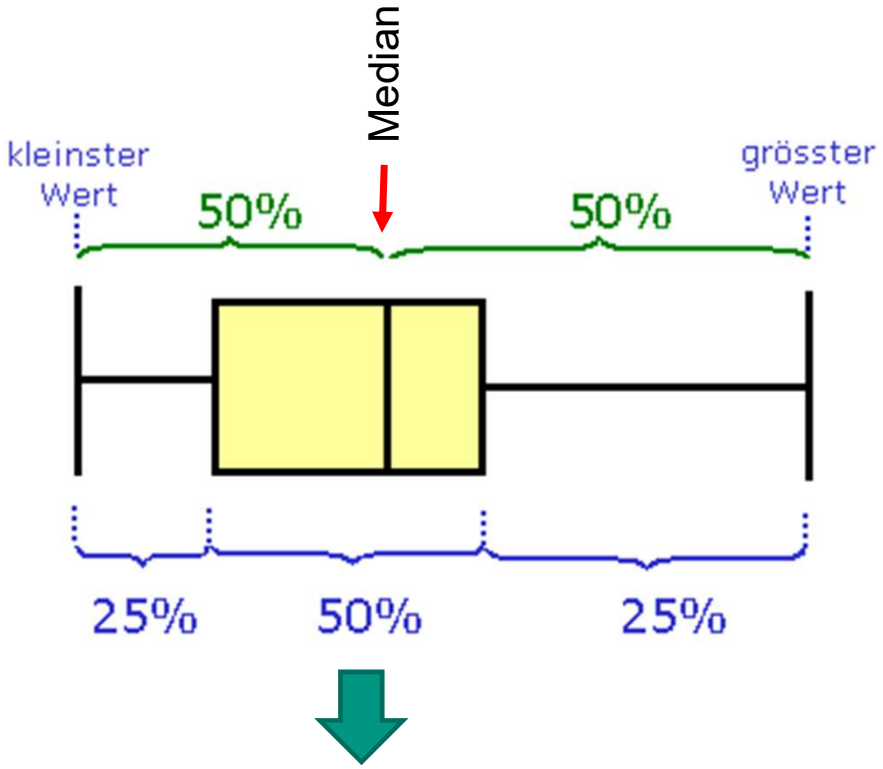
Visualisierung des Titanic Datasets

Kuchendiagramm



Visualisierung des Titanic Datasets

Boxplots

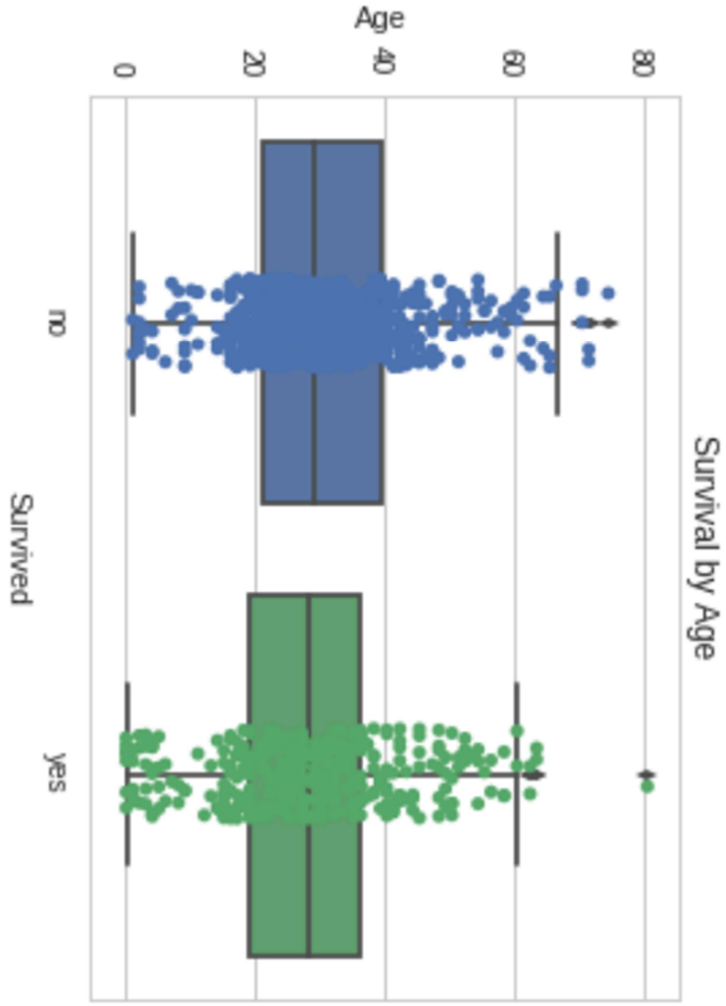


- Auch *Box-Whisker Plot* (deutsch: *Kastengrafik*)
- Kennwerte:
  - Minimum
    - Ende eines Whiskers (max. 1,5 fache des IQR)
  - Unteres Quantil (kleinsten 25%)
    - Beginn der Box
  - Median
    - Strich innerhalb der Box
  - Oberes Quantil (kleinsten 75%)
    - Ende der Box
  - Maximum
    - Ende eines Whiskers (max 1,5 fache des IQR)

Wdh.: IQR gibt an, wie weit das obere und das untere Quartil auseinanderliegen und damit auch, wie breit der Bereich ist, in dem die mittleren 50 % der Stichprobe liegen.

Visualisierung des Titanic Datasets

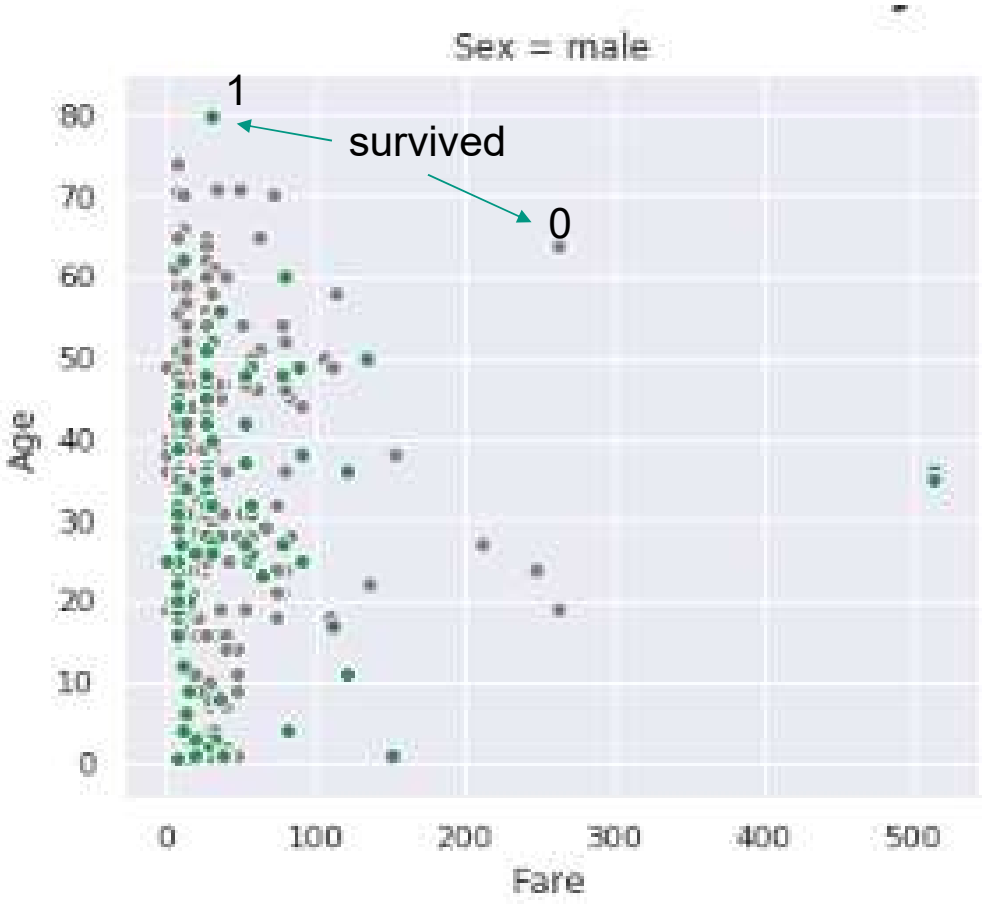
Boxplots



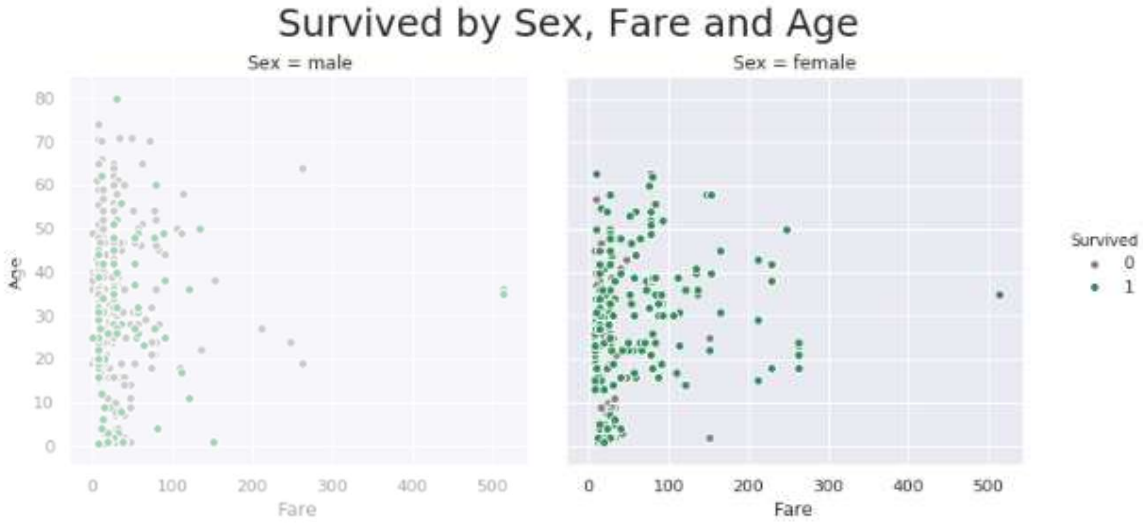
- Auch *Box-Whisker Plot* (deutsch: *Kastengrafik*)
- Kennwerte:
  - Minimum
    - Ende eines Whiskers (max. 1,5 fache des IQR)
  - Unteres Quantil (kleinsten 25%)
    - Beginn der Box
  - Median
    - Strich innerhalb der Box
  - Oberes Quantil (kleinsten 75%)
    - Ende der Box
  - Maximum
    - Ende eines Whiskers (max 1,5 fache des IQR)

Visualisierung des Titanic Datasets

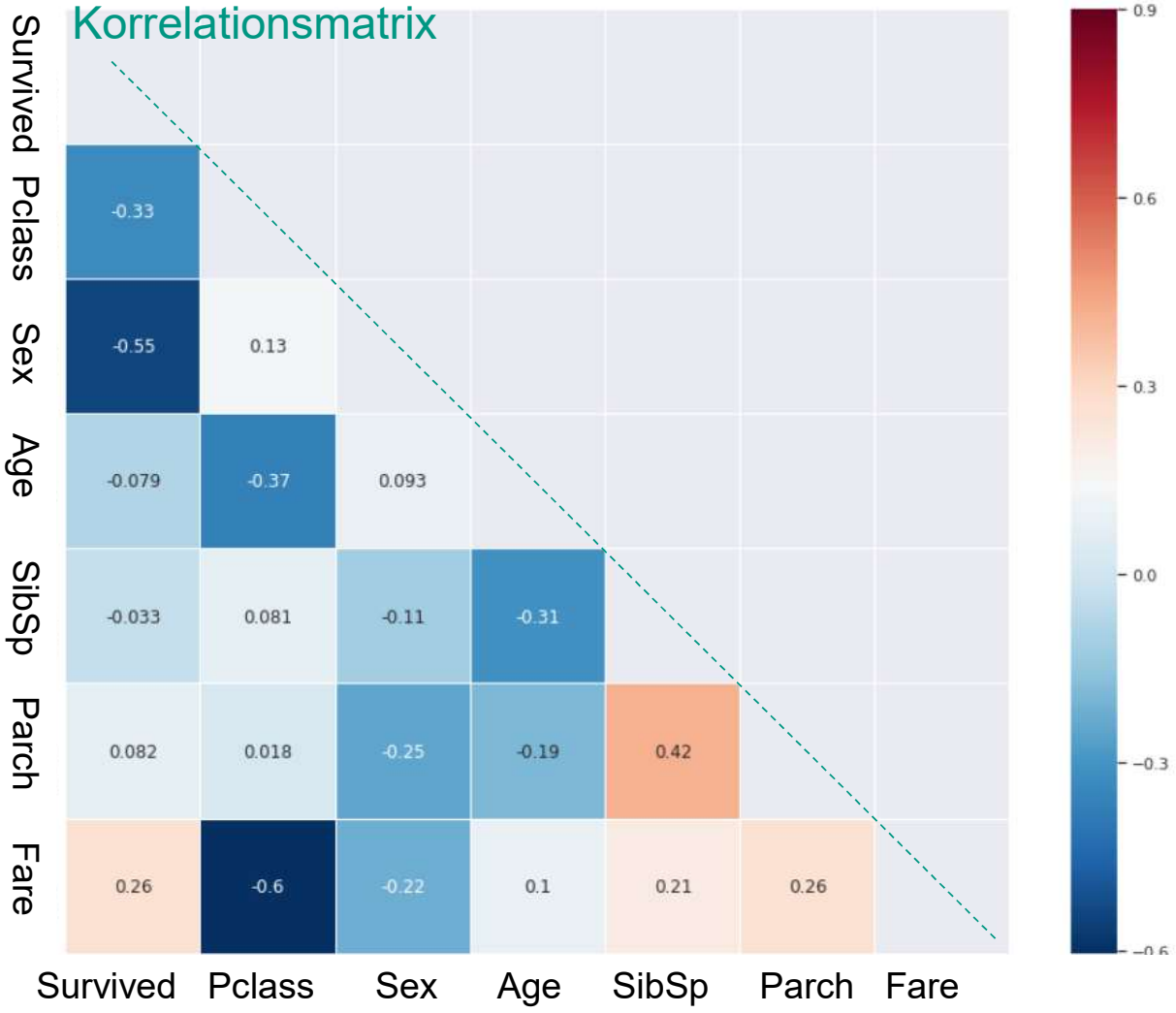
Scatter Plot



- Streudiagramm, Punktwolke
- Beobachtung von Wertepaaren zweier Merkmale (hier: *Fare vs Age*)
- Punkte werden in ein kartesisches Koordinatensystem eingetragen
- Darstellung der Punkte kann durch verschiedene Symbole oder Farben erfolgen



Visualisierung des Titanic Datasets

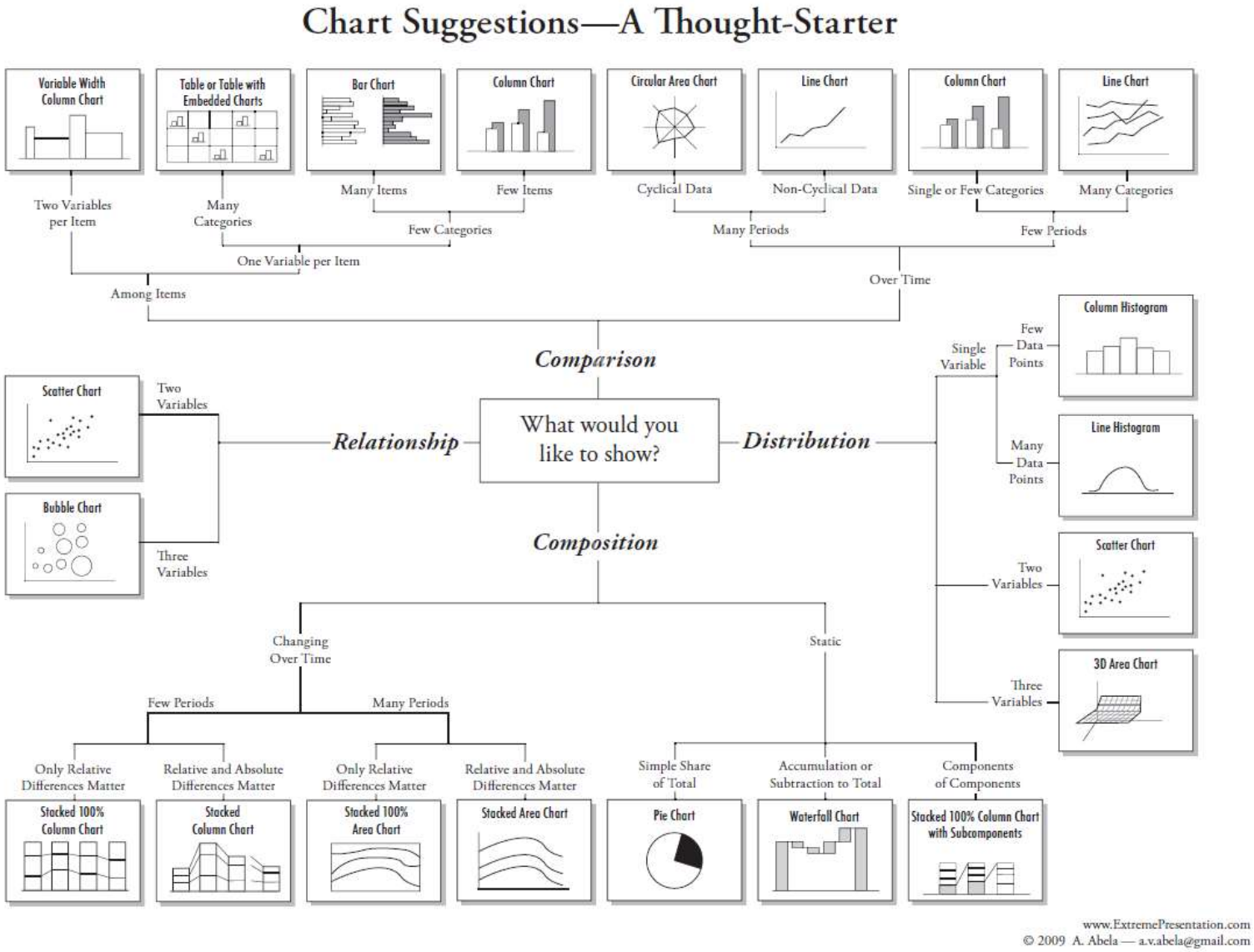


- *Positive Correlation Features*:
  - „Wenn Merkmal x wächst, wächst auch Merkmal y“
    - ▶ Fare and Survived: 0.26
- *Negative Correlation Features*:
  - Antikorrelation, d.h. „wenn Merkmal x wächst, sinkt Merkmal y“
    - ▶ Fare and Pclass: -0.6
    - ▶ Sex and Survived: -0.55
    - ▶ Pclass and Survived: -0.33

1, 2, 3

SURVIVED	Überlebensvariable mit 0=Nein und 1=Ja
PCLASS	Klasse auf dem Schiff 1= Erste, 2=Zweite, 3= Dritte
NAME	Nachname, Title, Vorname(n)
SEX	Geschlecht „male“=männlich, „female“=weiblich
AGE	Alter in Jahren
SIBSP	Anzahl der Geschwister bzw. Partner, die mit an Bord waren
PARCH	Anzahl der Eltern bzw. Kinder, die mit an Bord waren
TICKET	ID Nummer des Tickets
FARE	Preis des Tickets
CABIN	Kabinennummer
EMBARKED	Einstiegshafen C= Cherbourg, Q= Queenstown, S=Southampton

Graphische Datenaufbereitung (Überblick)



- Data Understanding
- Merkmale, Datentypen
- (Un-)Strukturierte Daten
- Deskriptive Statistik
- Visualisierungen
  - Eindimensional
  - Multidimensional

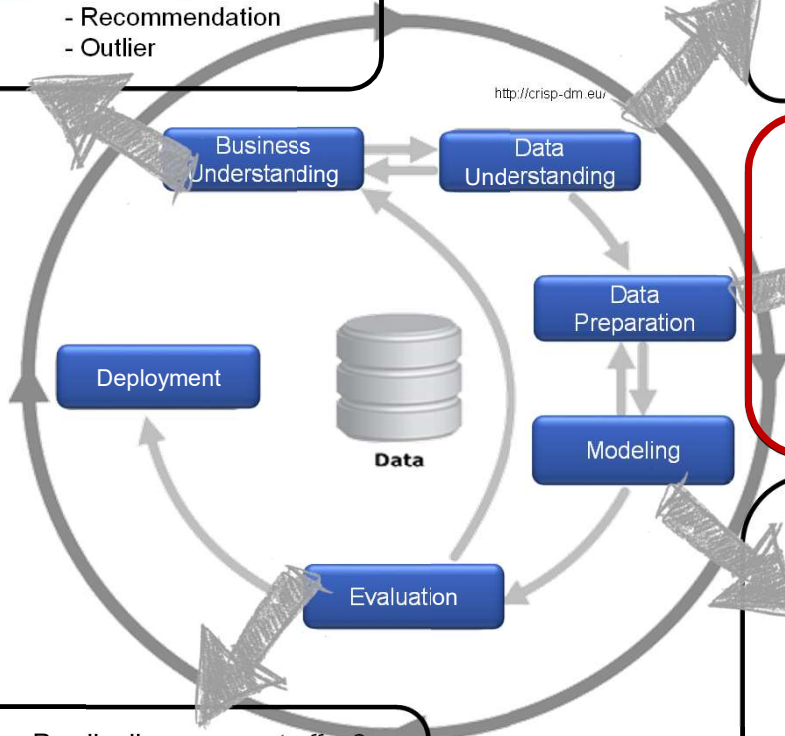


CRISP-DM im Detail

- Zielsetzung
- Rand-bedingungen

Analytic Aproach

- Regression
- Klassifikation
- Clustering
- Recommendation
- Outlier



Daten
- Struktur
- Statistik
- Visualisierung

Collect Initial Data
- Feature
- Type
- Struktur

Describe & Explore Data
- Statistik: Mean,Var, Median
- Visualisierung: Plots

Daten
Vor- & Aufbe-reitung

Select & Clean Data
- Auswahl
- Konsistenz
- Bereinigen

Construct Data
- Aggregation
- Aufteilung

Format Data
- Syntaktische Modifzierung
- Transformation
- Scaling

Algorithmus Selektion

Modeling Technique
- Group by Learning Style
  - Un- vs Supervised
  - Semi
  - Reinforcement
- Group by Similarity
  - Regression, Neighbor, Clustering, Decision Tree, NN, Bayes

Testdesign
- Splitting
  - Test, Train, Validate

Build & Assess Model
- ROC
- Confusion Matrix
- Cross Validation
- Precision/Recall
- AUC

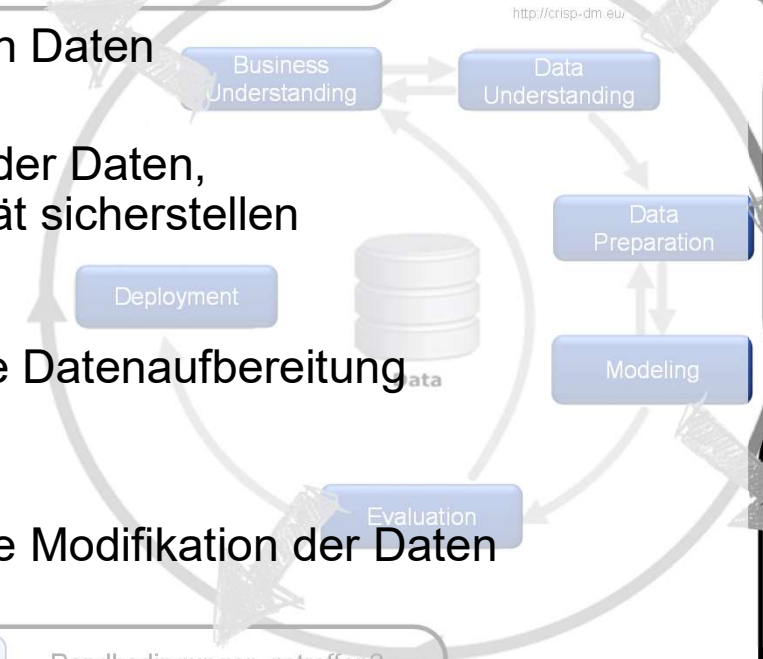
Ergebnisse

Evaluate Results
- Randbedingungen getroffen? HW Constraints
- Summarize assessment results

Review of process
- Summarize process and highlight activites

CRISP-DM im Detail

- Auswahl von Daten
- Bereinigen der Daten, Datenqualität sicherstellen
- Konstruktive Datenaufbereitung
- Syntaktische Modifikation der Daten



- Zielsetzung
- Rand-bedingungen

Analytic Aproach

- Regression
- Klassifikation
- Clustering
- Recommendation
- Outlier

Daten
- Struktur
- Statistik
- Visualisierung

Collect Initial Data
- Feature
- Type
- Struktur

Describe & Explore Data
- Statistik: Mean,Var, Median
- Visualisierung: Plots

Daten Vor- & Aufbe-reitung

Select & Clean Data
- Auswahl
- Konsistenz
- Bereinigen

Construct Data
- Aggregation
- Aufteilung

Format Data
- Syntaktische Modifzierung
- Transformation
- Scaling

Algorithmus Selektion

Modeling Technique
- Group by Learning Style
  - Un- vs Supervised
  - Semi
  - Reinforcement
- Group by Similarity
  - Regression, Neighbor, Clustering, Decision Tree, NN, Bayes
- Splitting
  - Test, Train, Validate

Testdesign

Build & Assess Model
- ROC
- Confusion Matrix
- Cross Validation
- Precision/Recall
- AUC

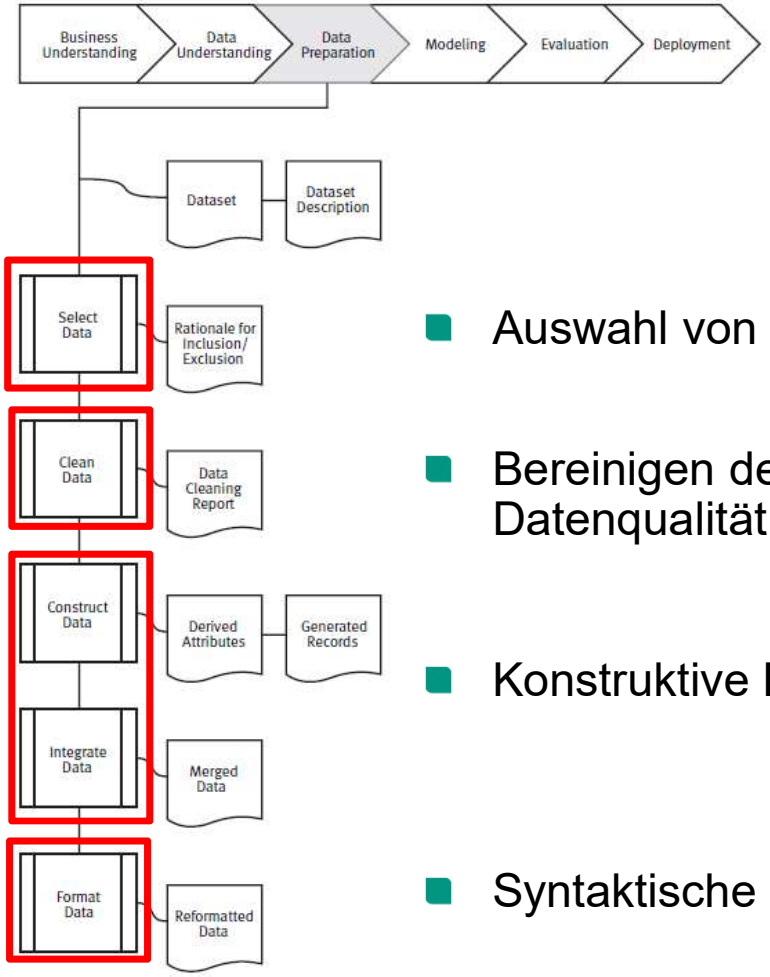
Ergebnisse

Evaluate Results
- Randbedingungen getroffen? HW Constraints
- Summarize assessment results

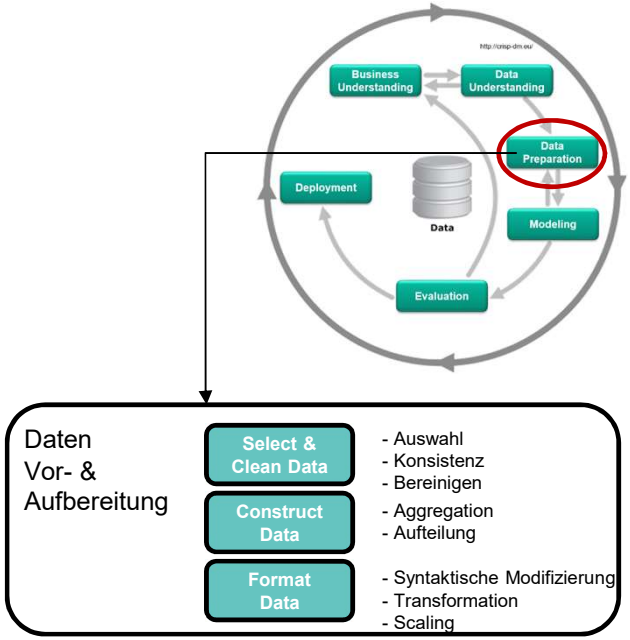
Review of process
- Summarize process and highlight activites

BN(7

CRISP-DM im Detail: Data Preparation



- Auswahl von Daten
- Bereinigen der Daten, Datenqualität sicherstellen
- Konstruktive Datenaufbereitung
- Syntaktische Modifikation der Daten

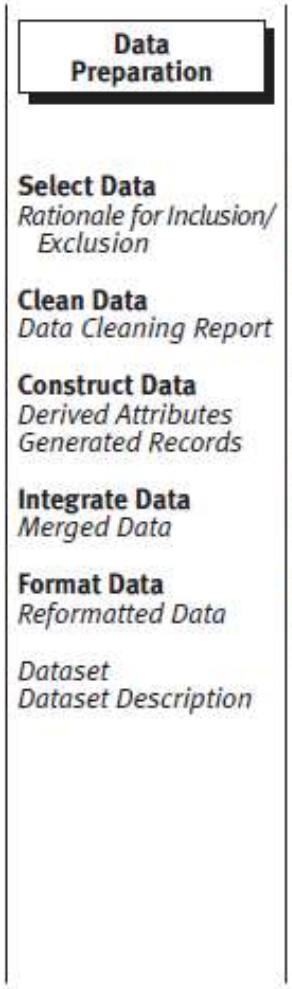


BN(7 Integration, PCA, noch weitere?

Brenner, Nathalie (ITIV); 31.10.2018

BN(7

CRISP-DM im Detail: Data Preparation



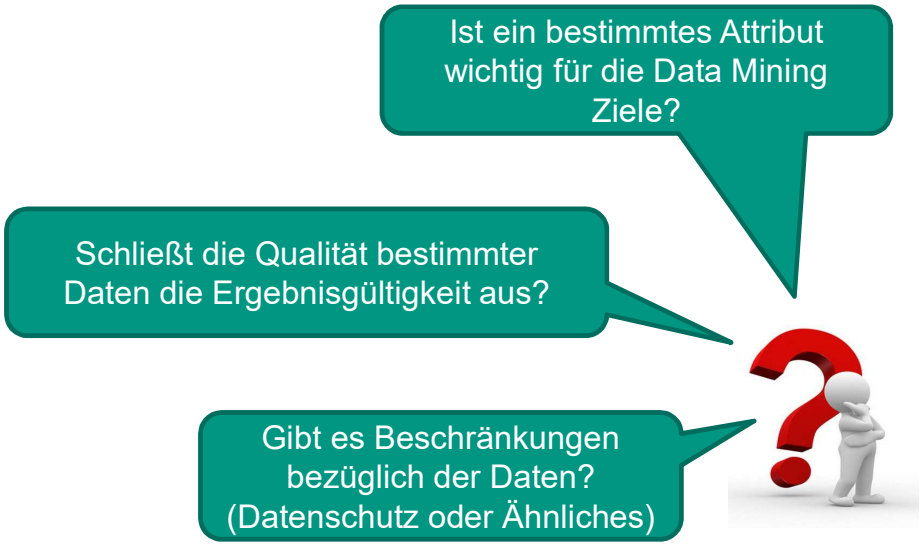
- Auswahl von Daten
- Bereinigen der Daten, Datenqualität sicherstellen
- Konstruktive Datenaufbereitung
- Syntaktische Modifikation der Daten
- Hauptkomponentenanalyse / PCA (Principal Component Analysis)

BN(7 Integration, PCA, noch weitere?

Brenner, Nathalie (ITIV); 31.10.2018

Auswahl von Daten
Data Preparation

- **Aktivitäten um den für die Kategorisierung verwendeten Datensatz aus den Rohdaten zu erhalten**
- **Umsetzungsschritte:**
  - Einbeziehung geeigneter zusätzlicher Daten aus verschiedenen Quellen (intern und extern)
  - Signifikanz und Korrelationstests
  - Aufstellen der Datenauswahlkriterien hinsichtlich „Erfahrungswerten“ mit
    - Datenqualität
    - Modellierungsverfahren
  - Auswahl der Datensätze
- **Methoden zur Datenauswahl**
  - Auswahl von Elementen
  - Auswahl von Attributen oder Merkmalen



Bereinigen der Daten - Fehlerarten

Data Preparation

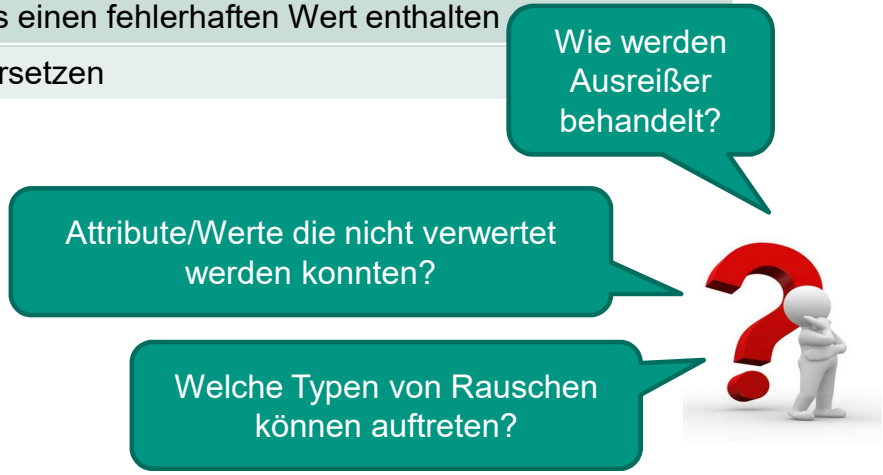
- **Ausgleichen zufälliger Fehler**
  - Mess- und Übertragungsfehler können als additives Rauschen modelliert werden
    - ► Gaußsches Rauschen mit Mittelwert 0 (Normalverteilung)
- **Erkennen systematischer Fehler**
  - Fehlerhafte Formeln/Berechnung von Merkmalen
  - Fehlerhafte Kalibrierung von Messgeräten
  - Ausreißer (können auch zufällige Effekte sein)
    - Lokale/globale Ausreißer → weichen stark von der Verteilung der übrigen Daten ab
  - Ungültige Daten
    - Merkmale liegen außerhalb des zulässigen Wertebereichs x^i_{min} und x^i_{max}
      x^i_k ist ungültig, wenn $\left(x^i_k < x^i_{min}\right)$ v $\left(x^i_k > x^i_{max}\right)$



Bereinigen der Daten - Behandlung von fehlerhaften Daten

Data Preparation

Datenproblematik	Umsetzungsschritte zur Lösung
Fehlerliste	Daten bleiben unverändert, Indizes werden in separater Liste gespeichert
Fehlerwert	Ungültige Daten werden durch Fehlerwert (z.B. *Not a Number, NaN*) ersetzt
Korrektur/Schätzung	Nur einzelne Merkmalswerte werden korrigiert oder geschätzt • Ersetzen durch Mittelwert, Median, Minimum, Maximum • Ersetzen durch nächsten Nachbarn • lineare oder nichtlineare Interpolation • Schätzung durch Regression • Filterung
Entfernen ganzer Merkmalsvektoren	Entfernen aller Merkmalsvektoren, welche mindestens einen fehlerhaften Wert enthalten
Entfernen ganzer Merkmale	Entfernen aller Merkmale, welche mindestens einen fehlerhaften Wert enthalten
Codierungsinkonsistenzen	Codierungsschema, Werte umrechnen und ersetzen



Konstruktive Datenaufbereitung
Data Preparation

- Attribute ableiten
  - Welches weitere Attribut kann den Informationsgehalt der Daten erhöhen?
  - Welches weitere Attribut wird für vorgesehenen Data Mining Algorithmus benötigt?
- Attribute generieren
  - Aufstellen und Ausführung der mathematischen Formeln/Berechnung des neuen Attributes
- Daten zusammenführen
  - Ursprüngliche Daten und neu generierte Daten in einen Datensatz zusammenführen

Name	PClass	Fare	Embarked
Wirz, Mr. Albert	3	86.625	Cherbourg
Connolly, Miss. Kate	3	76.292	Queenstown
Fortune, Miss. Ethel Flora	1	263.00	Southhampton
Myles, Mr. Thomas Francis	2	96.875	Queenstown



Konstruktive Datenaufbereitung
Data Preparation

- Attribute ableiten
  - *Preis pro zurückgelegten Kilometer* ρ
- Attribute generieren
  - *Mathematische Beziehung:* ρ = *Ticketpreis/Gesamtkilometer*
- Daten zusammenführen
  - Ursprüngliche Daten und neu generierte Daten in einen Datensatz zusammenführen
  - S. zuvor: Korrelation nutzen und stark korrelierte Daten zusammenfassen

Name	PClass	Fare	Embarked
Wirz, Mr. Albert	3	86.625	Cherbourg
Connolly, Miss. Kate	3	76.292	Queenstown
Fortune, Miss. Ethel Flora	1	263.00	Southhampton
Myles, Mr. Thomas Francis	2	96.875	Queenstown

Ticketpreis/km ρ
0.02128
0.02280
0.06238
0.02895

BN(11

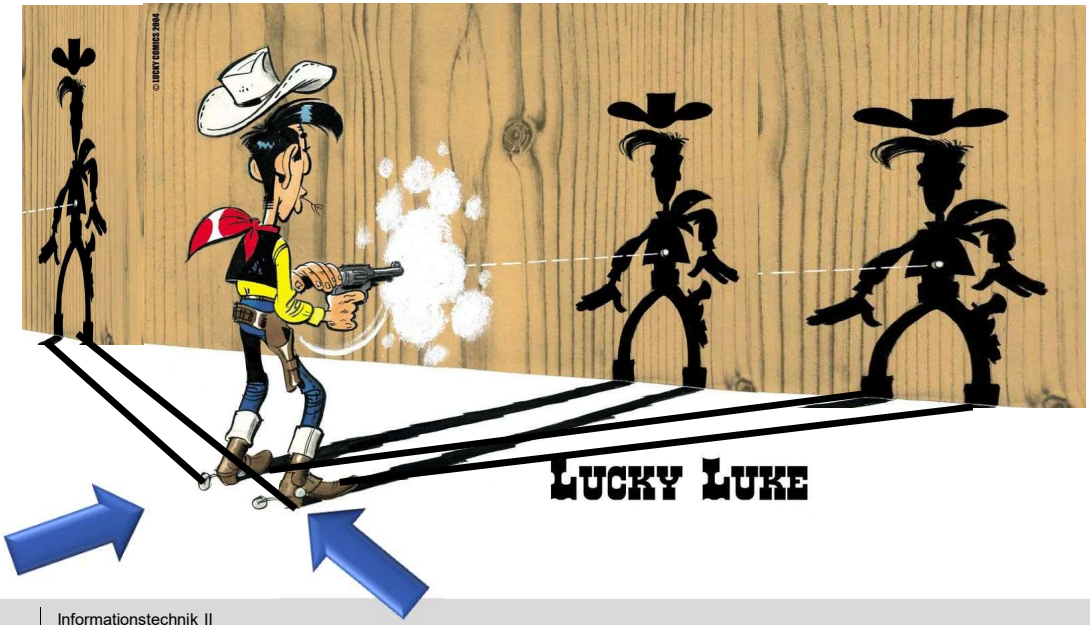
Abfahrtshafen	Gesamt kilometer
Southhampton	4216
Cherbourg	4071
Queenstown	3346

BN(11 Zeichne ich noch in Inkscape selber eins

Brenner, Nathalie (ITIV); 19.12.2018

Konstruktive Datenaufbereitung
Data Preparation

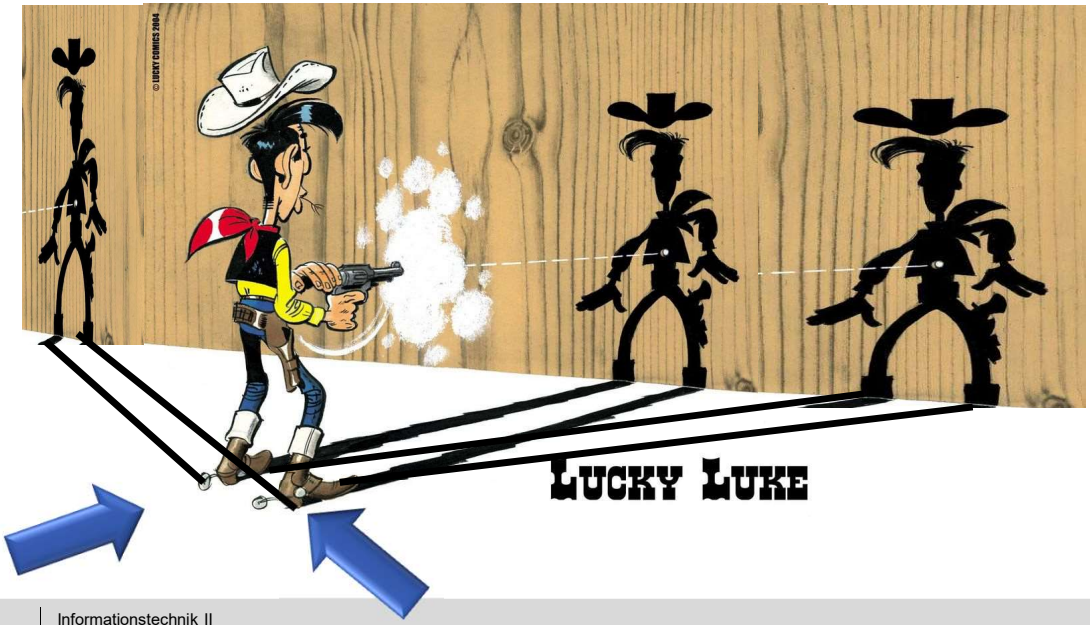
- Die Hauptkomponentenanalyse (Principal Component Analysis – PCA) ist eine variablenorientierte Methode, mit der versucht wird, einen hochdimensionalen Datensatz in einen niederdimensionalen Raum zu projizieren.



- Die zentrale Idee dabei ist, die Daten so auf eine 2-dimensionale Ebene zu projizieren, dass die gesuchten Zusammenhänge sichtbar werden.
  - Die sichtbar werdende Struktur der projizierten Daten hängt von der Richtung der Projektion ab.
  - Es stellt sich die Frage, wie eine Rotation der Daten (oder der Achsen - was für diese Zwecke dasselbe ist) gefunden werden kann, die ein Maximum an Information im projizierten Bild darstellt.

Konstruktive Datenaufbereitung
Data Preparation

- Die Hauptkomponentenanalyse (Principal Component Analysis – PCA) ist eine variablenorientierte Methode, mit der versucht wird, einen hochdimensionalen Datensatz in einen niederdimensionalen Raum zu projizieren.



- Wenn wir davon ausgehen, dass aus den Daten nur dann Informationen gewonnen werden können, wenn die Streuung ein Maximum ist, müssen wir die Richtung der maximalen Streuung der Daten finden.
  - Darüber hinaus sollten diese neuen Achsen wiederum orthogonal zueinander stehen.

- **Standardabweichung (Streuung)**
  - Streuung der Werte einer Zufallsvariablen um ihren Erwartungswert oder einer Stichprobe um den ihr eigenen Mittelwert bezeichnet man als Standardabweichung.
  - Die Standardabweichung ist die Größe ohr

97 | Informationstechnik II
Kapitel 6: Big Data & Machinelles Lernen

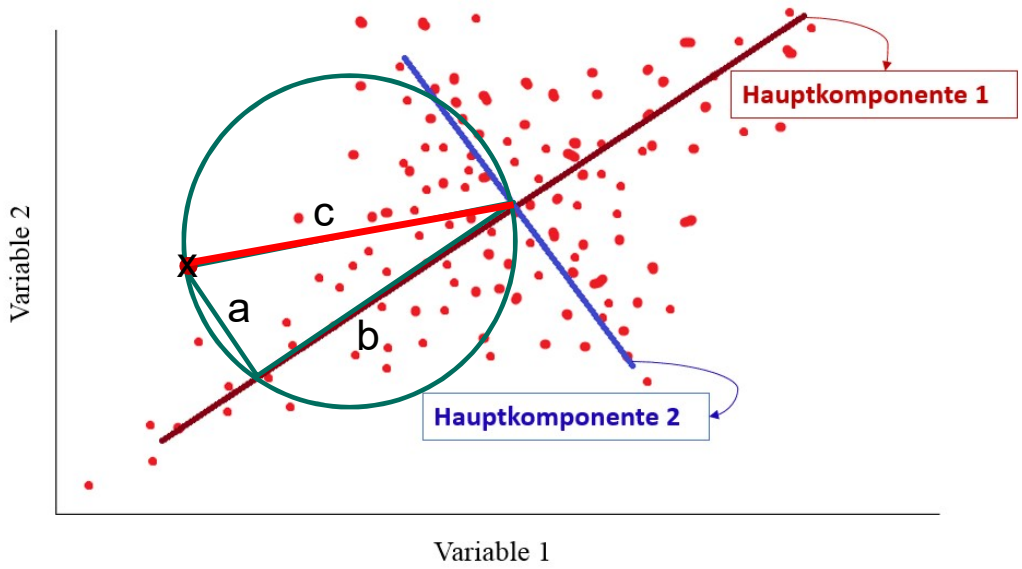
Konstruktive Datenaufbereitung
Data Preparation

- **Problem:**
  - große Datenmengen werden schnell unübersichtlich oder sind ungeeignet formatiert
  - Multivariate Daten
    - können nicht zweidimensional dargestellt werden
    - ► Können also auf nicht auf Papier oder auf dem Computerbildschirm veranschaulicht werden.
    - Je mehr Variablen (Dimensionen) ein Datensatz hat, umso komplizierter und undurchschaubarer wird die Situation.
    - ► Dies führt dazu, dass man eventuell bestehende Zusammenhänge nicht mehr erkennt.
- **Ziel:**
  - Ausgesuchte und bereinigte Daten für vorgesehenen Modellierungsalgorithmus passend transformieren
- **Lösung:**
  - Skalierung und Normierung der Daten
  - Anpassen des Datentyps

- Hauptkomponentenanalyse / PCA (Principal Component Analysis)
  - Mathematisches Verfahren der multivariaten Statistik (mehrere Variablen betreffend)
  - Vielzahl statistischer Variablen wird durch geringe Zahl möglichst aussagekräftiger Linearkombinationen angenähert

Hauptkomponentenanalyse (1)
Projektion

- Die Distanz zwischen dem Zentrum der Daten (Mittelwert) und einem Datenpunkt ist unabhängig davon, welche Gerade durch das Zentrum als „Referenz“ betrachtet wird (s. die hellrote Line c vom Zentrum der Daten zum Datenpunkt).
- Mittels des Satzes von Pythagoras können wir aber den Abstand zerlegen in den Anteil in Richtung der grünen Geraden (a) und einen weiteren Anteil (b) rechtwinklig dazu.
- Eine Minimierung der Abstände a rechtwinklig zur Geraden (unter Beibehaltung des Abstands zum Datenzentrum, Länge der hellroten Linie) bedeutet also eine Maximierung von b ($a^2 + b^2 = c^2$ muss erhalten bleiben).

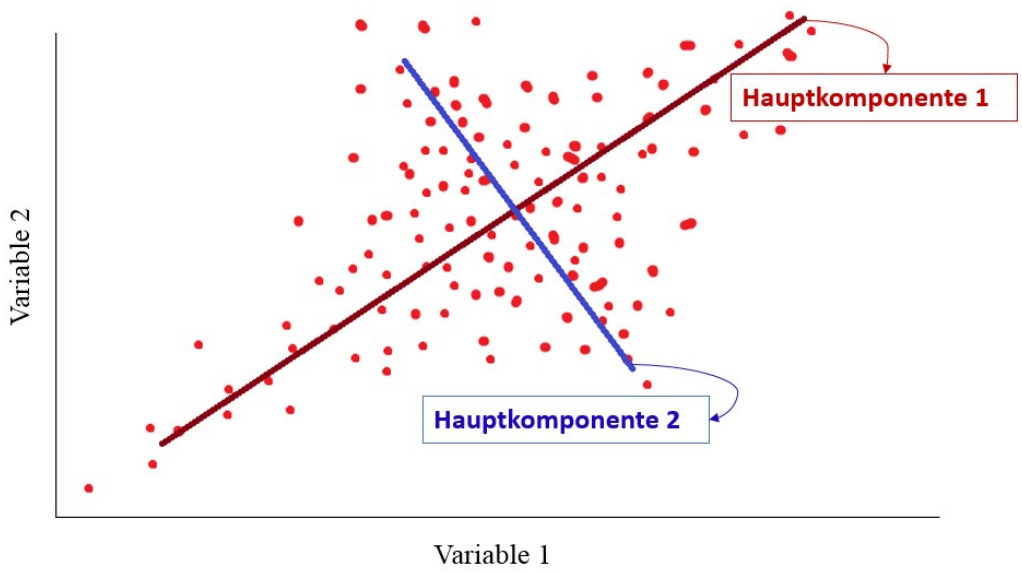


- Die aufsummierten Quadrate der Abstände in gleicher Richtung wie die dunkelrote Gerade bilden die Varianz der Daten in dieser Richtung.

Hauptkomponentenanalyse

Hauptkomponenten

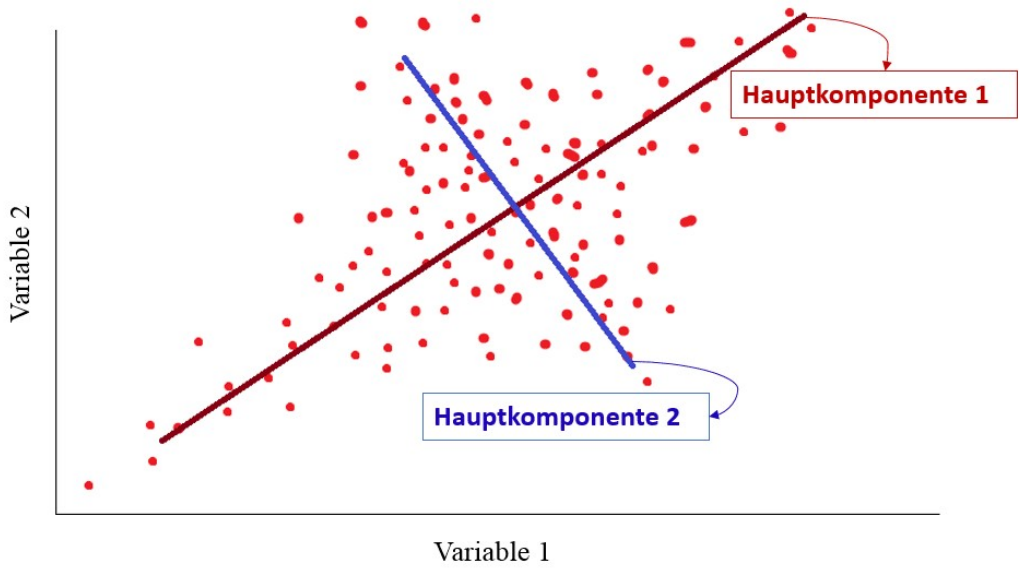
- In der nebenstehenden Abbildung ist die Funktionsweise der Hauptkomponenten-analyse schematisch dargestellt.
  - Die Kleinstquadratmethode legt die dunkelrote Gerade durch die Datenwolke und bildet die Hauptkomponente 1.
  - Hauptkomponente 2 resultiert aus der blauen Geraden, welche eine Varianz aufklärt, die durch die erste Hauptkomponente kaum erfasst werden kann.
- Kleinstquadrat-Methode → die Summe der Quadrate aller Fehler wird minimal (s. Regression, 6.2b)
  - Fehler ist hierbei der Abstand zur Geraden (s. zuvor a).
  - Ziel ist die Minimierung des mittleren quadratischen Fehlers



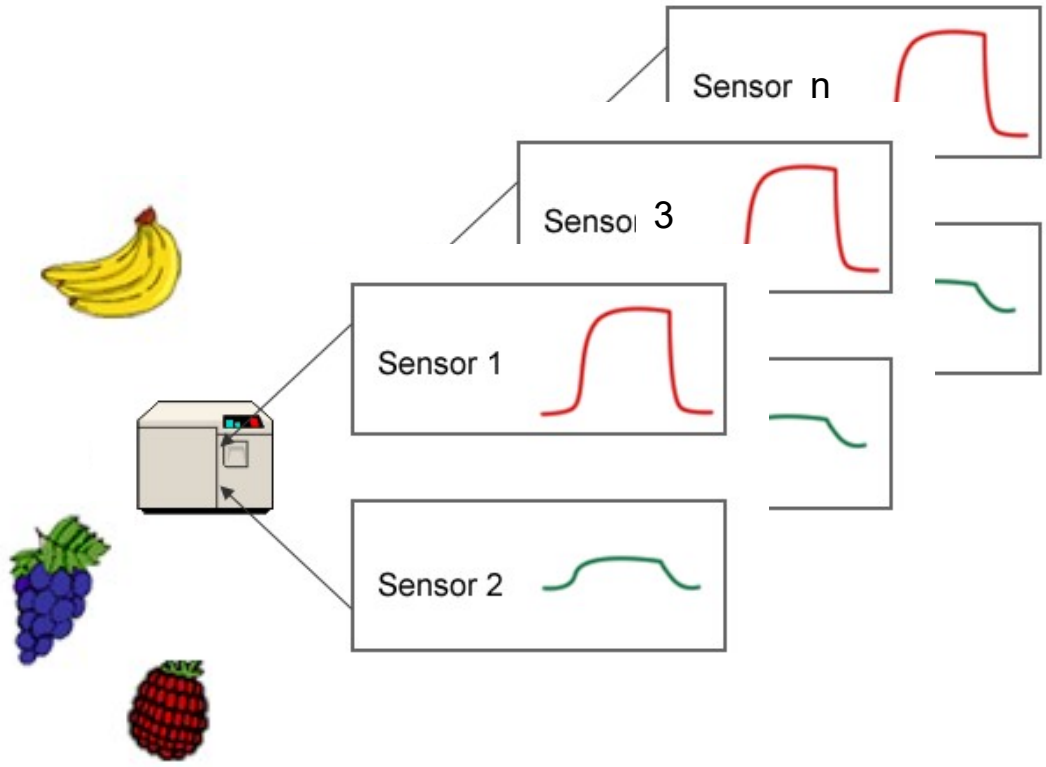
Hauptkomponentenanalyse (2)

Hauptkomponenten

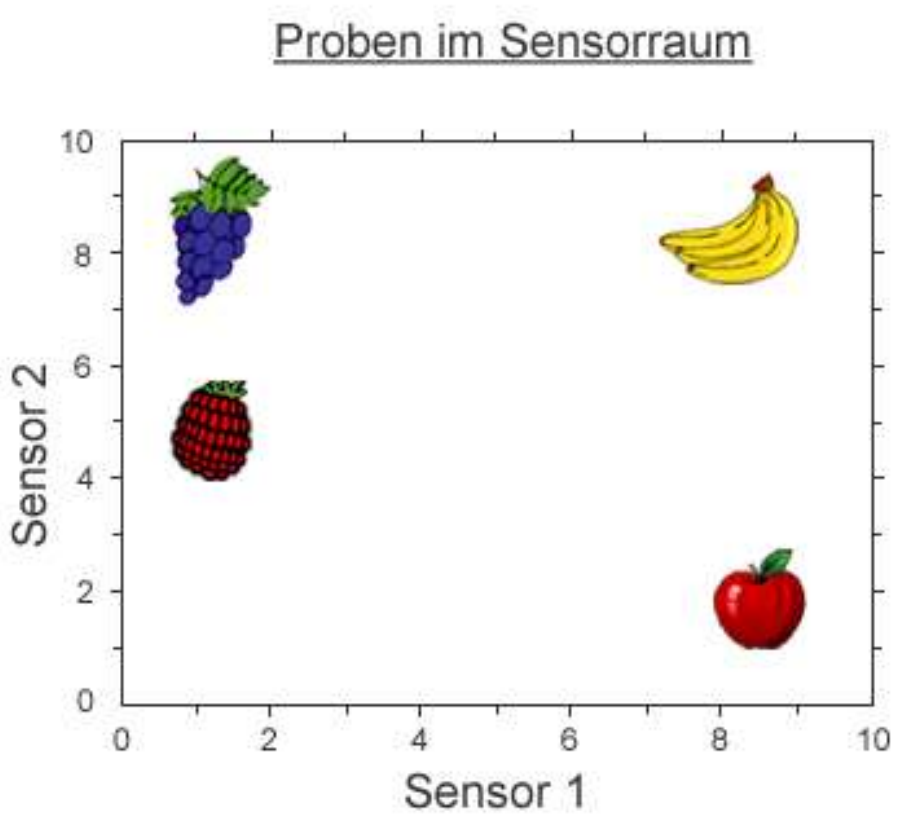
- Gegeben
  → Daten liegen als Punktwolke vor
- Gesucht
  → beste lineare Approximation
  - D.h. diejenige Gerade, welche die Daten am besten approximiert.
  - Fehler ist hierbei der (euklidische) Abstand zur Geraden.
    - → Hauptkomponente 1 ist die Gerade, bei der die Summe der Quadrate aller Fehler bzw. der mittlere quadratische Fehler (MQF, s. nächste Folie) minimal ist.
    - → Hauptkomponente 2 ist die Gerade, die den Mittelwert der Daten wiedergibt und orthogonal zur Hauptkomponente 1 liegt

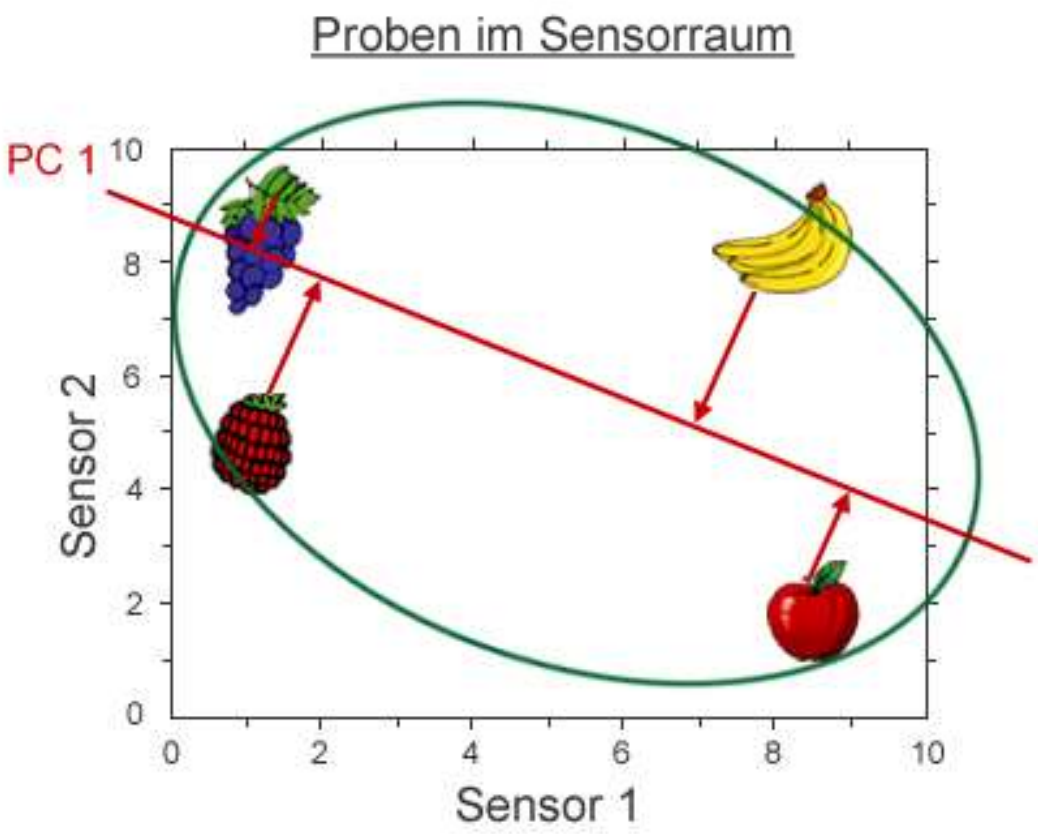


Beispiel

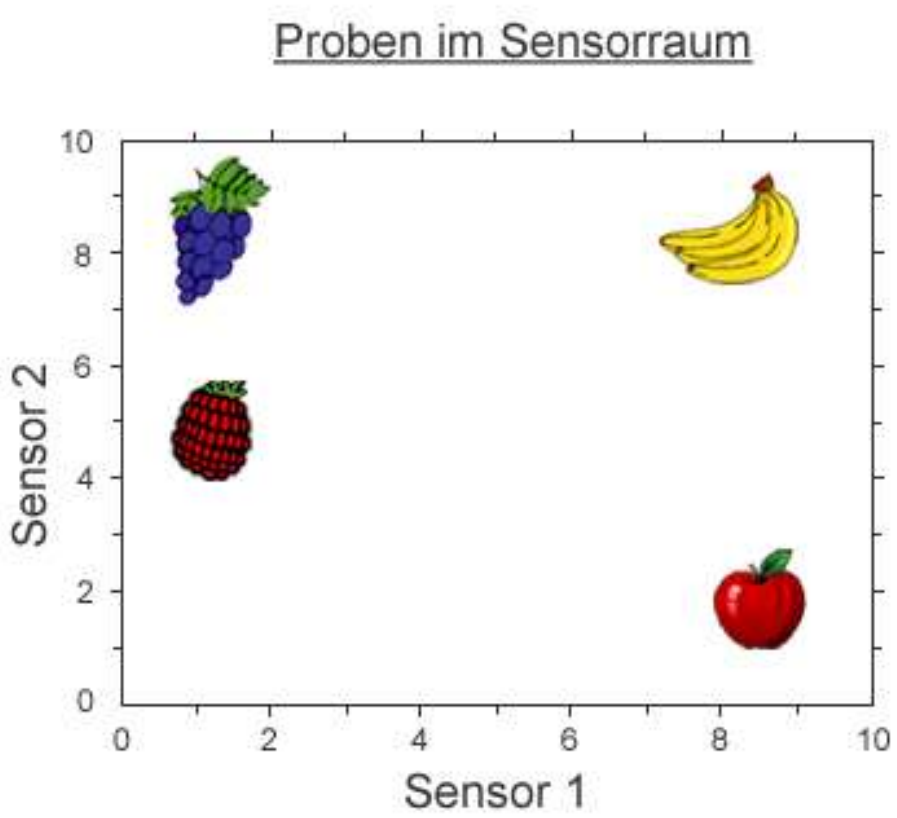


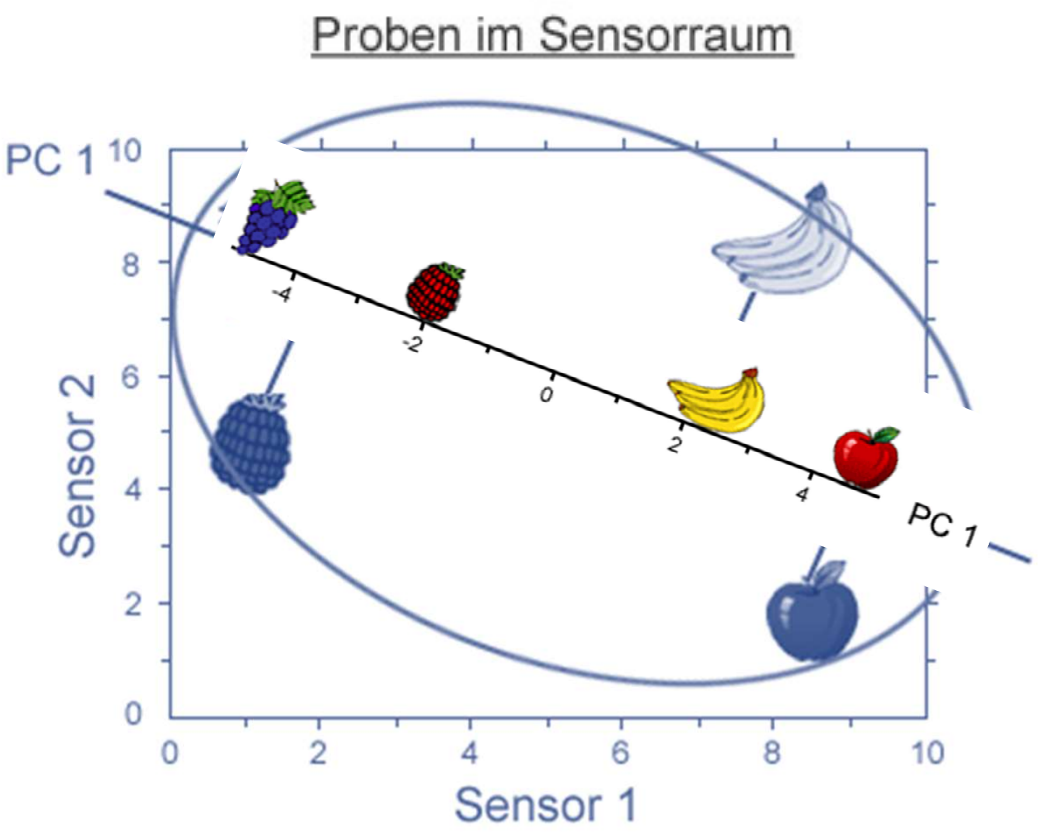
Beispiel


Proben im Sensorraum
Sensor 2
Sensor 1
0
2
4
6
8
10

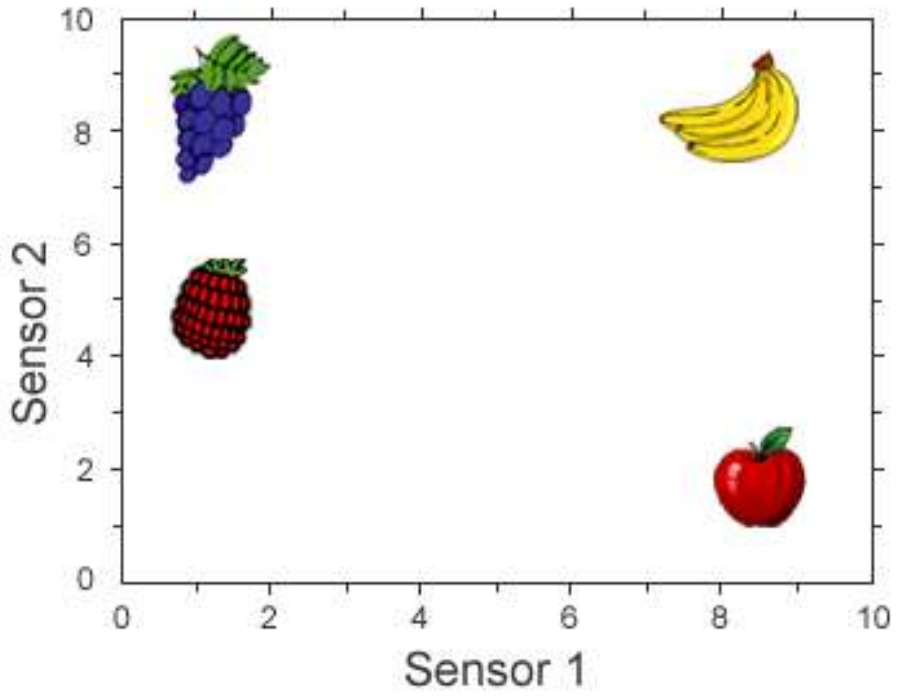

Proben im Sensorraum
PC 1
Sensor 2
Sensor 1
0
2
4
6
8
10

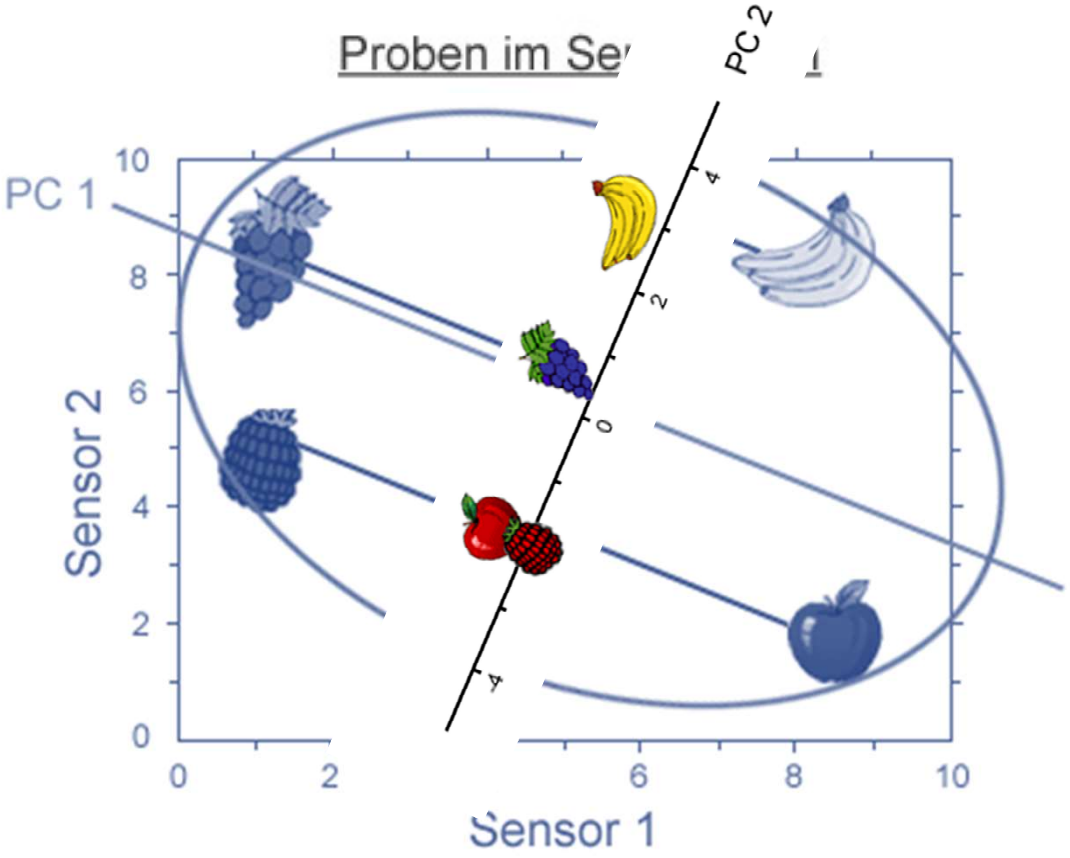
Beispiel


Proben im Sensorraum
Sensor 2
Sensor 1
0
2
4
6
8
10


Proben im Sensorraum
PC 1
PC 1
Sensor 2
Sensor 1
-4
-2
0
2
4
0
2
4
6
8
10

Beispiel


Proben im Sensorraum
Sensor 2
Sensor 1
0
2
4
6
8
10

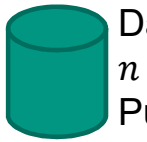

PC 1
PC 2
Sensor 2
Sensor 1
0
2
4
6
8
10
-4
-2
0
2
4

Zusammenfassung PCA

- Die Interpretation und Visualisierung der Daten in einem niederdimensionalen Raum ist im Allgemeinen einfacher
  - ► insbesondere bei Reduktion auf zwei oder drei Dimensionen wegen der menschlichen Vorstellkraft).
  - ► Die Hauptkomponentenanalyse wird daher hauptsächlich als Vorbereitung für die grafische Analyse und Visualisierung von Daten verwendet
- Hauptkomponenten können auch als Input für weitere Analysen verwendet werden.
  - Das macht jedoch bei vielen multidimensionalen Verfahren keinen Sinn, da diese Verfahren eigene Methoden besitzen, um den erklärenden Anteil der einzelnen Dimensionen zu ermitteln.

Transformation der Daten mit PCA

Data Preparation



Datenset als
n x m Matrix
Punktwolke

Mittelwert berechnen $\mu = \Sigma x_n / n$
Daten um Mittelwert verschieben

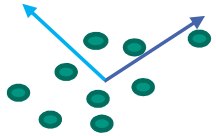
Normiertes Datenset



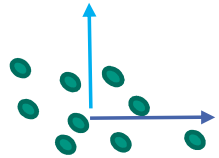
Kovarianzmatrix
$$Cov(X,Y) := E\left[\left(X - E(X)\right) * \left(Y - E(Y)\right)\right]$$
Eigenwerte $(X - \lambda E) = 0$
Eigenvektoren $(X - \lambda_i E) * x_i = 0$
berechnen

Für Interessierte

Streuung in Richtung der Eigenvektoren, Gewichtet über die Eigenwerte

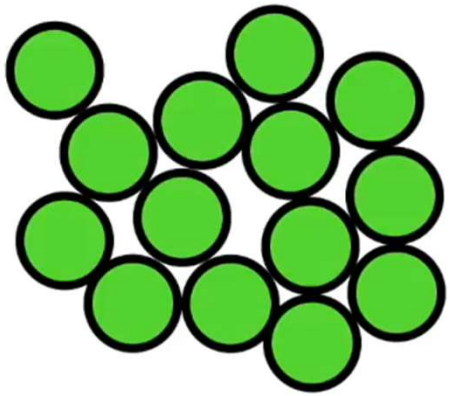


Transformationsmatrix aus Eigenvektoren auf Daten anwenden → Rotation der Daten



Daten in Abhängigkeit der Komponente mit der höchsten Abhängigkeit/ Information

Ziel: Reduzierung der Dimensionalität, aber Erhalt der Repräsentationsfähigkeit der Daten



Let's say we had some normal cells...

- Merkmalsauswahl
- Datenqualität
  - Missing Values, Ausreißer
- Konstruktive Datenaufbereitung
  - Aggregation
- Syntaktische Modifikation
  - Skalierung und Normierung
  - Transformation mit PCA



LQI 2019

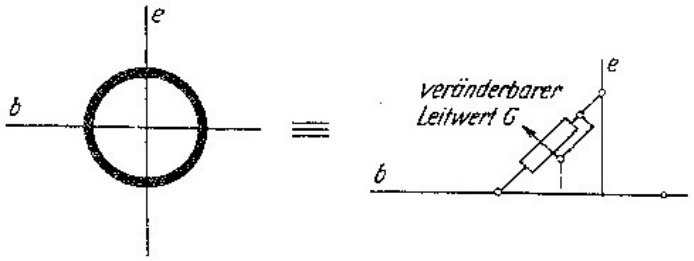
- ..\LQI\SS_19-Informationstechnik_II_und_Automatisierungstechnik.pdf

Backup

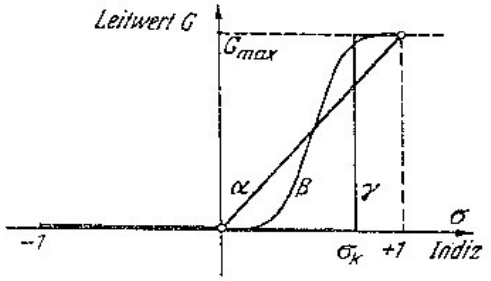
3. Die Lernmatrix
3.1 Matrixelemente

3. Die Lernmatrix
3.1 Matrixelemente

- a: Realisierung der Matrixelemente durch veränderbare Leitwerte
- b: einige Kennlinien:
  - α: lineare Widerstände
  - β: Elemente mit abgerundetem Treppenstufenverlauf
  - γ: Elemente mit Treppenstufen-charakteristik



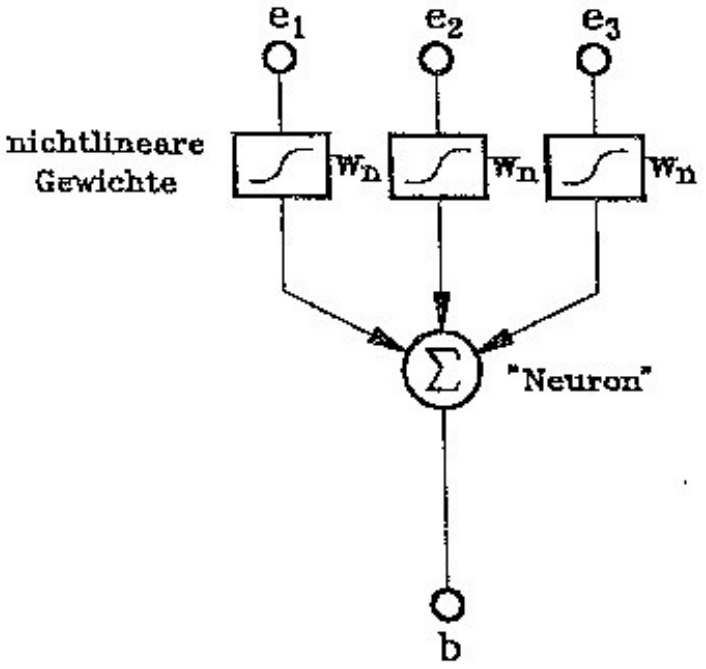
a. Ersatzschaltbild



b. Kennlinie

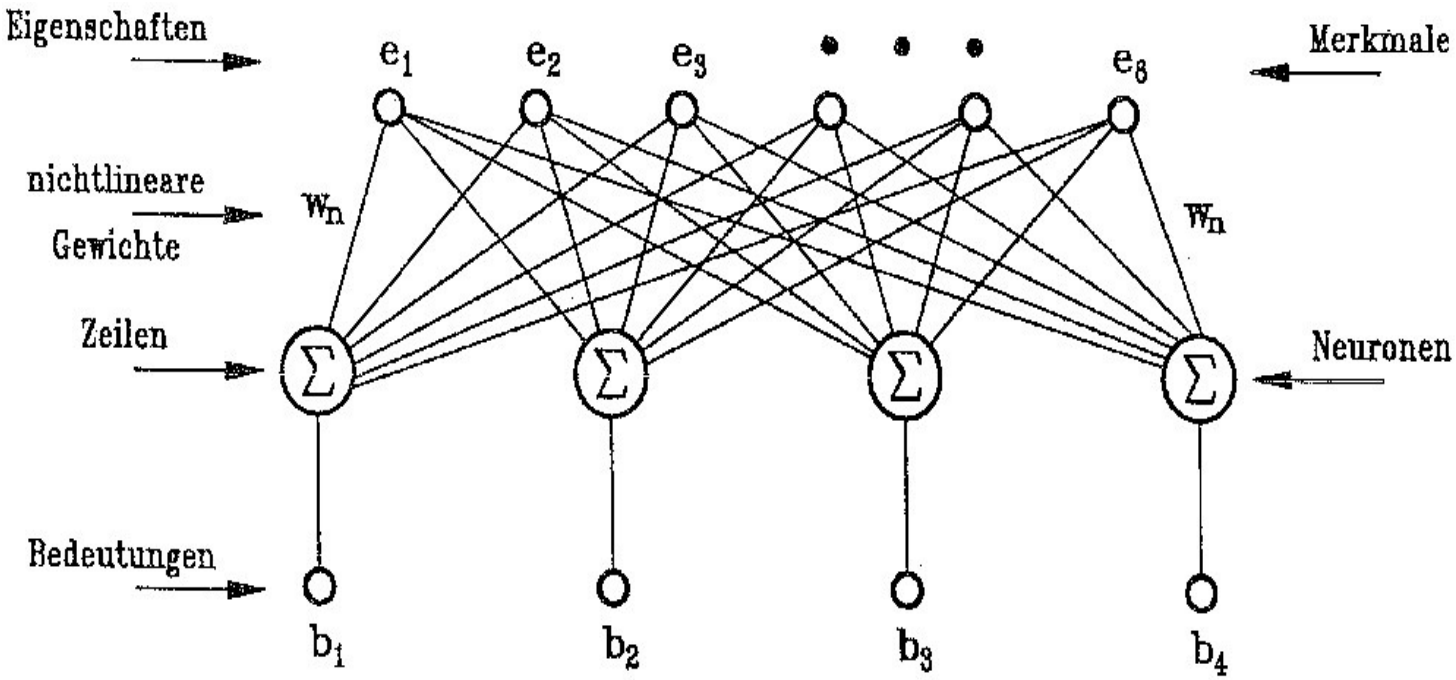
3. Die Lernmatrix
3.1 Matrixelemente

- (nichtlineare) Leitwerte befinden sich am Eingang und sind individuell einstellbar
- jede Zeile der Matrix wird zusammengefasst zu einem „Neuron“
- vom „Neuron“ geht es direkt zum Ausgang b



modernere Darstellung neuronaler Netze

3. Die Lernmatrix
3.2 als Neuronenschicht

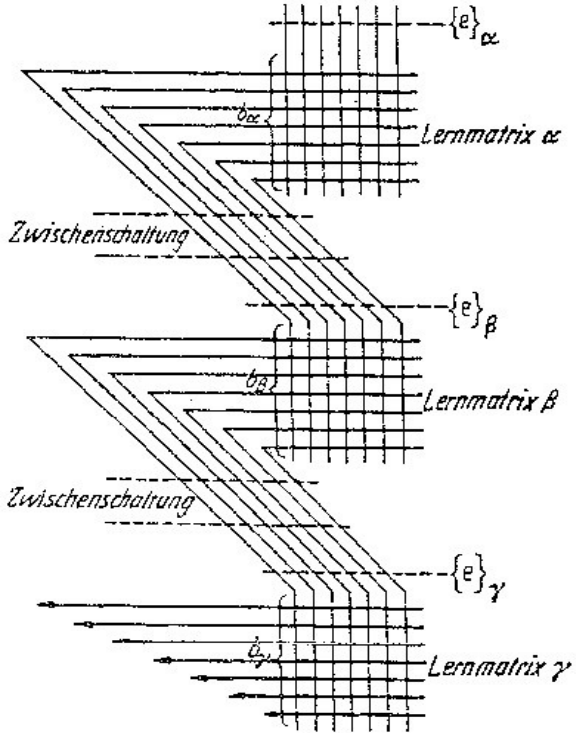


Darstellung der Lernmatrix als Neuronenschicht

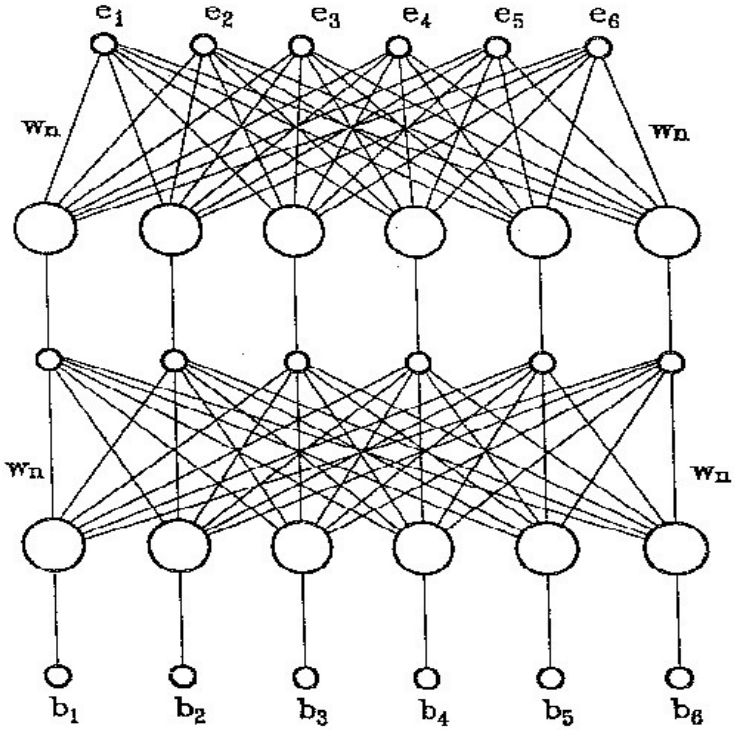
3. Lernmatrix
3.3 Schichtung

- Grundidee: die Bedeutungen dienen als neue Eigenschaften
- die neuen Eigenschaften werden jeder Zeile einer neuen Matrix zugeführt
- erhalten neuen Satz an Bedeutungen
- diese können wiederum als neue Eigenschaften interpretiert werden
- usw.

3. Die Lernmatrix
3.3 Schichtung



Schichtung von Lernmatrizen



Neuronendarstellung

4. Der Lernalgorithmus

- dem System werden Eigenschaften e und Bedeutungen b zugeführt
- Matrixelemente werden in ihren Werten verändert
- nach einer Vielzahl solcher Lernvorgänge ergibt sich eine ausreichende Veränderung der Matrixelemente, damit jedem Eigenschaftssatz die jeweilige Bedeutung ausgegeben wird
- komplementären Eigenschaftssignale e und ē repräsentieren fördernde und hemmende Einflüsse

4. Der Lernalgorithmus

- erster Nutzen und Anwendung:

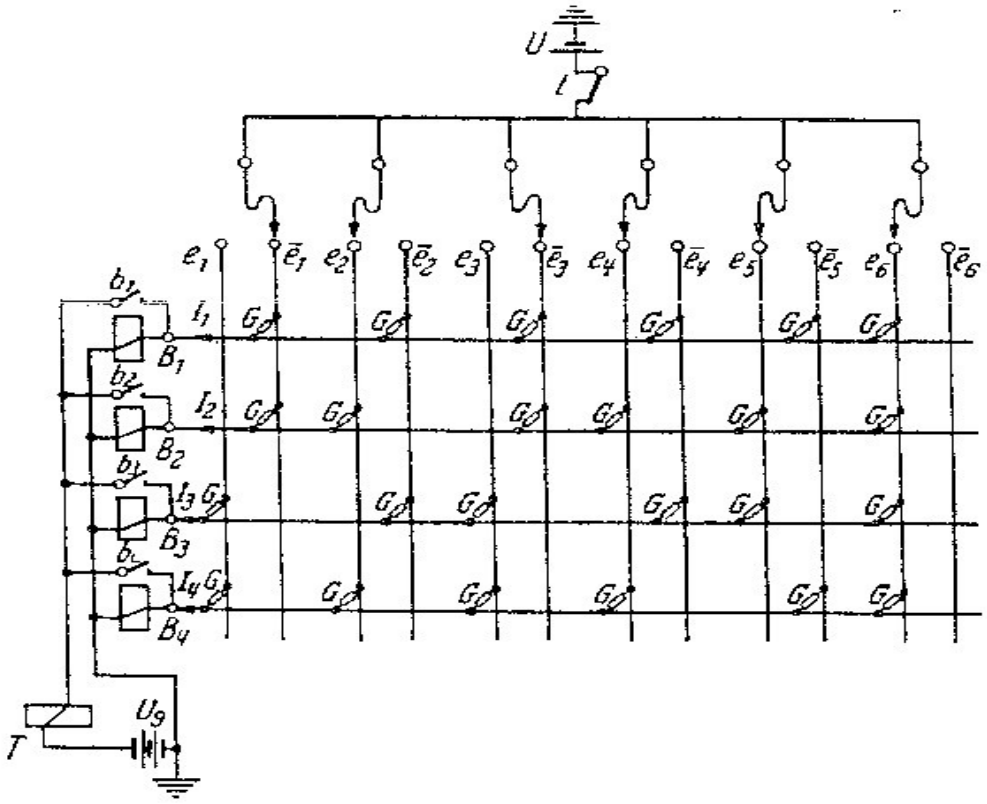
  mit ausreichendem Arbeitsaufwand ließen sich die vorgeschlagenen Systeme in Hardware realisieren
- das war vor allen Dingen ein Vorteil, wenn man bedenkt, dass es die heutzutage allseits gegenwärtigen Computer noch nicht gab

5. Die Kann-Phase

- nach Abschluss der Lernphase ist das Steinbuch´sche System einsatzbereit
- System befindet sich in der „Kann-Phase“
- Lernmatrix Bedeutungsmatrix (reine Begriffsunterscheidung)
- Matrizen haben die nicht nur Fähigkeit, erlernte Muster wiederzuerkennen, sondern sie erkennen, wenn ähnliche Muster auftauchen
- Forschung bis heute



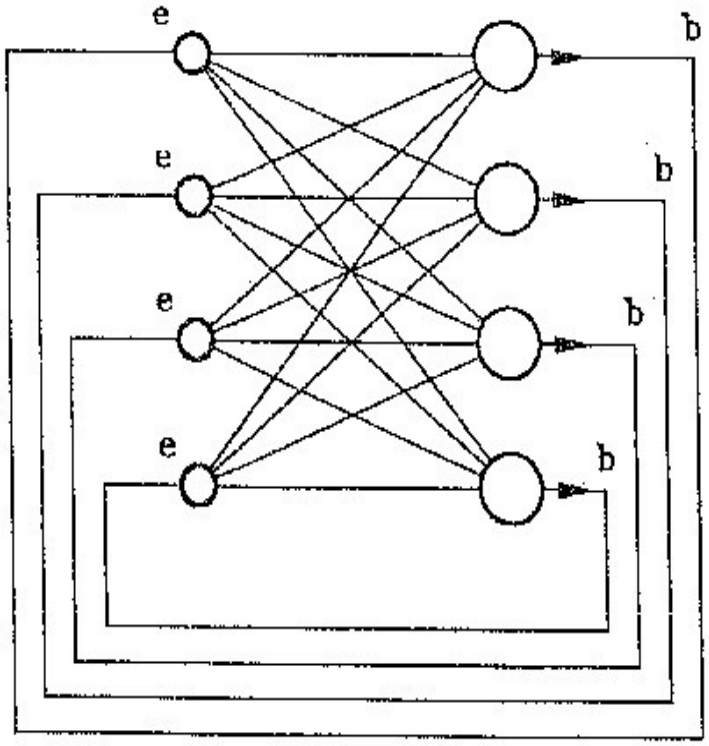
5. Die Kann-Phase



Prinzip der Bedeutungsmatrix

6. Rückkopplung

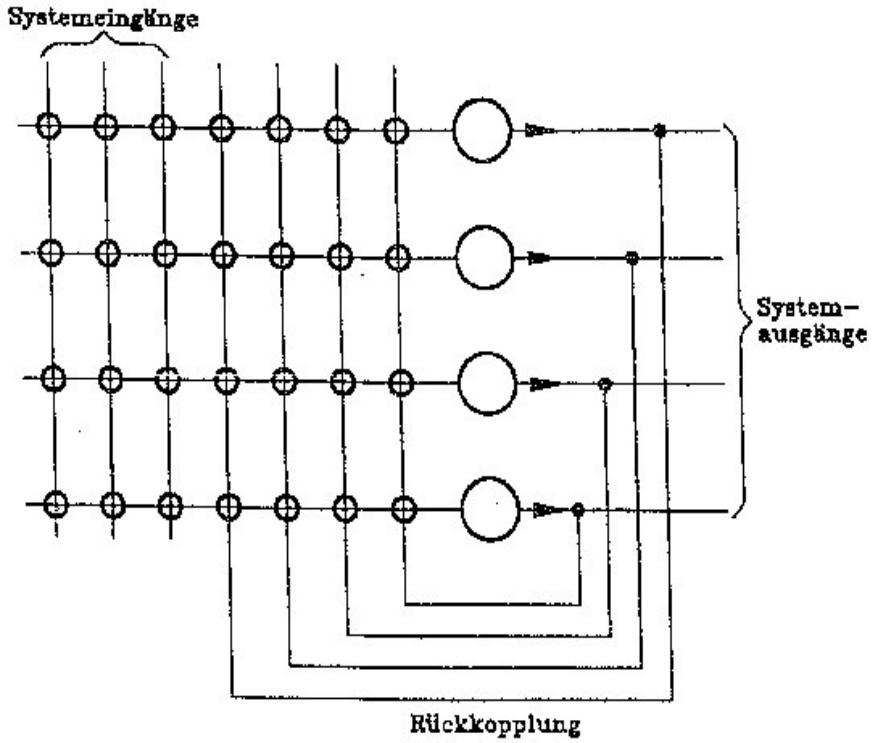
- Untersuchung des Verhaltens des Systems, wenn der Ausgang wieder mit dem Eingang verbunden wird



Neuronenschicht mit Rückkopplung

6. Rückkopplung

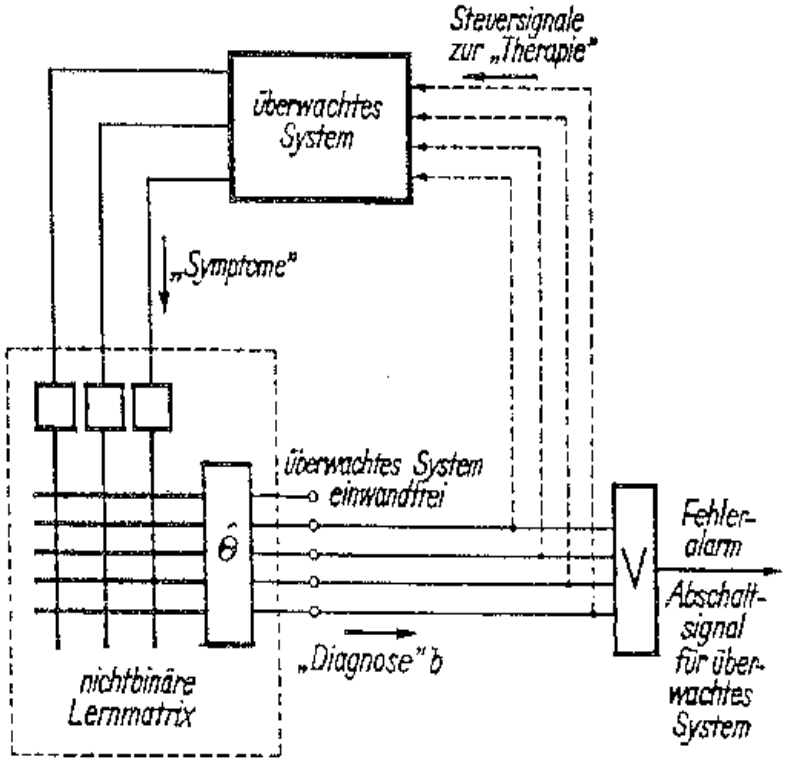
- rückgekoppelte Schaltung hat wieder Ein- und Ausgänge



rückgekoppelte Schaltung in Matrixform

6. Rückkopplung

- Anwendung: selbstprüfendes und selbstreparierendes System
- zusätzliche Information über den Zustand des Systems wird gespeichert



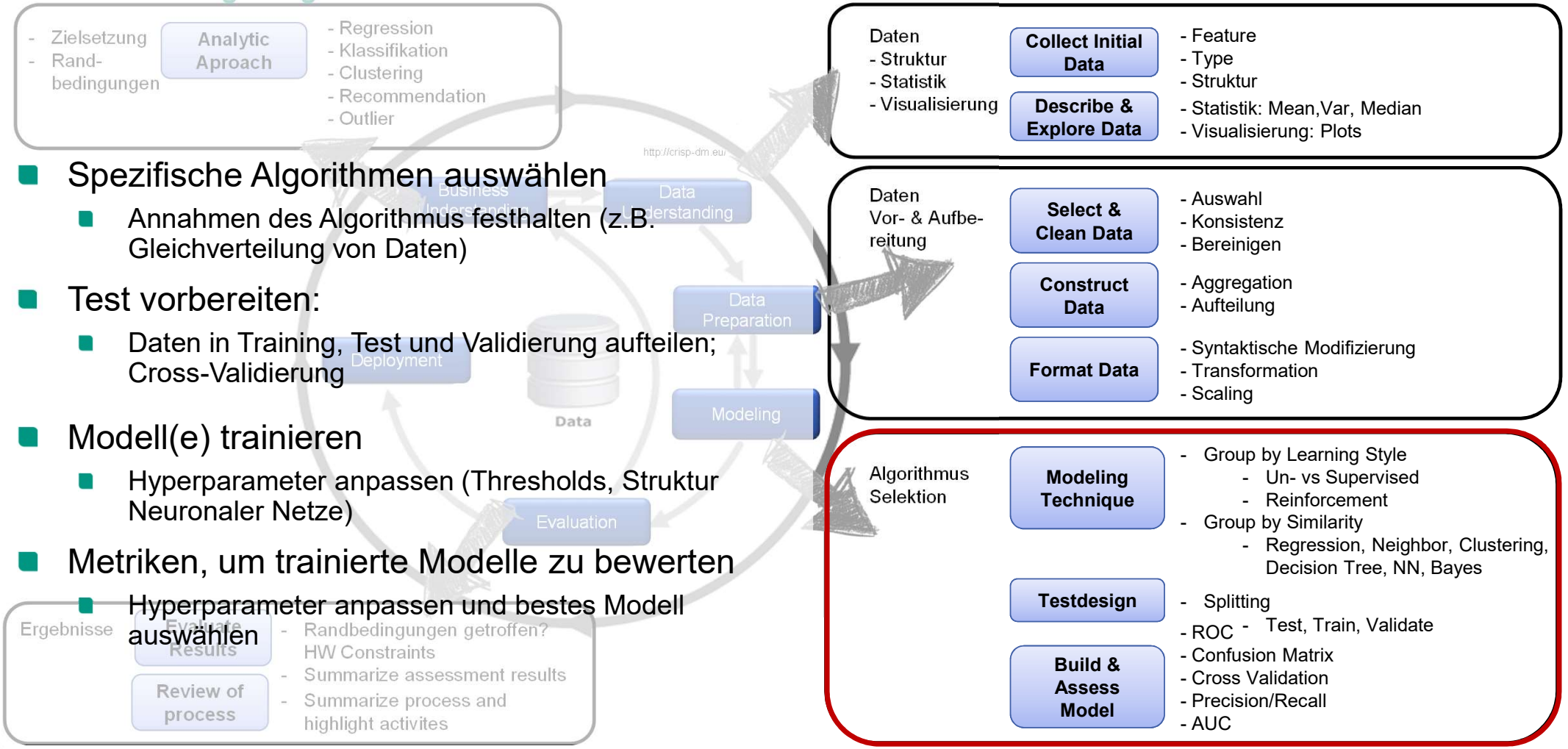
selbstprüfendes und selbstreparierendes System

Literatur

- Hilberg, Wolfgang (1995). Karl Steinbuch, ein zu Unrecht vergessener Pionier der künstlichen neuronalen Systeme
- Künstliches neuronales Netz, Online unter: http://de.wikipedia.org/wiki/Künstliches_neuronales_Netz. Stand: 8.7.2010
- Karl Steinbuch, Online unter: http://de.wikipedia.org/wiki/Karl_Steinbuch.
  Stand: 8.7.2010
- Karl Steinbuch, Online unter: http://xputers.informatik.uni-kl.de/papers/publications/karl-steinbuch.html.
  Stand: 8.7.2010
- http://www.cyranos.ch/doppel32.htm. Stand: 8.7.2010

CRISP-DM im Detail

Modellierung / Algorithmus Auswahl

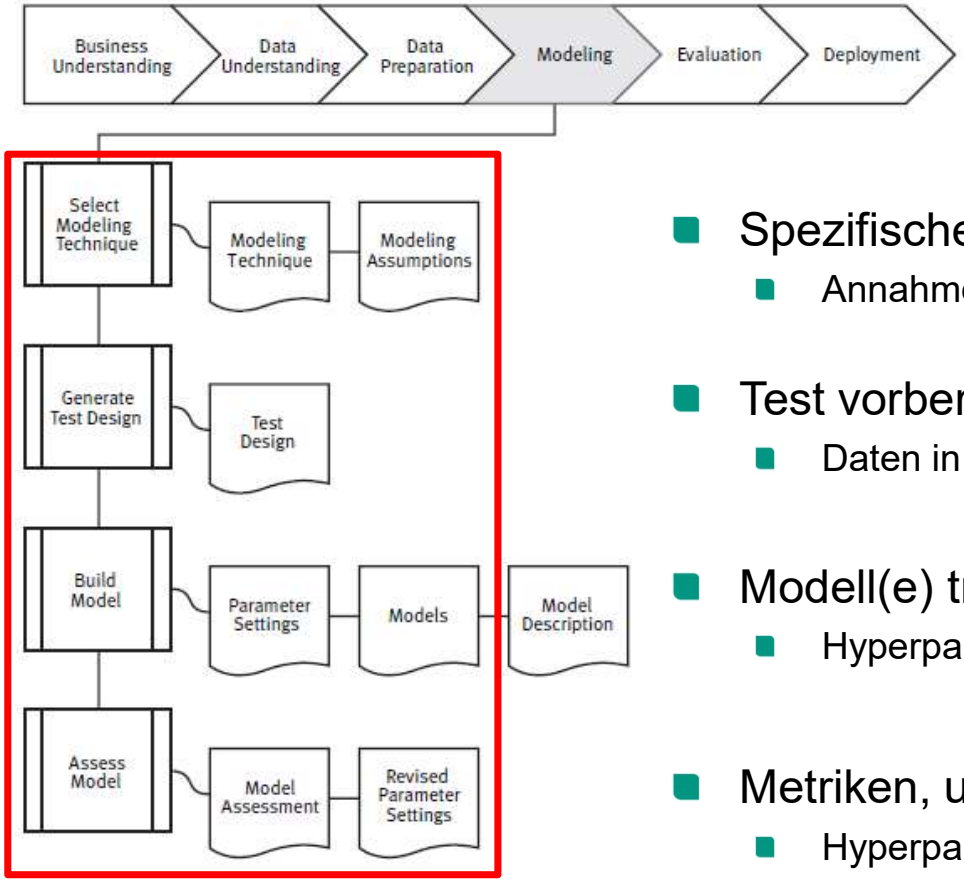


- Spezifische Algorithmen auswählen
  - Annahmen des Algorithmus festhalten (z.B. Gleichverteilung von Daten)
- Test vorbereiten:
  - Daten in Training, Test und Validierung aufteilen; Cross-Validierung
- Modell(e) trainieren
  - Hyperparameter anpassen (Thresholds, Struktur Neuronaler Netze)
- Metriken, um trainierte Modelle zu bewerten
  - Hyperparameter anpassen und bestes Modell auswählen

BN(8

CRISP-DM im Detail: Modeling

- Modeling = **A**lgorithmen-Auswahl, **A**ssessment, Parameter-**A**daption (**A³**)



- Spezifische Algorithmen auswählen
  - Annahmen des Algorithmus festhalten (z.B. Gleichverteilung von Daten)
- Test vorbereiten:
  - Daten in Training, Test und Validierung aufteilen; Cross-Validierung
- Modell(e) trainieren
  - Hyperparameter anpassen (Thresholds, Struktur Neuronaler Netze)
- Metriken, um trainierte Modelle zu bewerten
  - Hyperparameter anpassen und bestes Modell auswählen

Folie 2

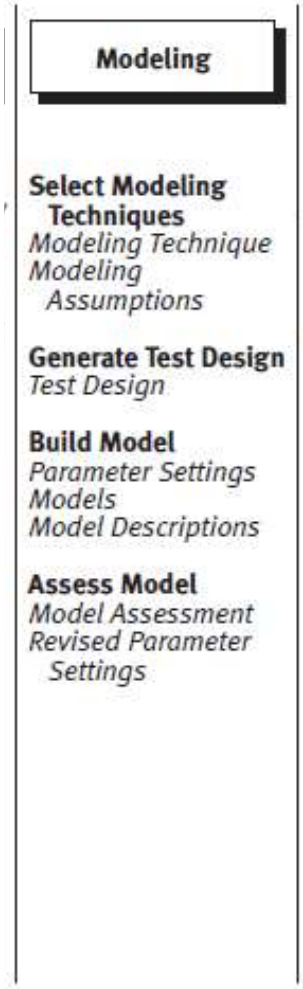
BN(8 alles markieren

Brenner, Nathalie (ITIV); 31.10.2018

BN(7

CRISP-DM im Detail: Data Modeling

- Auswahl der Modellierungsmethode
- Test vorbereiten
- Modell erstellen
- Modell bewerten

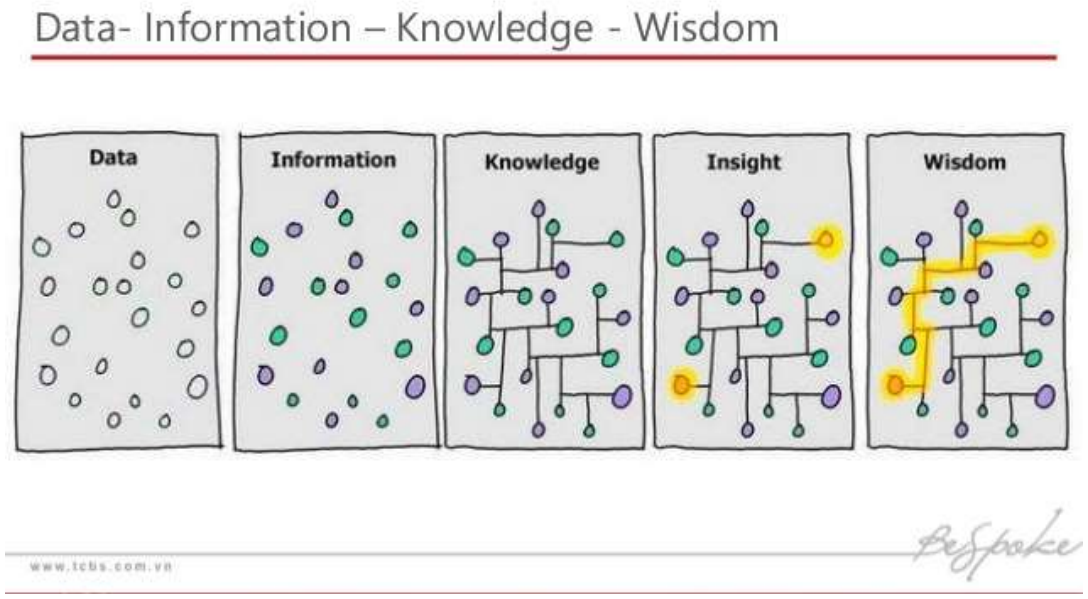


BN(7 Integration, PCA, noch weitere?

Brenner, Nathalie (ITIV); 31.10.2018

Worum geht es denn gleich nochmal?

- Klassisch geben wir Kriterien vor, die zu einer Klassifikation bzw. Ausführung führen.
- Intelligente Verfahren lernen aus Erfahrung, mit Belohnung, ohne dass konkrete Beschreibungen a priori vorliegen.



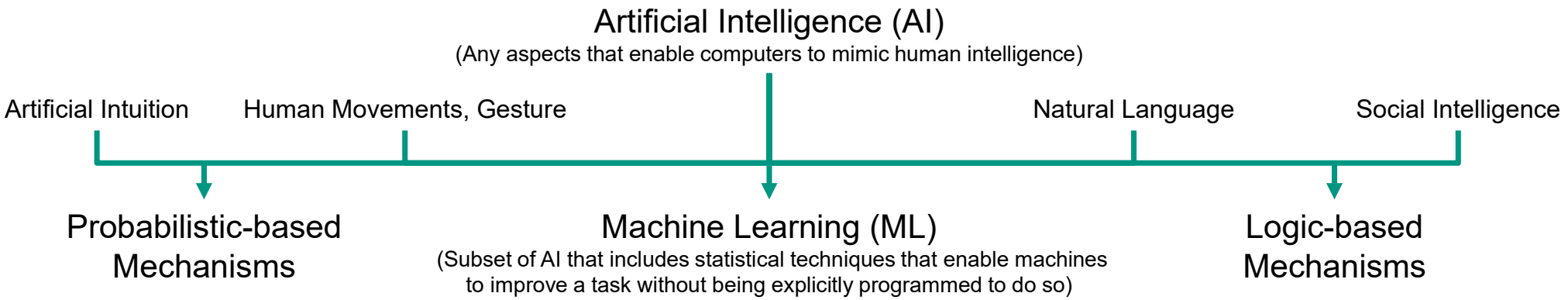


Worum geht es denn gleich nochmal?

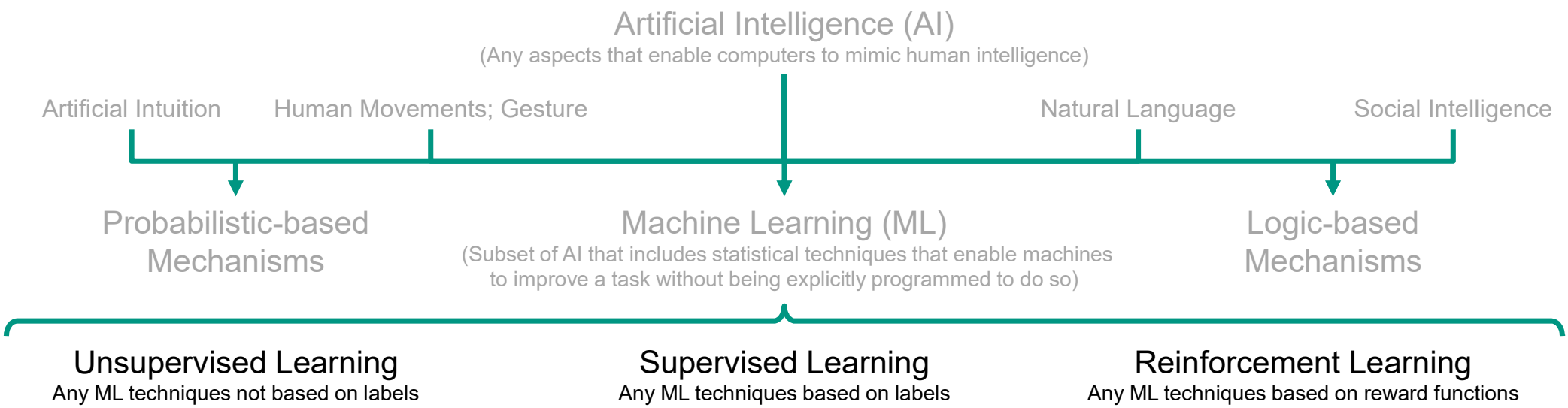
- Klassisch geben wir Kriterien vor, die zu einer Klassifikation bzw. Ausführung führen.
- Intelligente Verfahren lernen aus Erfahrung, mit Belohnung, ohne dass konkrete Beschreibungen a priori vorliegen.



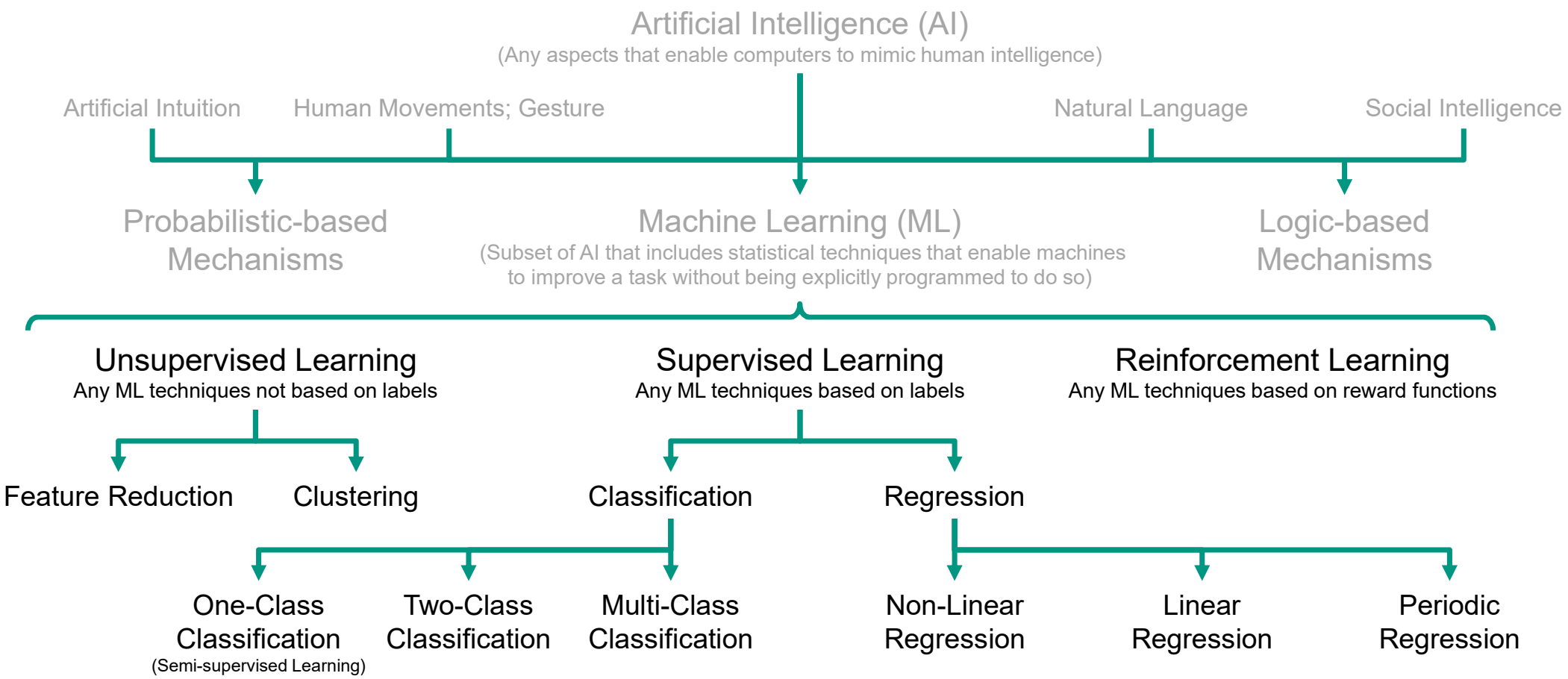
“Artificial Intelligence” und “Machine Learning”



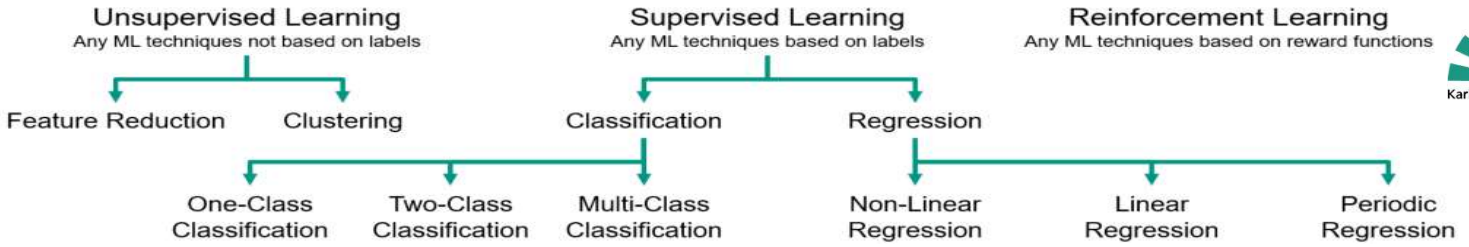
Lernmethoden



Spezielle Verfahren

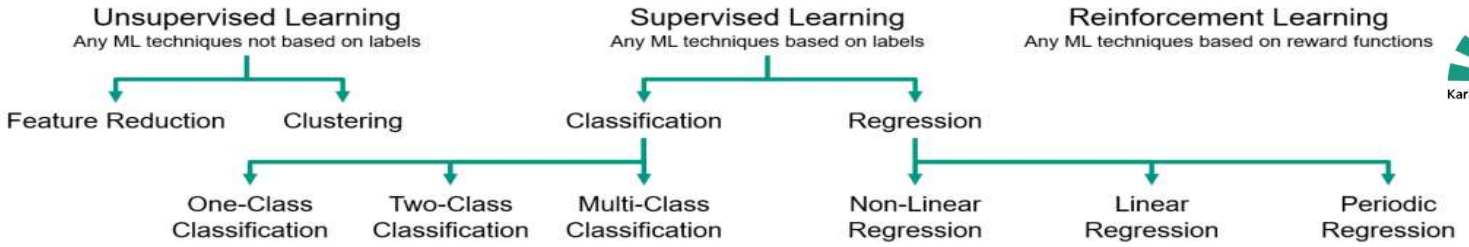


Machinelles Lernen



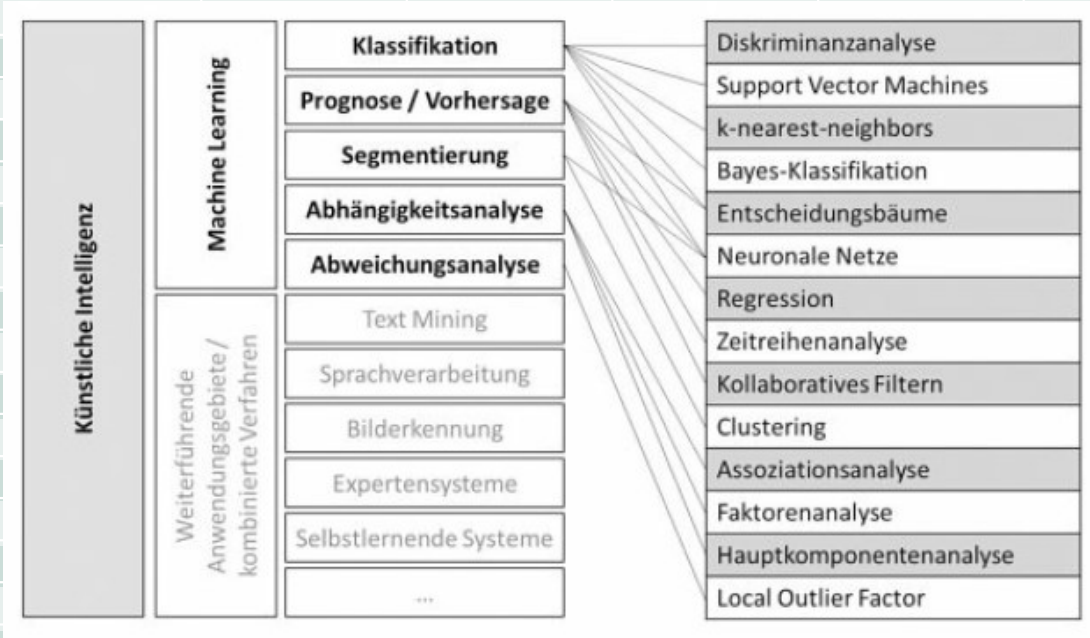
Algorithm ↓	Abk.	Feature Reduction	Clustering	One-Class Classification	Two-Class Classification	Multi-Class Classification	Non-Linear Regression	Linear Regression	Periodic Regression
Faktorenanalyse		X							
Principal Component Analysis	PCA	X		X					
	K-means		X						
hierarchical cluster analysis	HCA		X						
DBSCAN			X						
One Class Support Vector Machine	OCSVM			X					
Isolation Forest				X					
	LODA			X					
(künstliche) Neuronale Netze	NN			X (Autoencoder)	X	X	X	X	X
Support Vector Machine	SVM				X				
Decision Tree					X	X			
Bayes-Klassifikation					X	X			
Random Forest						X			
Diskriminanzanalyse				x	x	x			
Logistic Regression							X	X	
Linear Regression								X	
Harmonic Regression									X
Nächste-Nachbar-Klassifikation	K-NN	x	x						

Machinelles Lernen

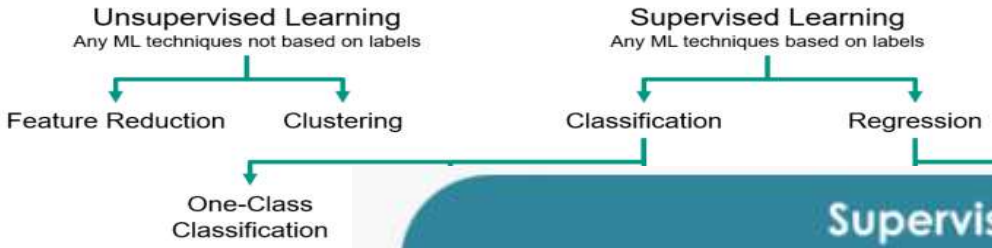


Algorithm	Abk.	Feature Reduction	Clustering	One-Class Classification	Two-Class Classification	Multi-Class Classification	Non-Linear Regression	Linear Regression	Periodic Regression
Faktorenanalyse		X							
Principal Component Analysis	PCA	X							
	K-means		X						
hierarchical cluster analysis	HCA		X						
DBSCAN			X						
One Class Support Vector Machine	OCSVM								
Isolation Forest									
	LODA								
(künstliche) Neuronale Netze	NN								X
Support Vector Machine	SVM								
Decision Tree									
Bayes-Klassifikation									
Random Forest									
Diskriminanzanalyse									
Logistic Regression									
Linear Regression								X	
Harmonic Regression									X
Nächste-Nachbar-Klassifikation	K-NN	x	x						

Andere Darstellungsformen

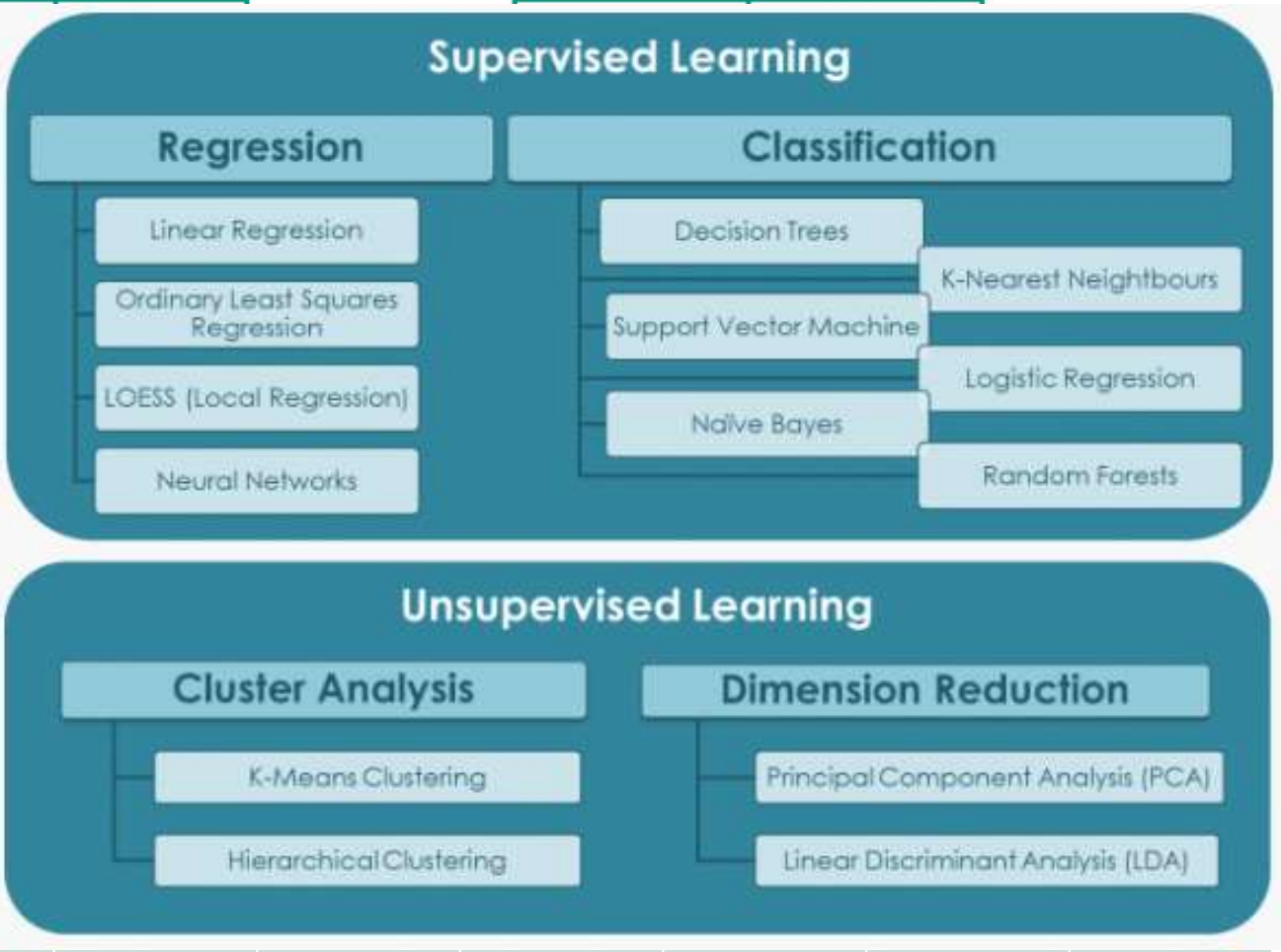


Machinelles Lernen

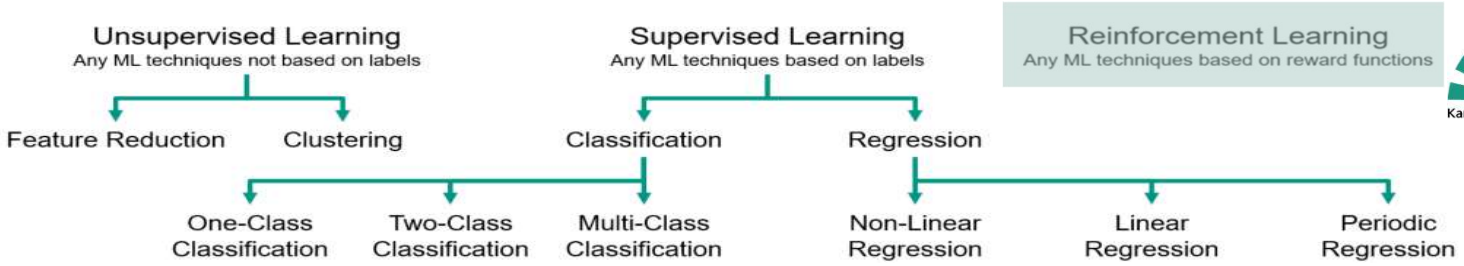


Algorithm	Abk.	Feature Reduction	Cluste
Faktorenanalyse		X	
Principal Component Analysis	PCA	X	
	K-means		X
hierarchical cluster analysis	HCA		X
DBSCAN			X
One Class Support Vector Machine	OCSVM		
Isolation Forest			
	LODA		
(künstliche) Neuronale Netze	NN		
Support Vector Machine	SVM		
Decision Tree			
Bayes-Klassifikation			
Random Forest			
Diskriminanzanalyse			
Logistic Regression			
Linear Regression			
Harmonic Regression			
Nächste-Nachbar-Klassifikation	K-NN	x	x

Andere Darstellungsformen



Machinelles Lernen



Algorithm ↓	Abk.	Feature Reduction	Clustering	One-Class Classification	Two-Class Classification	Multi-Class Classification	Non-Linear Regression	Linear Regression	Periodic Regression
Faktorenanalyse		X							
Principal Component Analysis	PCA	X		X					
	K-means		X						
hierarchical cluster analysis	HCA		X						
DBSCAN			X						
One Class Support Vector Machine	OCSVM			X					
Isolation Forest				X					
	LODA			X					
(künstliche) Neuronale Netze	NN			X (Autoencoder)	X	X	X	X	X
Support Vector Machine	SVM				X				
Decision Tree					X	X			
Bayes-Klassifikation					X	X			
Random Forest						X			
Diskriminanzanalyse				x	x	x			
Logistic Regression							X	X	
Linear Regression								X	
Harmonic Regression									X
Nächste-Nachbar-Klassifikation	K-NN	x	x						

Reinforcement Learning („Bestärkendes Lernen“)

Vorstufen

- Beim Teach-In-Verfahren (kurz: Teachen) fährt der Programmierer den Roboter mit einer Steuerkonsole in die gewünschte Position.
  - Alle so erreichten Koordinaten (Punkte) werden in der Steuerung gespeichert.
  - Dieser Schritt wird solange wiederholt, bis der gesamte Arbeitszyklus einmal durchlaufen ist.
  - Der Programmablauf besteht darin, dass der Roboter alle gespeicherten Punkte autonom anfährt.
  - Für die Bewegung zwischen den einzelnen Punkten können Parameter eingegeben werden.
  - So sind die Geschwindigkeit und die Beschleunigung einstellbar, bei einigen Robotern ist auch eine Angabe der notwendigen Genauigkeit möglich.



Reinforcement Learning
Vorstufen

- Man unterscheidet bei der Fahrt des Roboters grundsätzlich zwei verschiedene Varianten:
  - P2P: Point-to-Point: Der Roboter fährt von Punkt n nach Punkt n+1, wobei der geometrisch günstigste Weg gewählt wird (s. Folie zuvor).
    - ► Diese Methode ist schnell, führt aber zu einem unbekannten Pfad zwischen den Punkten. Diese Fahrt ist in aller Regel keine Gerade.
  - CP: Continuous Path: Hier wird der Roboter angewiesen, nicht nur von einem Punkt zum anderen zu fahren, sondern auch noch einem vorgegebenen Pfad zu folgen.
    - ► Diese Fahrt ist i. d. R. langsamer als P2P, dafür wird aber ein bestimmter Pfad nur innerhalb geringer Toleranzen verlassen.
    - ► Diese Methode eignet sich für komplexe Fügevorgänge, bei denen es weniger auf Geschwindigkeit als auf Genauigkeit (z. B. wegen Hindernissen) ankommt.

Reinforcement Learning
Vorstufen

- Man unterscheidet bei der Fahrt des Roboters grundsätzlich zwei verschiedene Varianten:
  - P2P: Point-to-Point: Der Roboter fährt von Punkt n nach Punkt n+1, wobei der geometrisch günstigste Weg gewählt wird (s. Folie zuvor).
    - ► Diese Methode ist schnell, führt aber zu einem unbekannten Pfad zwischen den Punkten. Diese Fahrt ist in aller Regel keine Gerade.
  - CP: Continuous Path: Hier wird der Roboter angewiesen, nicht nur von einem Punkt zum anderen zu fahren, sondern auch noch einem vorgegebenen Pfad zu folgen.
    - ► Diese Fahrt ist i. d. R. langsamer als P2P, dafür wird aber ein bestimmter Pfad nur innerhalb geringer Toleranzen verlassen.
    - ► Diese Methode eignet sich für komplexe Fügevorgänge, bei denen es weniger auf Geschwindigkeit als auf Genauigkeit (z. B. wegen Hindernissen) ankommt.
- Playback-Verfahren
  - Der Programmierer fährt durch direktes Führen des Roboterarms die vorgesehene Bahn ab.
    - ► Der Roboter wiederholt genau diese Bewegungen.
    - ► Diese Methode wird häufig bei Lackierrobotern eingesetzt.
  - Playback-Verfahren mit 3D-Messarmen (mobile Koordinatenmeßsysteme KMG).
    - ► Während des Führens des KMG entlang der Bauteilkontur werden Koordinaten aufgezeichnet und später in ein Roboterprogramm konvertiert.

MiWuLa …

- „Automatisierter Schiffsverkehr“ …
  - … findet ohne menschliche Interaktion statt
  - … klinkt sich in die Automatisierungsphilosophie des MiWuLa ein
- Ausblick
  - Schiffsignale passend zum Manöver
  - Mehrere Schiffe mit Kolli
    - Über „Karte“ oder „on b
  - „Längsseits gehen“ von Polizeiboot, Lotse, Proviar



Miniatur Wunderland
www.miniatur-wunderland.de

Reinforcement Learning

„Bestärkendes“ Lernen



- Ein lernendes System (Agent) kann die Umgebung beobachten, Aktionen auswählen und durchführen und dafür Belohnungen erhalten (Reward).
  - Der Agent wählt situativ eine Aktion aus einer Menge von verfügbaren Aktionen, wodurch er in einen Folgezustand gelangt und eine Belohnung erhält.
  - Ziel des Agenten ist es somit, den zukünftig erwarteten Gewinn zu maximieren.
- Es muss dann von selbst lernen, was die beste Strategie, eine so genannte Politik (Policy) ist, um im Laufe der Zeit die meiste Belohnung zu erhalten.
- Eine Richtlinie definiert, welche Aktion der Agent wählen soll, wenn er sich in einer bestimmten Situation befindet.

- Remark: *Unterschied Zustand - Situation*
  - *Ein Zustand eines Systems ist die Abspeicherung aller Variablen, die das System hat.*
    - *Aus diesen Variablen lässt sich der Zustand komplett rekonstruieren.*
  - *Bei den meisten Systemen kommen wir jedoch an alle Variablen gar nicht ran, weshalb die unvollständige Variablenmenge nur eine Situation ist.*

Reinforcement Learning

„Bestärkendes“ Lernen

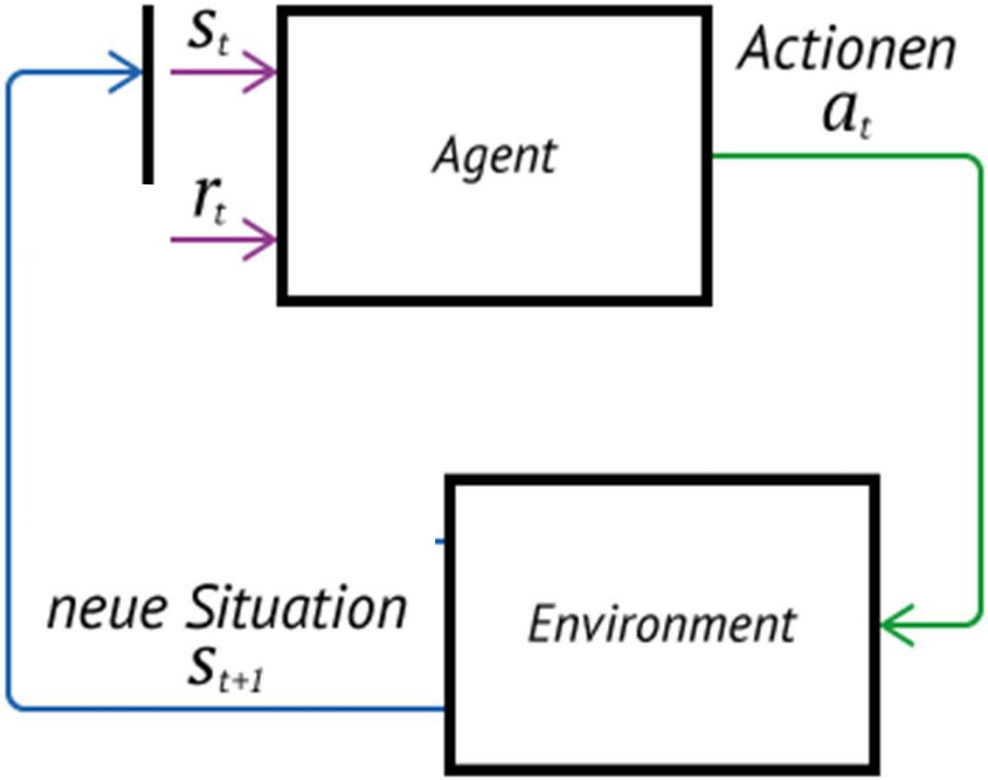


- Reinforcement Learning wird auf einen Agenten angewandt, der auf ein dynamisches Environment (System) Einfluss ausübt und in diesem Environment bestimmte Aufgaben ausüben muss.

Quelle: http://www.informatikseite.de/neuro/node67.php

Reinforcement Learning

„Bestärkendes“ Lernen

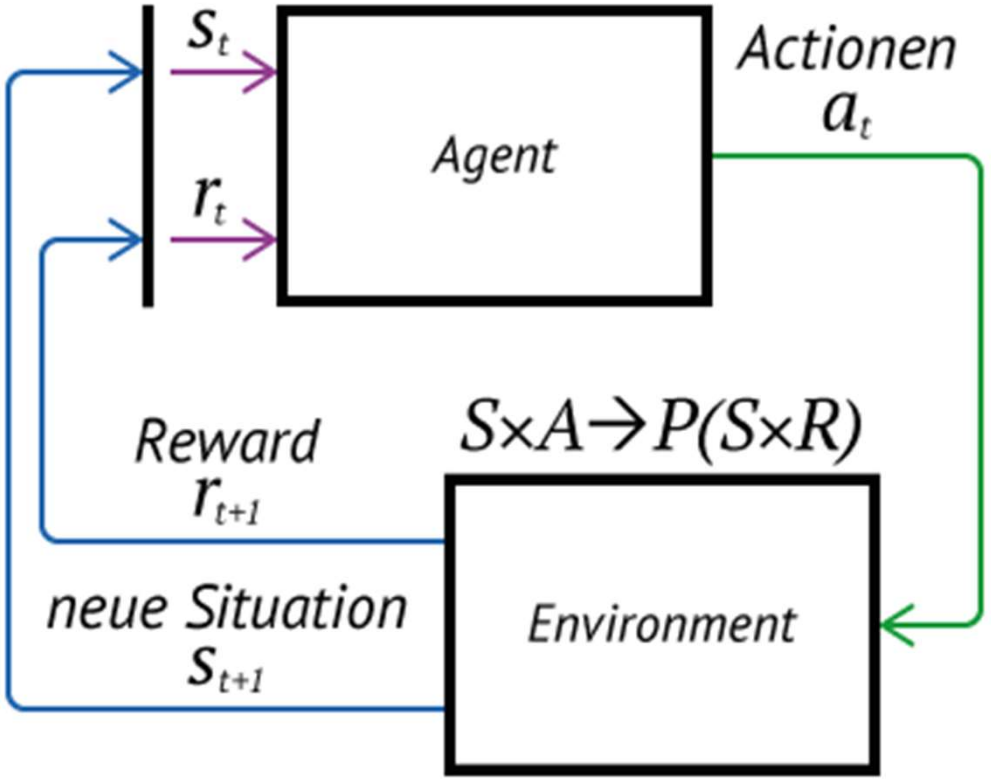


- Reinforcement Learning wird auf einen Agenten angewandt, der auf ein dynamisches Environment (System) Einfluss ausübt und in diesem Environment bestimmte Aufgaben ausüben muss.
- Der Agent bildet also die momentane Situation s $\in$ S des Environments auf eine Aktion a $\in$ A ab.

Quelle: http://www.informatikseite.de/neuro/node67.php

Reinforcement Learning

„Bestärkendes“ Lernen



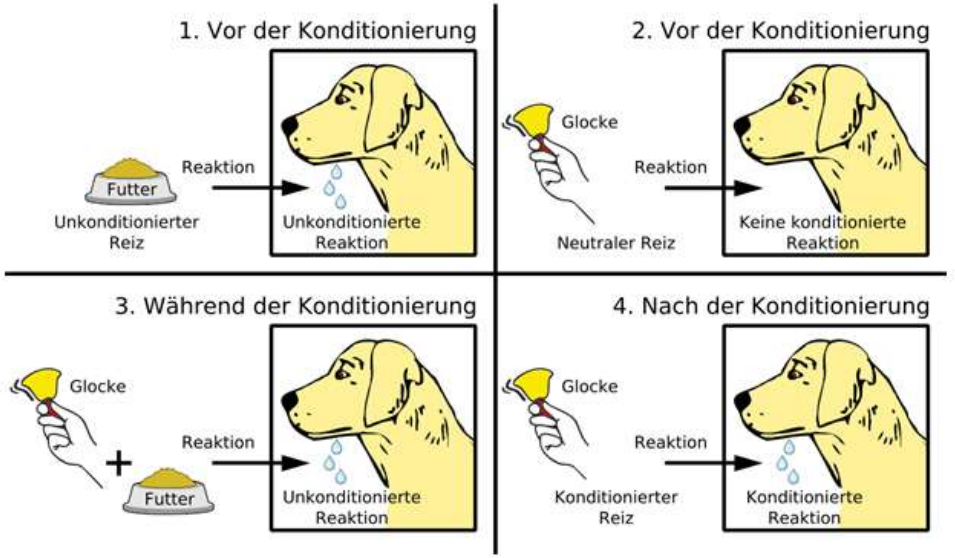
- Reinforcement Learning wird auf einen Agenten angewandt, der auf ein dynamisches Environment (System) Einfluss ausübt und in diesem Environment bestimmte Aufgaben ausüben muss.
- Der Agent bildet also die momentane Situation s $\in$ S des Environments auf eine Aktion a $\in$ A ab.
- Für jede Aktion bekommt der Agent einen Reward r $\in$ R, der seine Aktion bewertet, aber nicht sagt, was er falsch gemacht hat oder *was* er gut gemacht hat.
- Beim Reinforcement Learning ist es das Ziel einen möglichst hohen Reward über eine lange Zeit zu erwirtschaften.

Quelle: http://www.informatikseite.de/neuro/node67.php

Lernen

Bedingter Reflex nach Pawlow

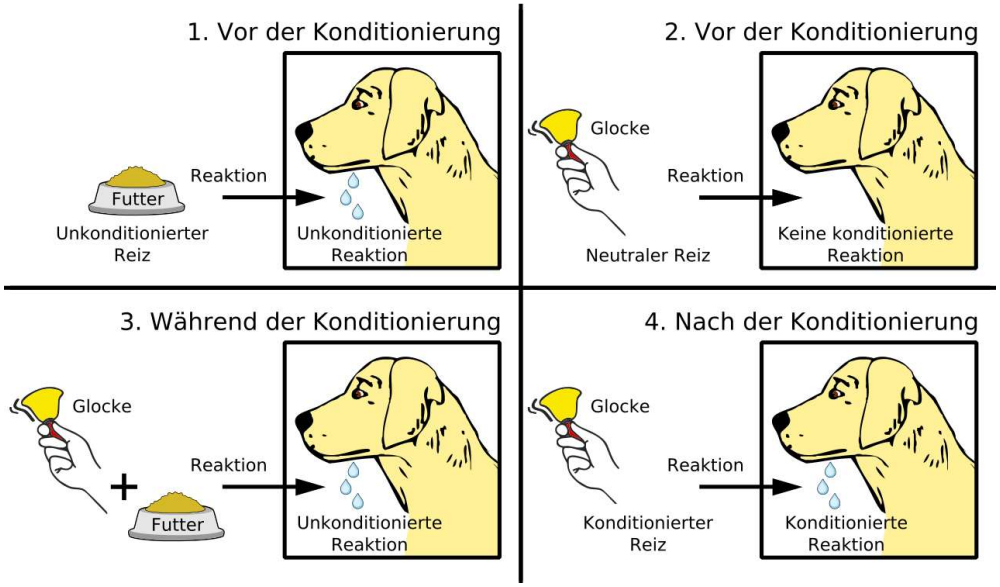
- Um diese Hypothese zu prüfen, gestaltete er 1905[1] ein aussagekräftiges Experiment:
  - Auf die Darbietung von Futter, einem unbedingten Reiz, folgt Speichelfluss (unbedingte Reaktion), auf das Ertönen eines Glockentons (neutraler Reiz) nichts.
  - Wenn aber der Glockenton wiederholt in engem zeitlichem Zusammenhang mit dem Anbieten von Futter erklingt, reagieren die Hunde schließlich auf den Ton allein mit Speichelfluss.
  - Dieses Phänomen bezeichnete Pawlow als Konditionierung.



- **Klassische Konditionierung** ist eine von dem russischen Physiologen Iwan Petrowitsch Pawlow begründete behavioristische Lerntheorie, die besagt, dass einer natürlichen, meist angeborenen, sogenannten *unbedingten* Reaktion durch Lernen eine neue, *bedingte* Reaktion hinzugefügt werden kann.

Reinforcement Learning
„Konditionierung“

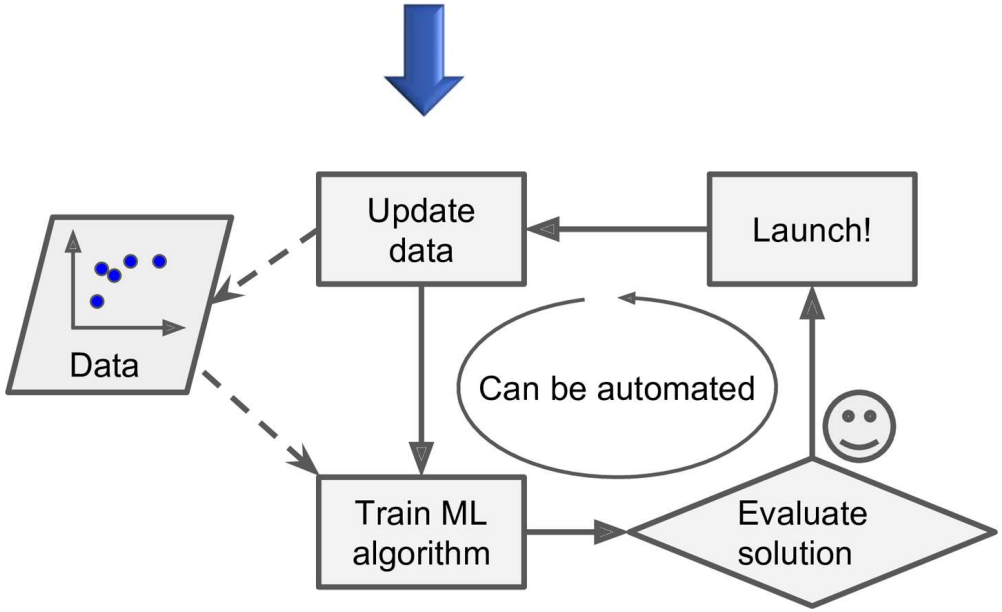
- Die Begriff Reinforcement wurde im Jahre 1903 von Pavlov eingeführt, um die Stärkung der Assoziation zwischen einer unbedingten und einen konditionierten Reiz, der sich ergibt, wenn die beiden zusammen dargestellt werden, zu beschreiben.
- Bei der Konditionierung bezeichnet man ein Ereignis, das die Wahrscheinlichkeit erhöht, dass ein bestimmtes Verhalten gezeigt wird, als Verstärkung.



- Klassische Konditionierung ist eine von dem russischen Physiologen Iwan Petrowitsch Pawlow begründete behavioristische Lerntheorie, die besagt, dass einer natürlichen, meist angeborenen, sogenannten *unbedingten* Reaktion durch Lernen eine neue, *bedingte* Reaktion hinzugefügt werden kann.

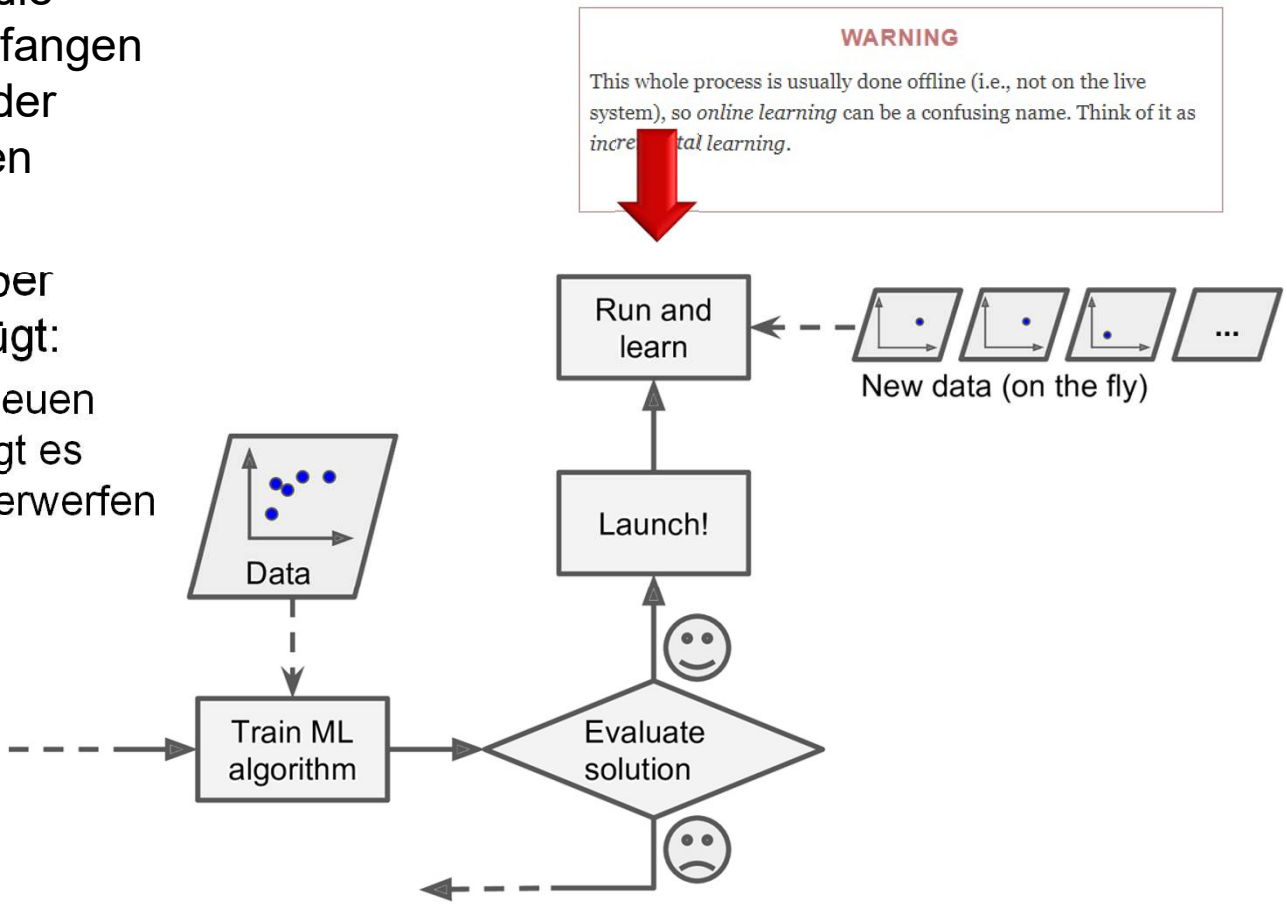
Batch (offline) Learning

- Beim Batch-Lernen ist das System nicht inkrementell lernfähig:
  - Es muss mit allen verfügbaren Daten trainiert werden.
  - Dies nimmt in der Regel viel Zeit und Rechenressourcen in Anspruch, so dass dies offline geschieht.
  - Zuerst wird das System trainiert, dann wird es in die Praxis eingeführt und läuft ohne Lernen weiter.
  - Es wendet an, was es gelernt hat.
  - Dies wird als Offline-Lernen bezeichnet.
- Wenn man möchte, dass ein Batch-Lernsystem über neue Daten (z.B. eine neue Art von Spam) informiert wird, muss man eine neue Version des Systems von Grund auf den gesamten Datensatz trainieren (nicht nur die neuen Daten, sondern auch die alten Daten).



Incremental (online) Learning

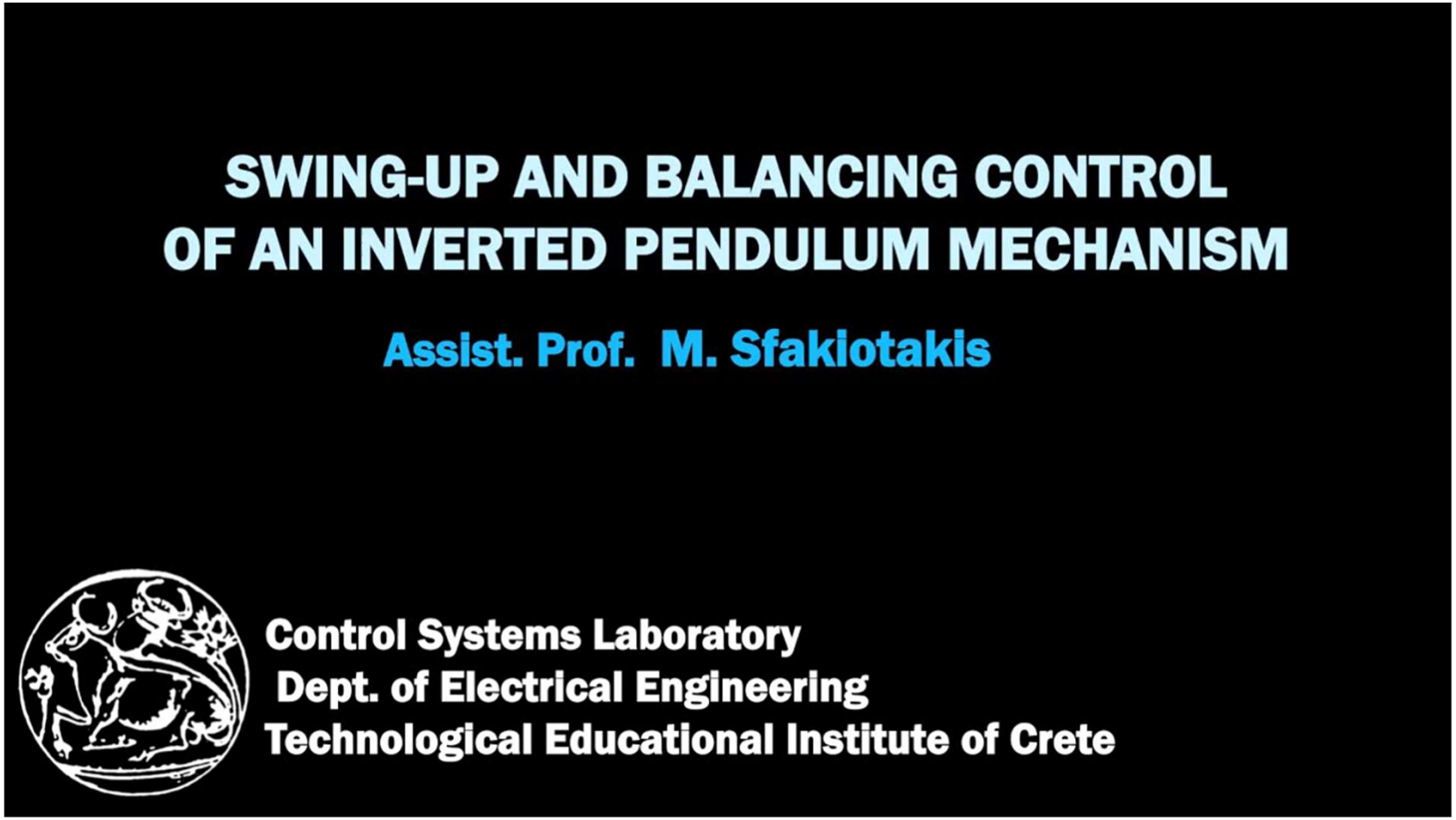
- Online-Lernen ist ideal für Systeme, die Daten als kontinuierlichen Fluss empfangen (z.B. Aktienkurse) und sich schnell oder autonom an Veränderungen anpassen müssen.
- Es ist eine gute Option, wenn man über begrenzte Computerressourcen verfügt:
  - Sobald ein Online-Lernsystem von neuen Dateninstanzen erfahren hat, benötigt es diese nicht mehr, so dass man sie verwerfen kann



„Polebalancer“

Ziel das Pole Balancers ist es, den Stab gerade zu halten.

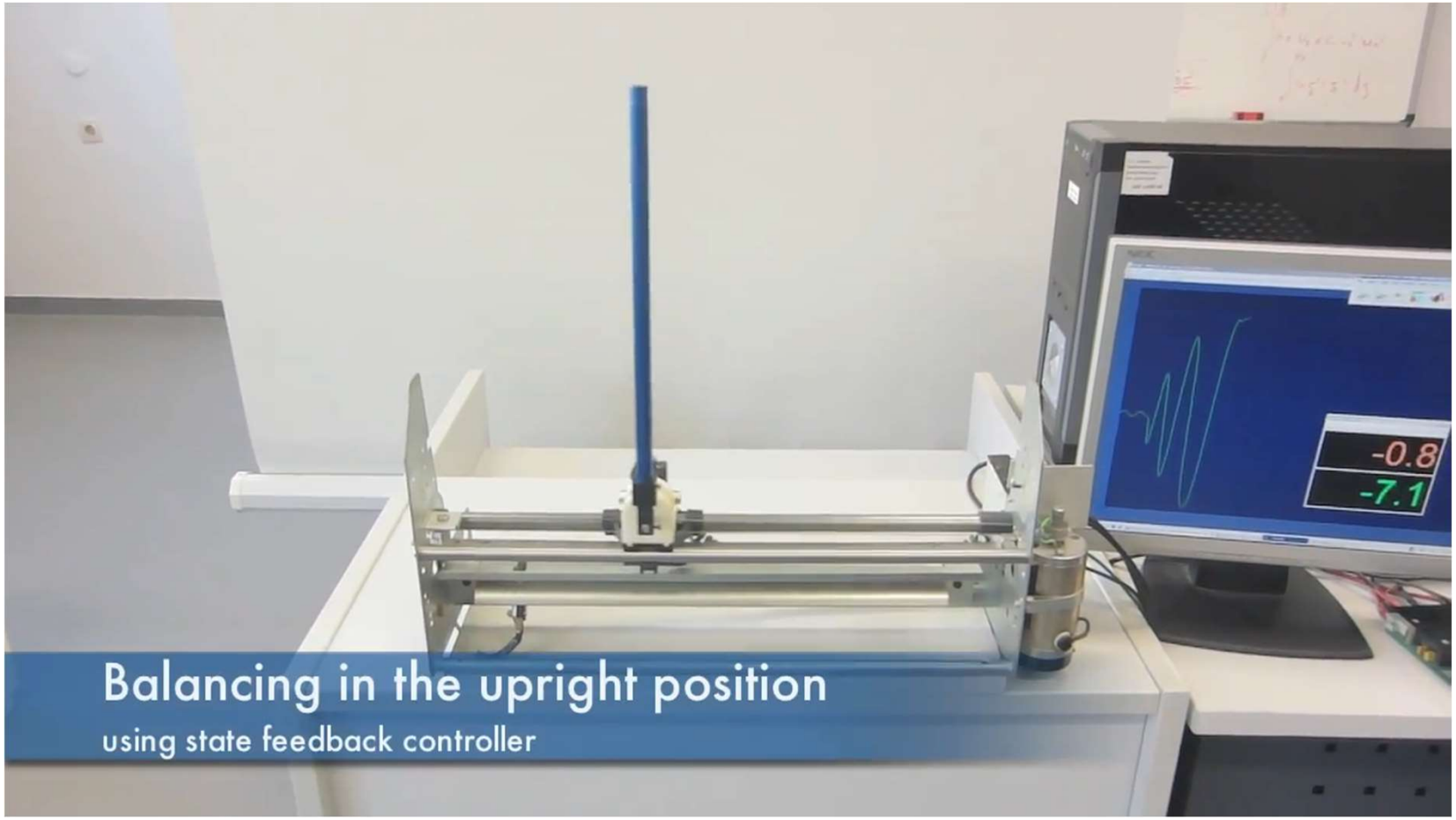
Reinforcment Learning mit einem „pure negativ reward“



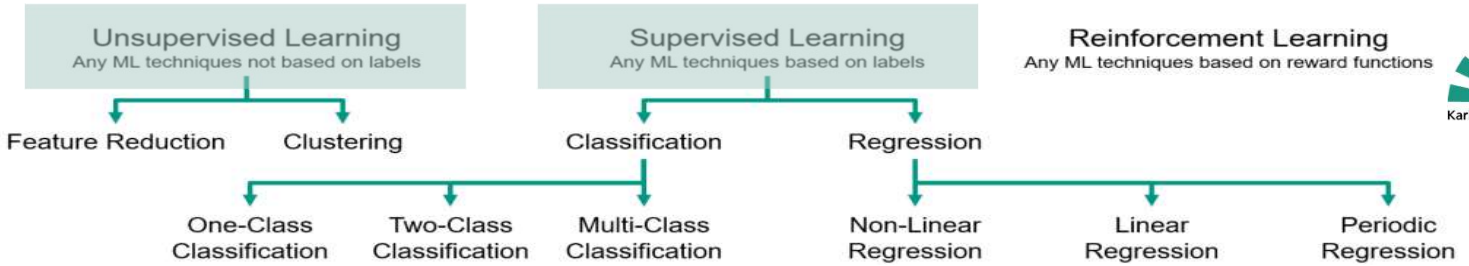
„Polebalancer“

Ziel das Pole Balancers ist es, den Stab gerade zu halten.

Reinforcment Learning mit einem „pure negativ reward“



Machinelles Lernen



Algorithm ↓	Abk.	Feature Reduction	Clustering	One-Class Classification	Two-Class Classification	Multi-Class Classification	Non-Linear Regression	Linear Regression	Periodic Regression
Faktorenanalyse		X							
Principal Component Analysis	PCA	X		X					
	K-means		X						
hierarchical cluster analysis	HCA		X						
DBSCAN			X						
One Class Support Vector Machine	OCSVM			X					
Isolation Forest				X					
	LODA			X					
(künstliche) Neuronale Netze	NN			X (Autoencoder)	X	X	X	X	X
Support Vector Machine	SVM				X				
Decision Tree					X	X			
Bayes-Klassifikation					X	X			
Random Forest						X			
Diskriminanzanalyse				x	x	x			
Logistic Regression							X	X	
Linear Regression								X	
Harmonic Regression									X
Nächste-Nachbar-Klassifikation	K-NN	x	x						

Eine Anmerkung zu den Begriffen

Überwachtes und unüberwachtes Lernen

- Die Begriffe »überwacht« und »unüberwacht«
  - Im übertragenen Sinn »überwacht« ein Lehrer den Lernenden, indem er ihm vorsichtige Hinweise auf das Lernziel sowie einige Beispiele gibt.
  - Beim »unüberwachten« Lernen könnten die selben Beispiele vorhanden sein, es gibt jedoch keine Hinweise auf das Lernziel.
- Der Lernende wird nicht über den Zweck des Lernens informiert, sondern muss seine eigenen Schlüsse ziehen, was die Beispiele gemeinsam haben (eine Art Autodidakt)

- Der Unterschied zwischen den beiden Fragen ist zwar nicht groß, aber bedeutsam.
  - Wenn ein Ziel angegeben ist, kann die Aufgabe als überwachter Task formuliert werden.
  - Überwachte Tasks erfordern andere Verfahren als unüberwachte und liefern oft sehr viel nützlichere Ergebnisse.
  - Einem überwachten Verfahren wird ein bestimmter Zweck der Gruppierung vorgegeben – die Vorhersage des Ziels.

Überwachtes vs. Unüberwachtes Lernen

- Beim überwachten Lernen liegen Daten vor, die schon ein Ergebnis oder „Label“ enthalten.
- Beim überwachten Lernen wird die vom Netz erzeugte Ausgabe betrachtet und ihre Abweichung von der gewünschten Ausgabe gemessen.
  - Danach werden die Gewichte entsprechend der Größe der Abweichung angepasst.
- Remark: *Das bestärkende Lernen (Reinforcement Learning) kann als überwachtes Lernen ohne Informationen über die Höhe der Abweichung vom gewünschten Wert interpretiert werden.*

- Das unüberwachte Lernen setzt keinen Lehrer voraus, d. h. für die Eingabewerte der Trainingsdaten ist die richtige Ausgabe nicht bekannt.
- Das Netz muss sich selber organisieren, d. h. die Gewichtungen anpassen, um Muster zu erkennen und z.B. Gruppierungen (Cluster) der Datensätze vorzunehmen.
- Beim unüberwachten Lernen handelt es sich um das allgemeine Verstehen der vorliegenden Daten und die Entdeckung versteckter Strukturen.
  - ► Die Daten sind also nicht durch „Labels“ gekennzeichnet.

Überwachtes vs. Unüberwachtes Lernen

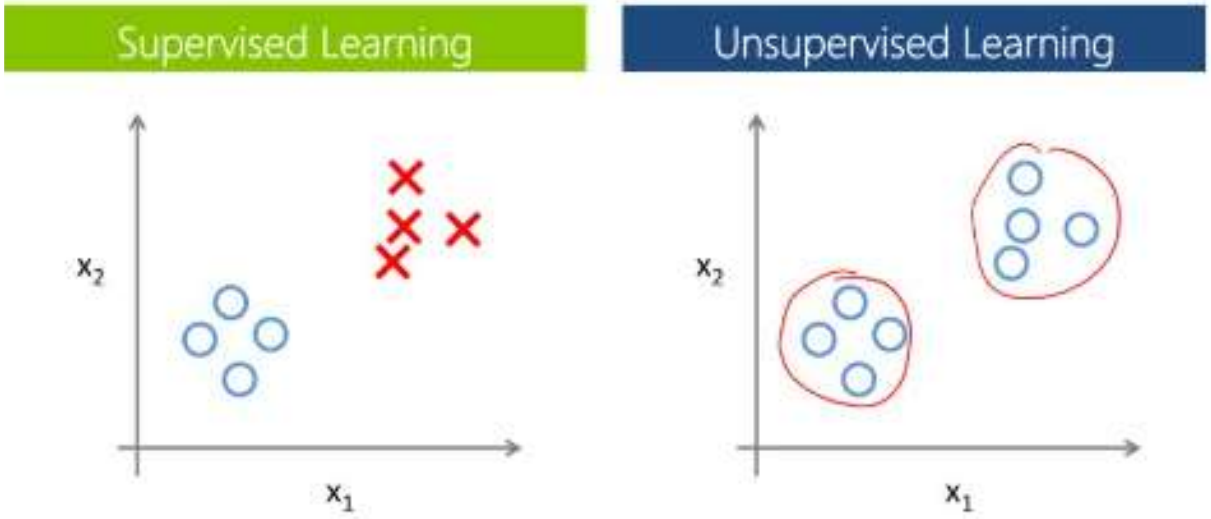
- Beim überwachten Lernen liegen Daten vor, die schon ein Ergebnis oder „Label“ enthalten.
- Beim überwachten Lernen wird die vom Netz erzeugte Ausgabe betrachtet und ihre Abweichung von der gewünschten Ausgabe gemessen.
  - Danach werden die Gewichte entsprechend der Größe der Abweichung angepasst.
- Remark: *Das bestärkende Lernen (Reinforcement Learning) kann als überwachtes Lernen ohne Informationen über die Höhe der Abweichung vom gewünschten Wert interpretiert werden.*

- Das unüberwachte Lernen setzt keinen Lehrer voraus, d. h. für die Eingabewerte der Trainingsdaten ist die richtige Ausgabe nicht bekannt.
- Das Netz muss sich selber organisieren, d. h. die Gewichtungen anpassen, um Muster zu erkennen und z.B. Gruppierungen (Cluster) der Datensätze vorzunehmen.
- Beim unüberwachten Lernen handelt es sich um das allgemeine Verstehen der vorliegenden Daten und die Entdeckung versteckter Strukturen.
  - ► Die Daten sind also nicht durch „Labels“ gekennzeichnet.

- Die Frage nach überwachtem oder unüberwachtem Lernen ist also weniger „dramatisch“ als es die Begriffe vermuten lassen.
- Es geht nicht um eine echte „Überwachung“ des Lernprozesses, sondern lediglich darum, ob die Daten ge-labelt sind.

Überwachtes vs. Unüberwachtes Lernen

	Deutsch	Englisch	Beschreibung
Teacher	Überwachtes Lernen	Supervised	Es gibt einen desired Output, welchen das Netz lernen soll.
Reward	Bestärkendes Lernen	Reinforcment Learning	Es gibt nur eine Belohnung, die sagt, ob die Aktion gut oder schlecht war.
-	Unüberwachtes Lernen	Unsupervised	Es gibt gar keinen Output mehr. Wir können die Muster nur zu Clustern zusammenfassen.



Überwachtes vs. Unüberwachtes Lernen

	Deutsch	Englisch	Beschreibung
Teacher	Überwachtes Lernen	Supervised	Es gibt einen desired Output, welchen das Netz lernen soll.
Reward	Bestärkendes Lernen	Reinforcment Learning	Es gibt nur eine Belohnung, die sagt, ob die Aktion gut oder schlecht war.
-	Unüberwachtes Lernen	Unsupervised	Es gibt gar keinen Output mehr. Wir können die Muster nur zu Clustern zusammenfassen.

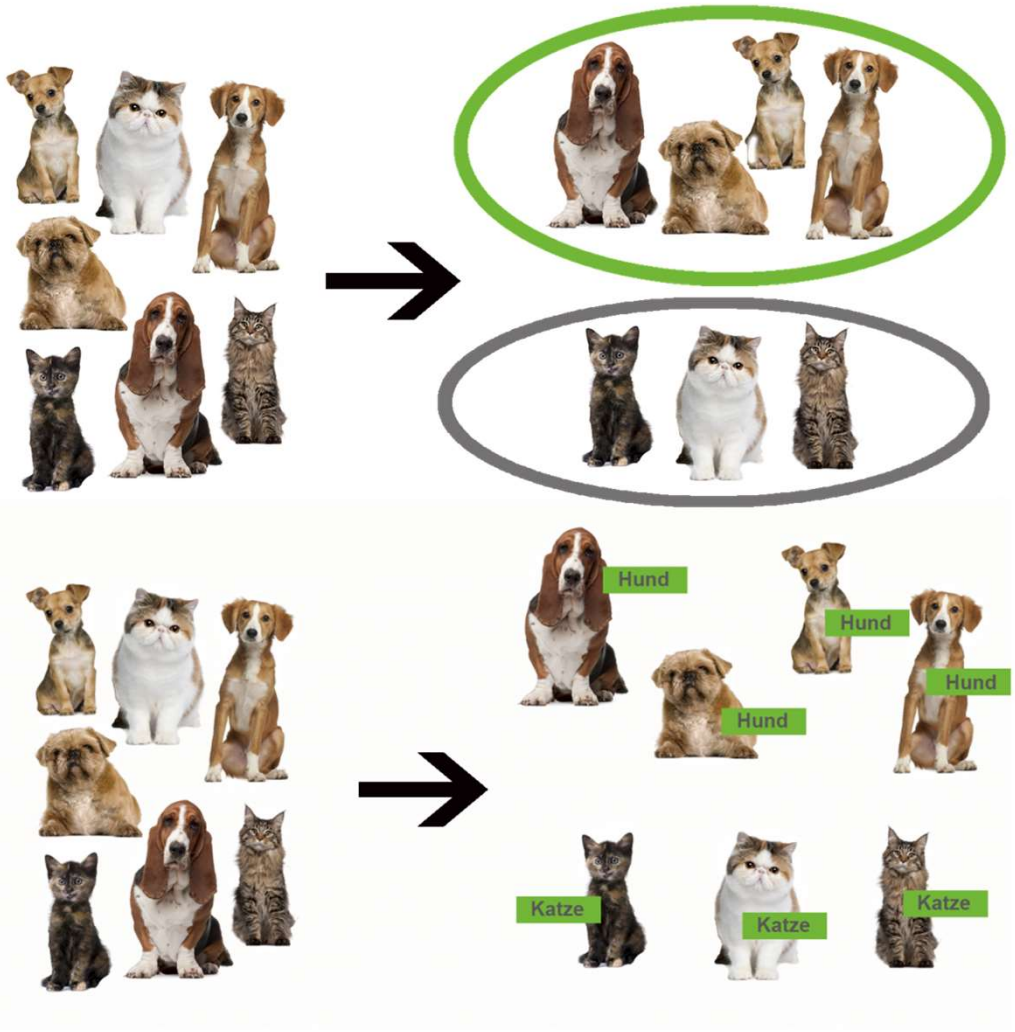




Überwachtes vs. Unüberwachtes Lernen

Beispiel: Hunde und Katzen

- Unüberwachtes Lernen:
  - Das System weiß nicht, was es erkennen soll.
  - Wenn es Bilder von Tieren verarbeitet, teilt das System alles was aussieht wie eine Katze oder alles was aussieht wie ein Hund in entsprechende Gruppen ein, ohne diese jedoch so zu benennen, da nicht definiert ist, was eine Katze und was ein Hund ist.
  - Diese Methode wird angewandt, wenn Daten noch unbekannt sind und entsprechend keine Vorgaben vorhanden sind.
- Überwachtes Lernen:
  - Es gibt Trainings-Daten, bei denen wir die Eingangs-Parameter sowie das Ergebnis kennen.
  - Aus den Trainings-Daten werden Modelle von Hunden und Katzen erstellt, die zusammen mit den Machine Learning Algorithmen das Ergebnis liefern.
  - Da die Modelle erstellt wurden, kann man „unbekannte“ Bilder liefern und das System berechnet dafür das Ergebnis (Prediction).



Überwachtes vs unüberwachtes maschinelles Lernen

- Während überwachte Lernverfahren über eine Trainingsphase regelrecht auf ein(!) Problem abgerichtet werden und dann produktiv als Assistenzsystem (bis hin zum Automated Decision Making) funktionieren, sind unüberwachte maschinelle Lernverfahren eine Methodik, um unübersichtlich viele Zeilen und Spalten von folglich sehr großen Datenbeständen für den Menschen leichter interpretierbar machen zu können.

- Trainiere einen Algorithmus mit überwachtem maschinellen Lernen
  - Wenn ein Modell anhand von mit dem Ergebnis (z. B. Klassifikationsgruppe) gekennzeichneter Trainingsdaten erlernt werden soll, handelt es sich um überwachtes Lernen.
  - Die richtige Antwort muss während der Trainingsphase also vorliegen und der Algorithmus muss die Lücke zwischen dem Input (Eingabewerte) und dem Output (das vorgeschriebene Ergebnis) füllen.
  - Die Überwachung bezieht sich dabei nur auf die Trainingsdaten!
  - Im produktiven Lauf wird grundsätzlich nicht überwacht
    ► Das Lernen könnte sich auf neue Daten in eine ganz andere Richtung entwickeln, als dies mit den Trainingsdaten der Fall war.

„Clusterbildung“ als Repräsentant des unüberwachten Lernens

Clusterbildung hat das Ziel nicht annotierte Trainingsdaten zu gruppieren.

- Im Unterschied zur Klassifikation ist damit die Zuordnung der Trainingsdaten zu den gesuchten Klassen nicht vorhanden.
- Die Cluster, in welche die Eingangswerte gruppiert werden, sind zu Beginn unbekannt.
- Durch die nicht annotierten Trainingsdaten ist Clusterbildung eine Problemstellung für unüberwachtes Lernen und ein wichtiger Bestandteil bei der Wissensentdeckung in großen Datenmengen.

*OBD = Onboard Diagnose

„Clusterbildung“ als Repräsentant des unüberwachten Lernens

Clusterbildung hat das Ziel nicht annotierte Trainingsdaten zu gruppieren.

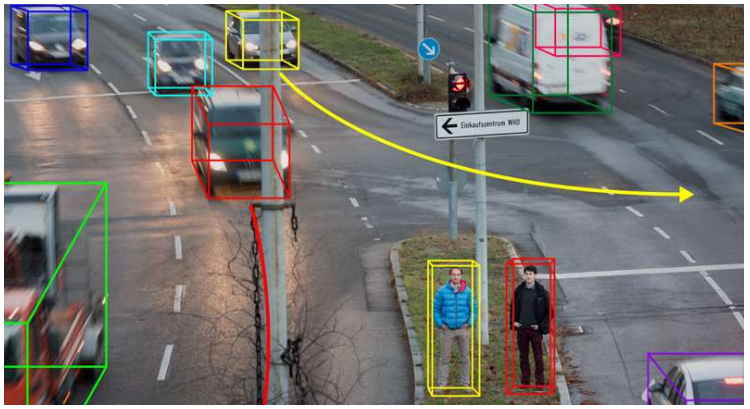
- Im Unterschied zur Klassifikation ist damit die Zuordnung der Trainingsdaten zu den gesuchten Klassen nicht vorhanden.
- Die Cluster, in welche die Eingangswerte gruppiert werden, sind zu Beginn unbekannt.
- Durch die nicht annotierten Trainingsdaten ist Clusterbildung eine Problemstellung für unüberwachtes Lernen und ein wichtiger Bestandteil bei der Wissensentdeckung in großen Datenmengen.



*OBD = Onboard Diagnose

„Klassifikation“ als Repräsentant des überwachten Lernens

- Bei der Klassifikation ist die Aufgabe den gegebenen Eingangswerten einen bestimmten diskreten Ausgangswert zuzuordnen.
  - ► Damit findet eine Klassifizierung der Eingangswerte statt.
- Entsprechende Aufgaben werden meistens durch überwachte Lernverfahren adressiert.
  - ► Mittels annotierter Trainingsdaten wird ein Klassifikator gelernt, der die Eingangswerte möglichst präzise in die vorgegebenen Klassen einordnet.
  - ► Ist das Training erfolgreich abgeschlossen, klassifiziert das Computerprogramm in der Testphase auch neue, bis dahin unbekannte Datensätze richtig.
- Ein bekanntes Beispiel ist die Objekterkennung in Bilddaten und die Zuordnung eines gefundenen Objekts zu einer bestimmten Klasse.
  - ► So müssen in modernen Fahrzeugen Objekte in Echtzeit erkannt und klassifiziert werden, beispielsweise Fahrzeuge, Verkehrsschilder, Fußgänger, Hindernisse …
  - ► Die Objekterkennung wird dafür mittels ausgesuchter, annotierter Datensätze trainiert.



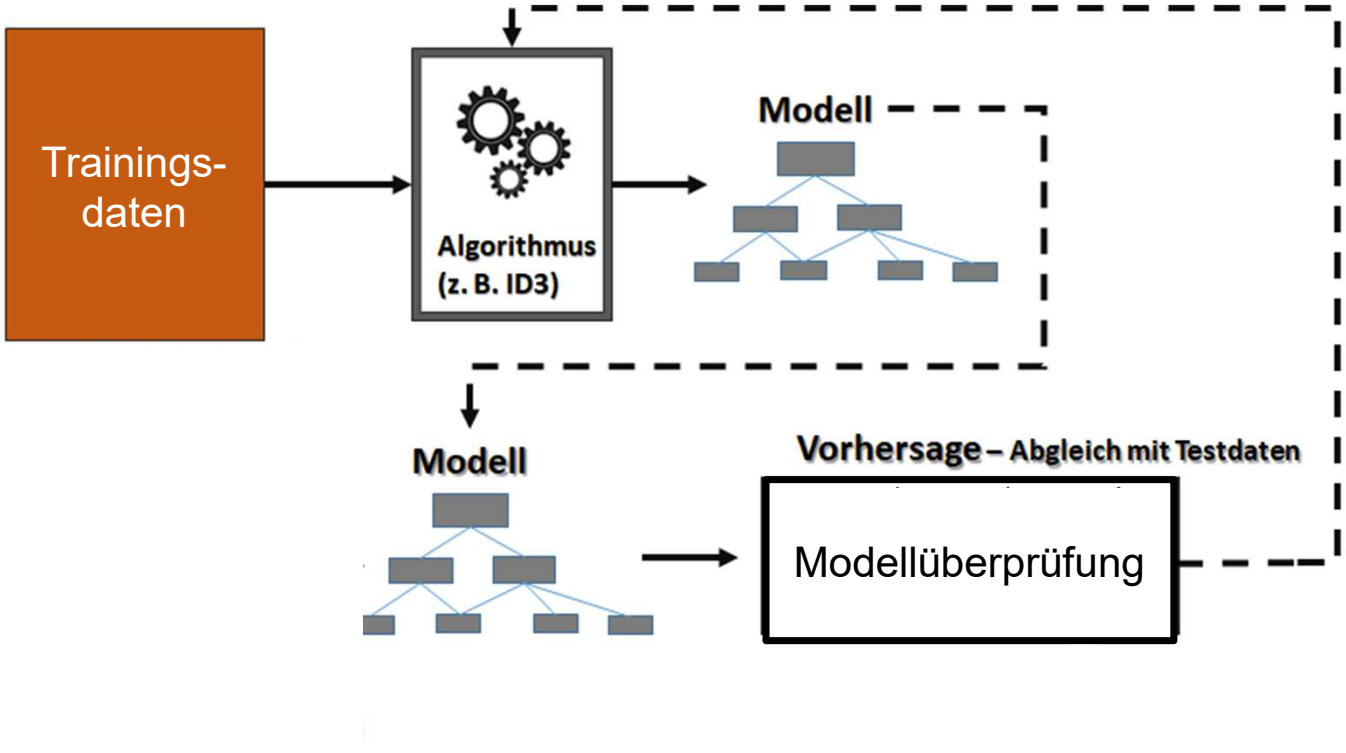
„Regression“ als Repräsentant des überwachten Lernens

Ziel: Approximation

- Bei den beiden bisher vorgestellten Problemstellungen ist das Ziel die Zuordnung von Eingangswerten zu einem diskreten Ausgangswert – einer Klasse oder einem Cluster.
  - ▶ Im Gegensatz dazu wird bei der Regression eine Funktion approximiert, um beliebigen Eingangswerten einen kontinuierlichen Ausgangswert zuordnen zu können.
  - ▶ Für die Approximation einer Funktion sind üblicherweise Stützstellen mit den dazugehörigen Stützwerten notwendig.
  - ▶ Daher ist die Regression eine Aufgabe für überwachtes Lernen.
  - ▶ Liegt beispielsweise das Wissen vor, dass Niederschlag, Temperatur und Treibstoffverbrauch zusammenhängen, kann durch eine Regression der veränderte Treibstoffverbrauch in Abhängigkeit von Niederschlag und Temperatur vorhergesagt werden.

- Die Problemstellungen können einzeln oder in Kombination auftreten, je nach zur Verfügung stehender Datensätze.
  1. Im Beispiel zu Clusterbildung wird ein Zusammenhang entdeckt.
  2. Das Beispiel zur Regression nutzt dieses neue Wissen, um eine Vorhersage zu treffen, die bisher nicht möglich war.

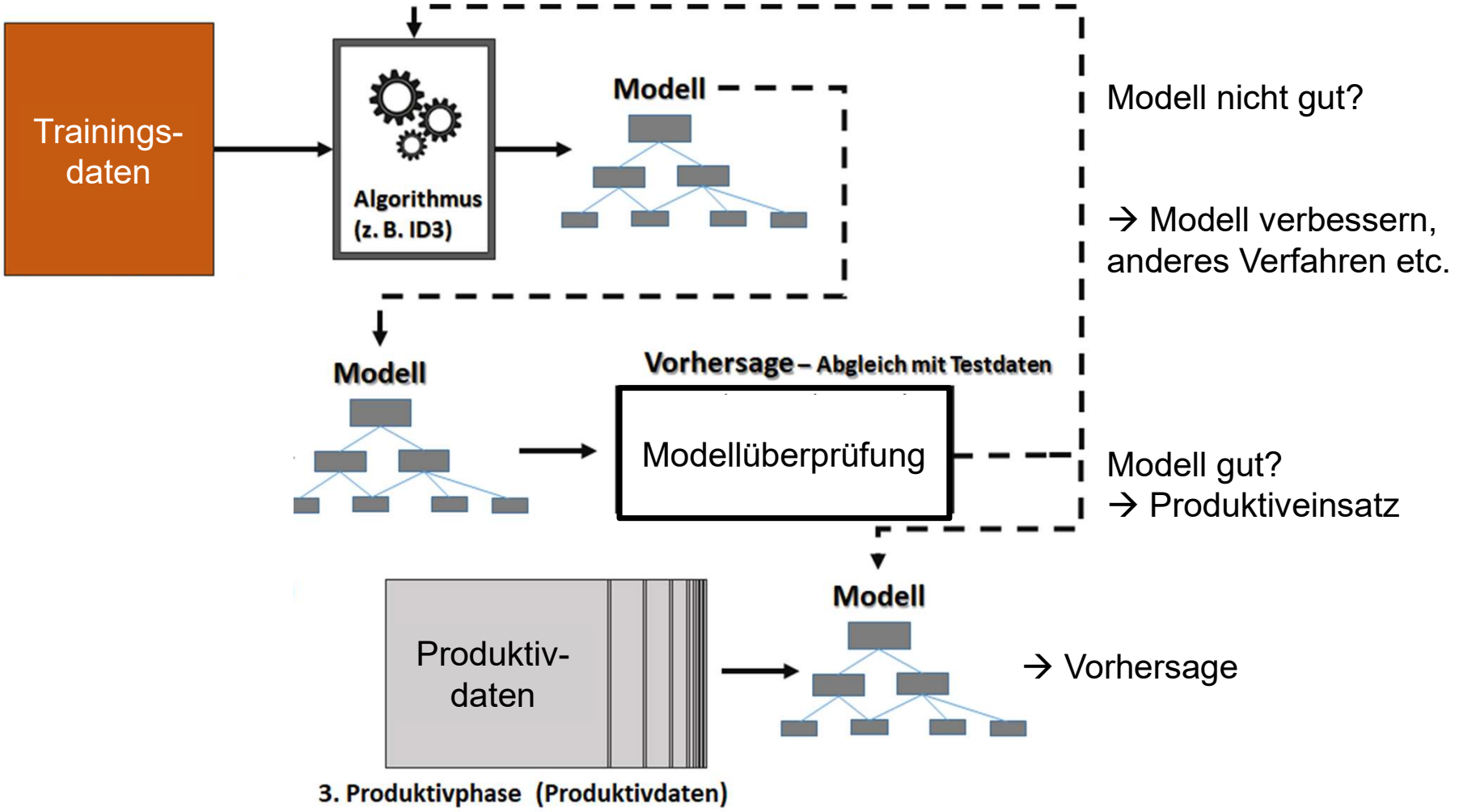
Grundsätzliches Vorgehen beim überwachten Lernen



Modell nicht gut?

→ Modell verbessern, anderes Verfahren etc.

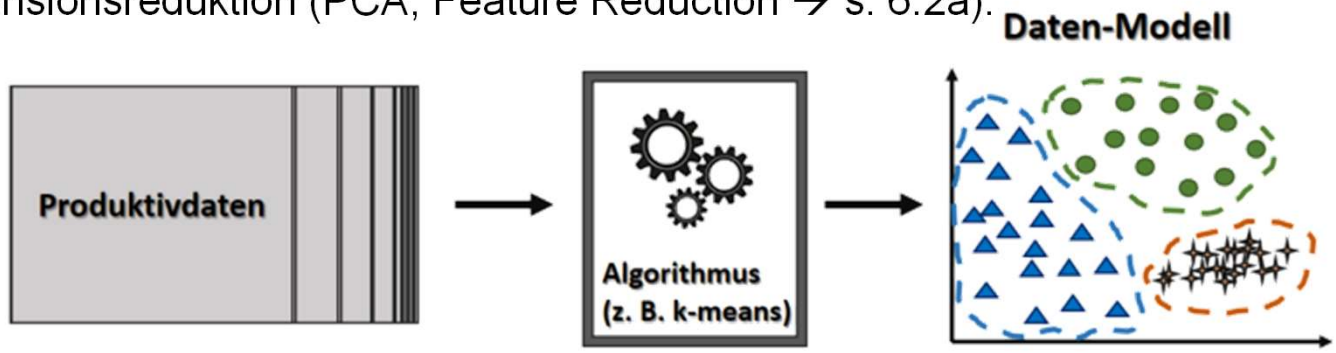
Grundsätzliches Vorgehen beim überwachten Lernen



Vorgehen beim unüberwachten Lernen

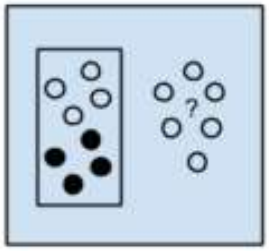
Mit unüberwachtem Lernen verborgene Strukturen identifizieren

- Beim unüberwachten Lernen geht es um nicht gekennzeichnete Daten (*unlabeled data*).
- Die möglichen Antworten/Ergebnisse sind gänzlich unbekannt.
- Folglich können wir den Algorithmus nicht trainieren, indem wir ihm die Ergebnisse, auf die er kommen soll, im Rahmen einer Trainingsphase vorgeben (vgl. überwachtes Lernen), sondern wir nutzen Algorithmen, die die Struktur der Daten erkunden und für uns Menschen sinnvolle Informationen aus Ihnen bilden.
  - ► oder auch nicht → häufig bleibt es beim Versuch → der Erfolg ist nicht garantiert!
- Zwei wesentliche Kategorien des unüberwachten Lernens
  - zum einem das Clustering, welches im Grunde ein unüberwachtes Klassifikationsverfahren darstellt
  - zum anderen die Dimensionsreduktion (PCA, Feature Reduction → s. 6.2a).



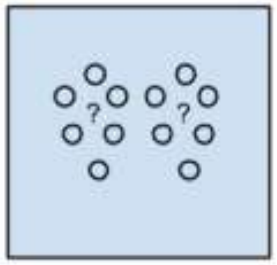
Unterschiedliche Arten von Lernen

Zusammenfassung



Supervised Learning

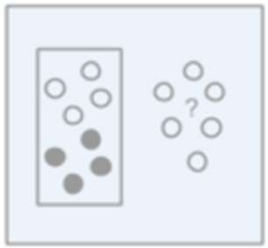
- Ge-labelte Daten
- Lernen/Vorhersagen von Output aus Input-Daten
- Herausforderung:
  - extrapolieren
  - generalisieren
- Beispiele
  - Klassifizierung
  - Regression



Unsupervised Learning

- Un-gelabelte Daten
- Finden von versteckten, Strukturen in Daten
- Herausforderung:
  - Subjektiver als SL
  - Validierung
- Beispiele
  - Clustering
  - Dimensionsreduktion

Unterschiedliche Arten von Lernen
Zusammenfassung



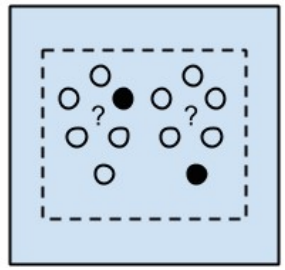
Supervised Learning

- Ge-labelte Daten
- Lernen/Vorhersagen von Output aus Input-Daten
- Herausforderung:
  - extrapolieren
  - generalisieren
- Beispiele
  - Klassifizierung
  - Regression



Unsupervised Learning

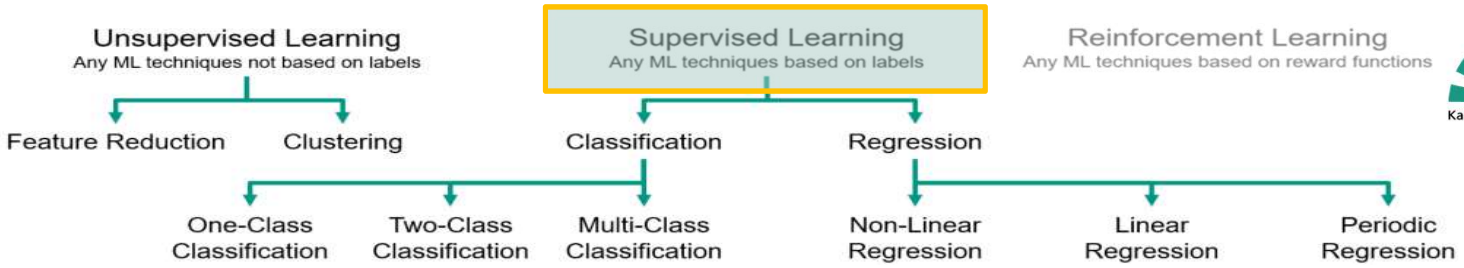
- Un-gelabelte Daten
- Finden von versteckten, Strukturen in Daten
- Herausforderung:
  - Subjektiver als SL
  - Validierung
- Beispiele
  - Clustering
  - Dimensionsreduktion



Semi-Supervised Learning

- nur ein Teil der Daten ist ge-labelt
- Nur für einen geringen Teil der Daten steht eine Annotation zur Verfügung
- geringerer Label-Aufwand im Vergleich zu Supervised Learning
- Bessere Leistung als bei Unsupervised Lernen

Machinelles Lernen



Algorithm ↓	Abk.	Feature Reduction	Clustering	One-Class Classification	Two-Class Classification	Multi-Class Classification	Non-Linear Regression	Linear Regression	Periodic Regression
Faktorenanalyse		X							
Principal Component Analysis	PCA	X		X					
	K-means		X						
hierarchical cluster analysis	HCA		X						
DBSCAN			X						
One Class Support Vector Machine	OCSVM			X					
Isolation Forest				X					
	LODA			X					
(künstliche) Neuronale Netze	NN			X (Autoencoder)	X	X	X	X	X
Support Vector Machine	SVM				X				
Decision Tree					X	X			
Bayes-Klassifikation					X	X			
Random Forest						X			
Diskriminanzanalyse				x	x	x			
Logistic Regression							X	X	
Linear Regression								X	
Harmonic Regression									X
Nächste-Nachbar-Klassifikation	K-NN	x	x						

Klassifikation vs. Regression

Beispiel

- Folgende Situation
  - Eine Versicherung bietet Leistungspaket D1 und D2 an.
  - Nun wird eine Werbekampagne mit Anreiz A konzipiert.

- Betrachtet man die folgenden drei ähnlichen Fragen:
  - *Wird der Kunde auf ein Angebot für das Paket D1 eingehen, wenn er den Anreiz A erhält?*
    - Hierbei handelt es sich um eine Klassifizierung, weil das Ziel binär ist (entweder geht der Kunde auf das Angebot ein oder nicht).
  - *Welches Dienstpaket (D1, D2 oder keines) wird der Kunde wahrscheinlich kaufen, wenn er den Anreiz A erhält?*
    - Hierbei handelt es sich ebenfalls um eine Klassifizierung, allerdings mit einer Zielvariablen, die 3 Werte annehmen kann.
  - *Wie häufig wird dieser Kunde das Paket bestellen/nutzen?*
    - Dies ist ein klassisches Beispiel für eine Regression, denn die Zielvariable ist numerisch.

Klassifikation vs. Regression

Beide gehören den überwachten Verfahren an.

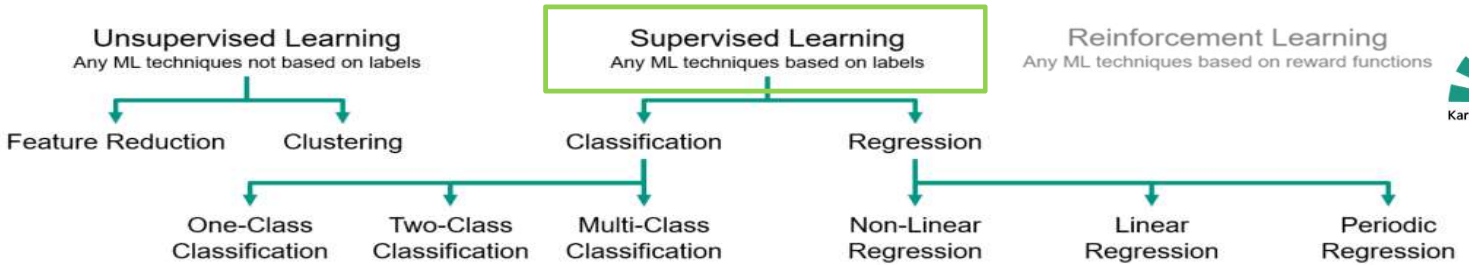
- Klassifizierung versucht, für ein zur Grundgesamtheit gehörendes Individuum vorherzusagen, zu welchen (einigen wenigen) Klassen dieses Individuum gehört.
  - Für gewöhnlich schließt die Zugehörigkeit zu einer Klasse die Zugehörigkeit zu einer anderen aus.
- Eine Frage, die der Klassifizierung dient, ist bspw.:
  - *»Welche Kunden werden ein bestimmtes Angebot wahrscheinlich annehmen?«*
    → In diesem Beispiel könnte man die beiden Klassen als „wird annehmen“ und „wird nicht annehmen“ bezeichnen.
- Die Klassifizierung liefert ein Modell, das bei Auftauchen eines neuen Individuums ermittelt, zu welcher Klasse dieses Individuum gehört.

- Regression versucht, für jedes Individuum den numerischen Wert einer Variable abzuschätzen oder vorherzusagen.
  - *»Wie oft wird ein bestimmter Kunde den Dienst nutzen?«*
    → ist ein Beispiel für eine Frage, die durch eine Regression beantwortet werden kann.
- Die hier vorherzusagende Eigenschaft (die Variable) ist die Dienstnutzung.
- Man könnte dafür ein Modell entwickeln, indem man die Dienstnutzung vergleichbarer Individuen der Grundgesamtheit betrachtet.
- Eine Regression liefert ein Modell, das den Wert einer Variable für ein bestimmtes Individuum abschätzt.

Anmerkungen

- Bei der Regression ist die Zielvariable numerisch, bei der Klassifizierung hingegen geht es um die Zuordnung zu einem (oft binären) kategorialen Ziel.
  - Zwanglos könnte man sagen, dass eine Klassifizierung vorhersagt, ob etwas der Fall ist, eine Regression hingegen sagt voraus, in welchem Ausmaß dies eintritt.
- Scoring ist sehr ähnlich.
  - Ein auf ein Individuum angewendetes Scoring-Modell ermittelt statt einer Klassenvorhersage eine Punktzahl, die die Wahrscheinlichkeit dafür angibt, dass ein Individuum zu einer bestimmten Klasse gehört.
- Im Szenario der Angebotsannahme würde ein Scoring-Modell die einzelnen Kunden auswerten und eine Punktzahl liefern, die angibt, wie wahrscheinlich es ist, dass ein Kunde auf das Angebot eingeht.

Machinelles Lernen



Algorithm ↓	Abk.	Feature Reduction	Clustering	One-Class Classification	Two-Class Classification	Multi-Class Classification	Non-Linear Regression	Linear Regression	Periodic Regression
Faktorenanalyse		X							
Principal Component Analysis	PCA	X		X					
	K-means		X						
hierarchical cluster analysis	HCA		X						
DBSCAN			X						
One Class Support Vector Machine	OCSVM			X					
Isolation Forest				X					
	LODA			X					
(künstliche) Neuronale Netze	NN			X (Autoencoder)	X	X	X	X	X
Support Vector Machine	SVM				X				
Decision Tree					X	X			
Bayes-Klassifikation					X	X			
Random Forest						X			
Diskriminanzanalyse				x	x	x			
Logistic Regression							X	X	
Linear Regression								X	
Harmonic Regression									X
Nächste-Nachbar-Klassifikation	K-NN	x	x						

Regression
Begriff

- Re·gres·si·on
  - Langsamer Rückgang; rückläufige Tendenz
  - Gegenteil: Progression
- Die Regression kann als die „Mutter aller Verfahren“ bezeichnet werden.
  - ► Das liegt einerseits daran, dass sie in ihrer einfacheren Ausprägung als lineare Regression gut verständlich und damit nachvollziehbar ist, aber auch daran, dass die im Verfahren eingesetzten Methoden in vielen anderen Verfahren eine analoge Anwendung finden.
  - ► Als Prognoseverfahren ist es sicher das am häufigsten verwendete Schätzmodell in der Praxis.
- Folglich
  - ► Regressions-Analysen sind statistische Analyseverfahren, die zum Ziel haben, Beziehungen zwischen einer abhängigen und einer oder mehreren unabhängigen Variablen zu modellieren.
  - ► Sie werden insbesondere verwendet, wenn Zusammenhänge quantitativ zu beschreiben oder Werte der abhängigen Variablen zu prognostizieren sind.

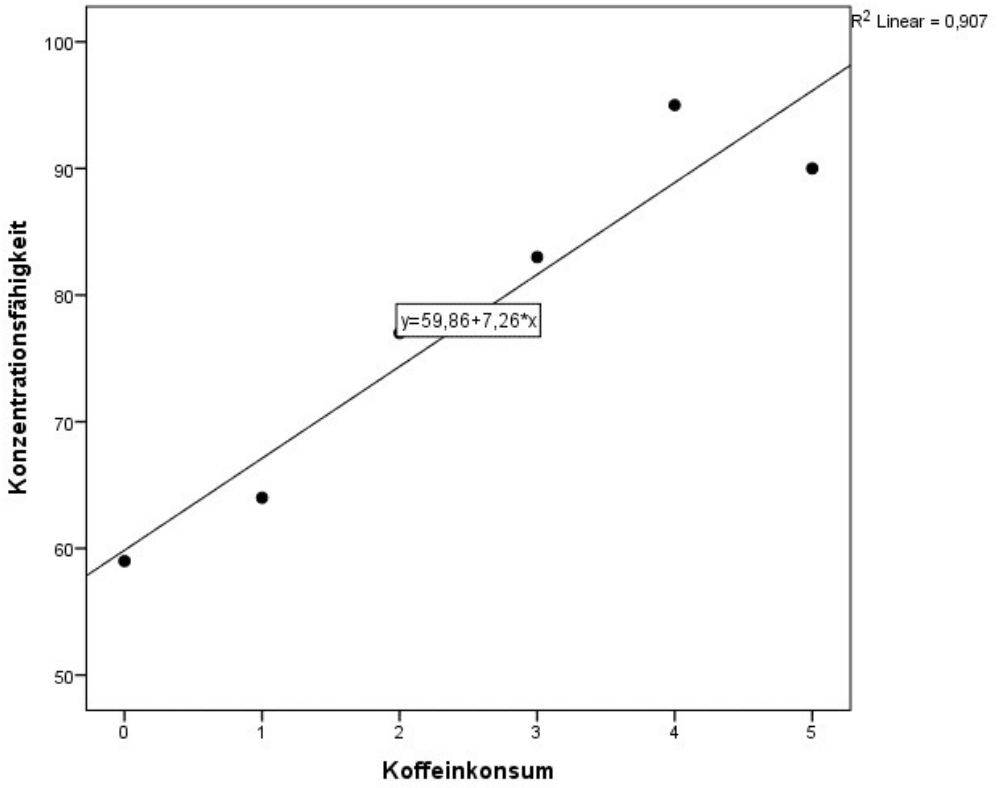

8RT

Einfachregression

Univariate Regression

- Der einfachste Fall ist die lineare, univariate Regression, also mit jeweils 1 messbaren abhängigen und unabhängigen Variablen .
  - Es wird ein linearer Zusammenhang zwischen den Variablen vermutet.
- Beispiel: 6 Probanden, die 0 bis 5 Tassen Kaffee trinken.
  - Annahme → Wir haben eine Methode zur Messung der Konzentrationsfähigkeit nach Kaffeekonsum
  - Ergebnis/Visualisierung → s. rechts
    - Y-Achse (abhängige Variable) ist die Konzentrationsfähigkeit
    - X-Achse (unabhängige Variable) ist der Koffeinkonsum
- Der Zusammenhang der beiden Variablen kann mit einer linearen Funktion dargestellt werden

$$y = \beta_0 + \beta_1 x_1$$



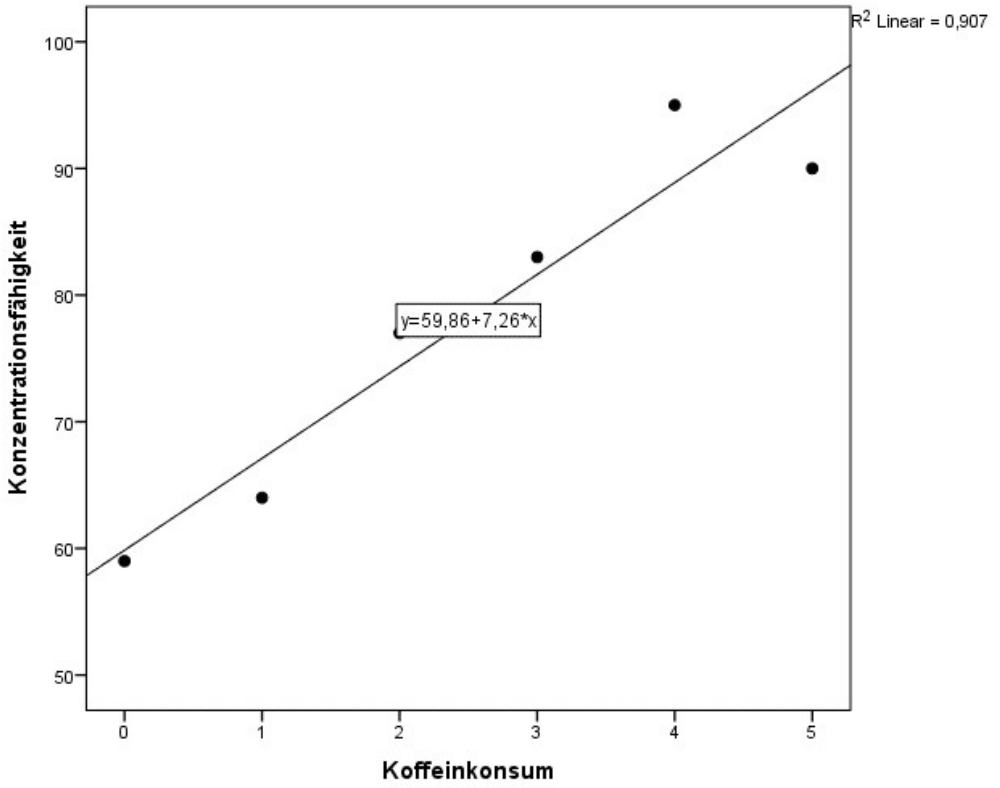
	Person 1	Person 2	Person 3	Person 4	Person 5	Person 6
Koffeinkonsum (x)	3	0	5	2	1	4
Konzentrationsfähigkeit (y)	83	59	90	77	64	95

univariat: nur eine Variable besitzend; eindimensional
linear: vergleiche PCA

Einfachregression

Interpretation

- Man versucht nun, die Regressionsgerade so in die Punktewolke einzupassen, dass die Abstände der Punkte von der Gerade minimiert werden
  - häufig wird hierbei das Quadrat der Abstände genommen → Ziel ist die Optimierung des mittleren quadratischen Fehlers.
- Wenn man mit den statistischen Gütezahlen zufrieden ist (also z. B. den quadrierten Abweichungsdistanzen), erhält man dann ein Modell, das für die Prognose verwendet werden kann.



	Person 1	Person 2	Person 3	Person 4	Person 5	Person 6
Koffeinkonsum (x)	3	0	5	2	1	4
Konzentrationsfähigkeit (y)	83	59	90	77	64	95

$$y = \beta_0 + \beta_1 x_1 \rightarrow y = 59{,}86 + 7{,}26 * x$$


8RT

Mehrfachregression

Komplexität: Häufig ist die Verteilung nicht linear approximierbar.

- Man wird daher:
  - … weitere erklärende Variablen miteinbeziehen wollen. Dadurch bekommt man eine nicht mehr so einfach zu visualisierende multivariate Regression.

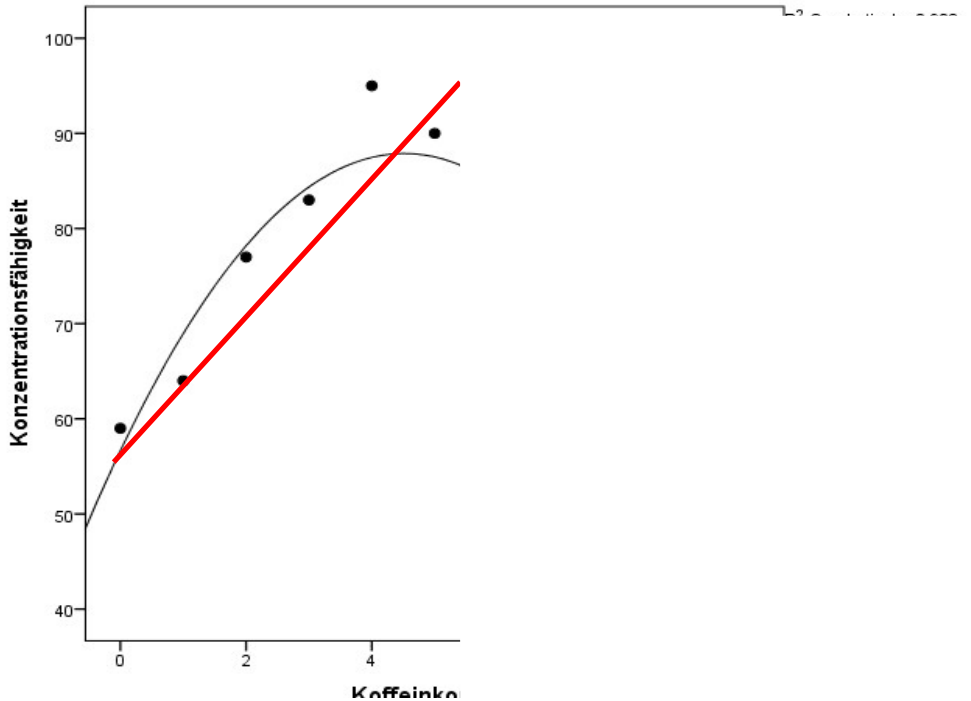
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$$

  - … das Schätzverfahren ändern. Durch das Quadrieren der Abstände haben einzelne Ausreißer eine relativ große Auswirkung auf die Regressionsfunktion, die man eventuell begrenzen möchte.
  - … keinen linearen Zusammenhang annehmen, sondern versuchen, die Art des funktionalen Zusammenhangs anhand der Daten herzuleiten.
- *Remark: Nichtlineare Zusammenhänge können durch eine nichtlineare Regression modelliert werden.*
  - *Hierzu wird versucht den nichtlinearen Ansatz in einen linearen zu transformieren, um dann die Gleichung mithilfe der Gauß'schen Methode der kleinsten Quadrate zu bestimmen*

Beispiel für eine nichtlineare Regression

„Koffeinkonsum“

- Mglw. führt eine zu hohe Koffeindosierung wieder zu einer Abnahme der Konzentrationsfähigkeit.
  - Dazu erweitert man die Stichprobe um 5 Personen, die zwischen 6 und 10 Tassen Kaffee trinken.



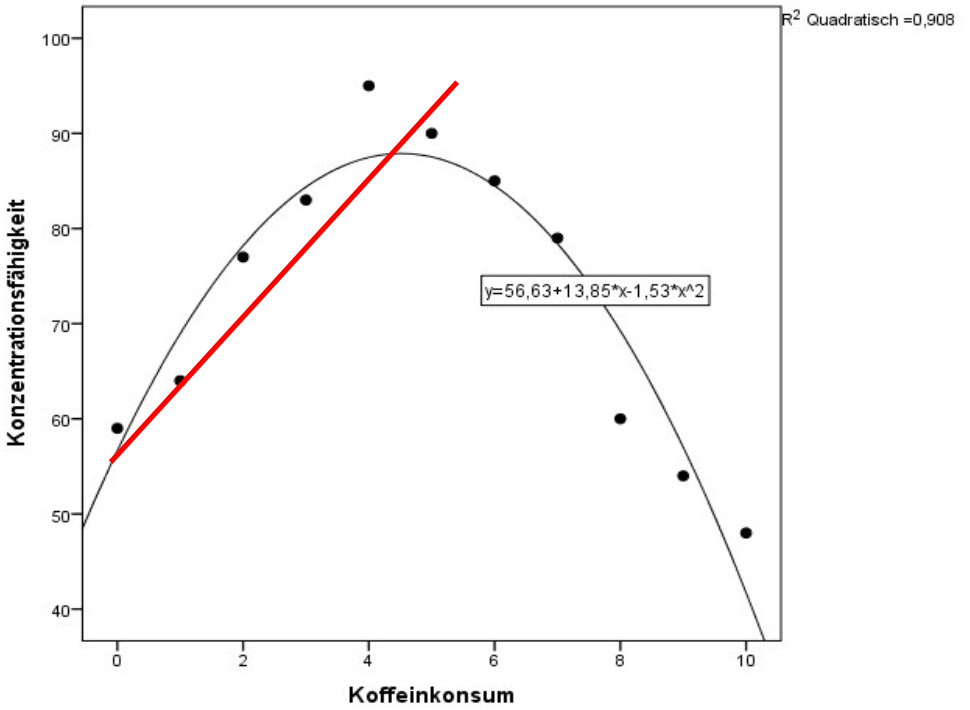
	Person 1	Person 2	Person 3	Person 4	Person 5	Person 6
Koffeinkonsum (x)	3	0	5	2	1	4
Konzentrationsfähigkeit (y)	85	41	87	64	58	53

Beispiel für eine nichtlineare Regression

„Koffeinkonsum“

- Mglw. führt eine zu hohe Koffeindosierung wieder zu einer Abnahme der Konzentrationsfähigkeit.
  - Dazu erweitert man die Stichprobe um 5 Personen, die zwischen 6 und 10 Tassen Kaffee trinken.
  - Vermutung offensichtlich richtig → ein zu hoher Koffeinkonsum führt wieder zu einer Abnahme der Konzentrationsfähigkeit.
  - Der ursprünglich lineare Zusammenhang war eine Täuschung, da man die Beziehung der Variablen lediglich bis 5 Tassen Kaffee untersucht hatte.
- Eine Kurve wie bei unseren Daten ist eine sogenannte quadratische Funktion.
  - Quadratische Funktionen sind allerdings ähnlich unkompliziert wie lineare Funktionen, da man sie im Plot leicht anhand ihrer Kurvenform erkennen kannst.
- Man erweitert die ursprüngliche Regressions-gleichung um den Term x^2.
  - Die Gleichung sieht dann folgendermaßen aus:

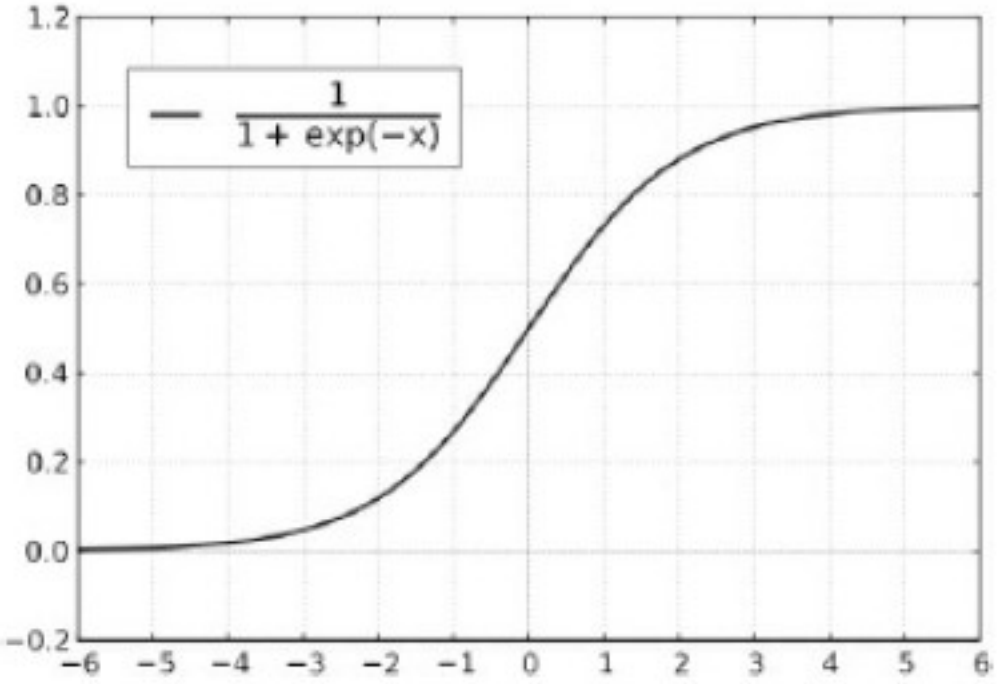
$$y = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2$$



	Person 1	Person 2	Person 3	Person 4	Person 5	Person 6
Koffeinkonsum (x)	3	0	5	2	1	4
Konzentrationsfähigkeit (y)	85	41	87	64	58	53
	Person 7	Person 8	Person 9	Person 10	Person 11	
Koffeinkonsum (x)	8	10	7	9	6	
Konzentrationsfähigkeit (y)	60	48	79	54	85	

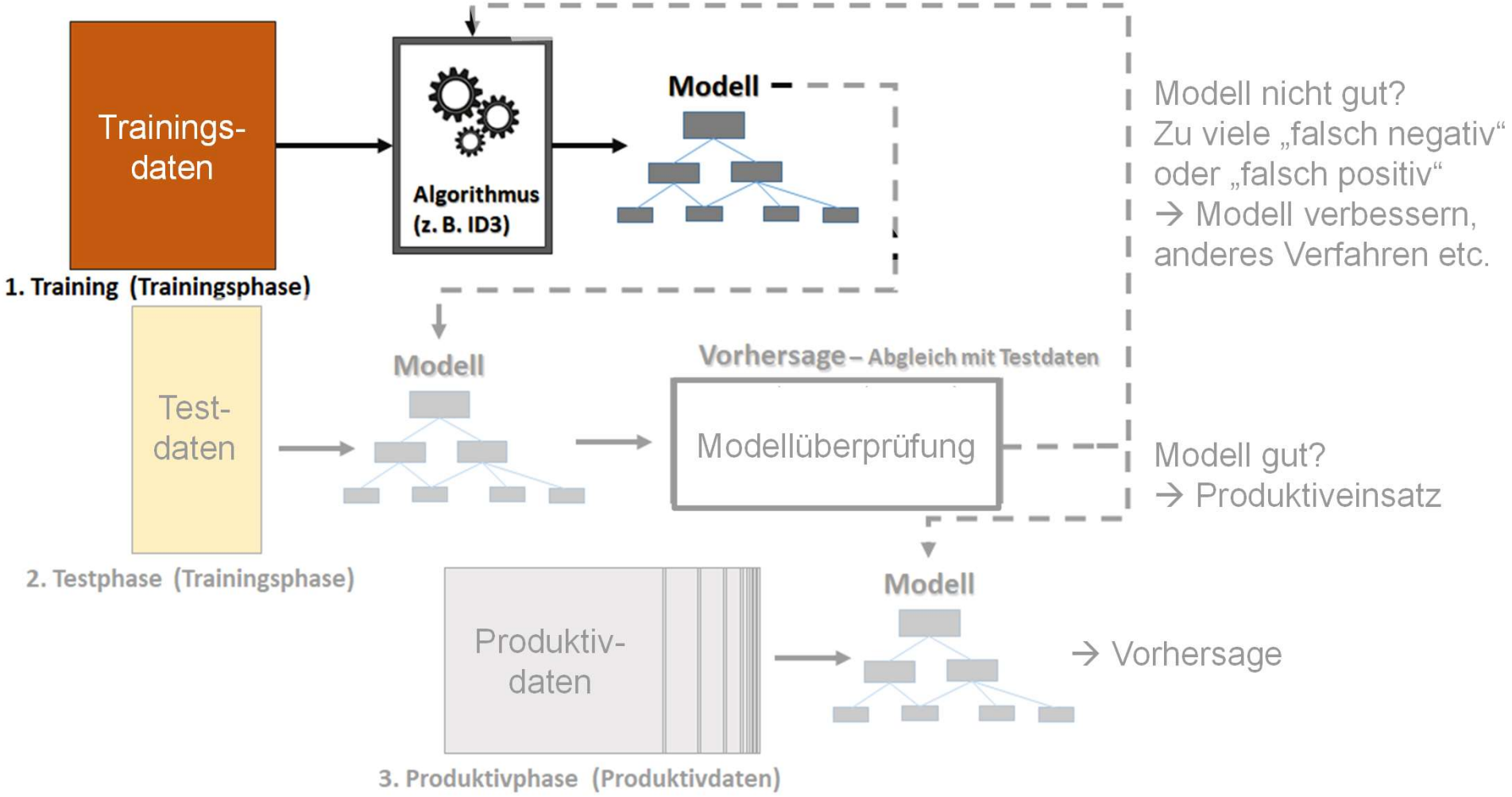
Logistische Regression

- Darunter versteht man eine Regressions-analyse zur Modellierung der Verteilung diskreter abhängiger Variablen, z.B. im Falle von Ja-Nein-Entscheidungen (z.B. ob der Kunde kreditwürdig ist oder nicht).
- Man könnte derartige Entscheidungsmodelle auch mit einer linearen Regression lösen, wobei man dann einen Wert für y als Grenze zwischen „Ja“ und „Nein“ definiert.
- Das führt dazu, dass Werte knapp an diesem Wert relativ willkürlich in die eine oder andere Klasse zugeordnet würden.
- Die logistische Verteilfunktion mit ihrer hohen Steigung im „Mittelteil“ der Kurve teilt sozusagen das „Ja“ besser von dem „Nein“, da der Bereich in der Nähe des Cut Offs sehr steil ist und damit nur wenige Werte „betrifft“.



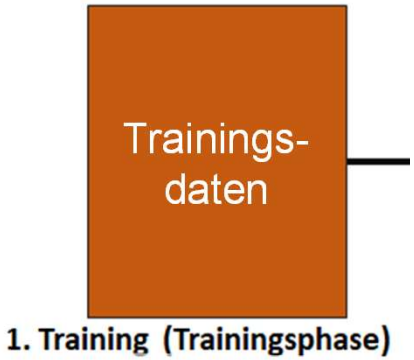
Grundsätzliches Vorgehen beim überwachten Lernen

Trainieren

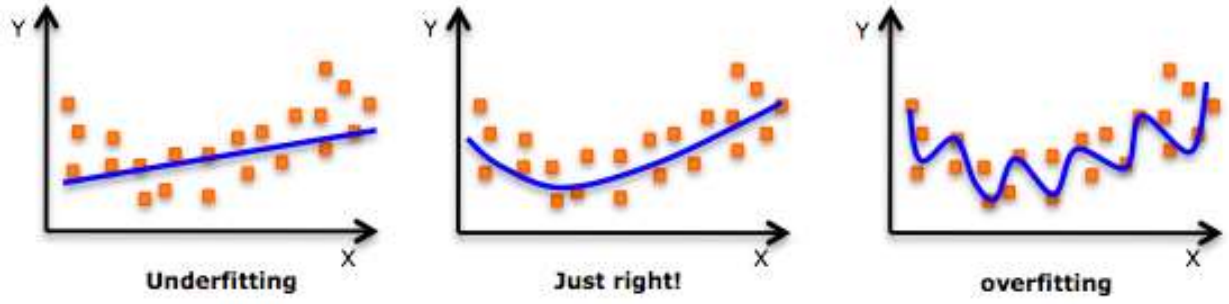


Vorgehen beim überwachten Lernen

Over-/Underfitting

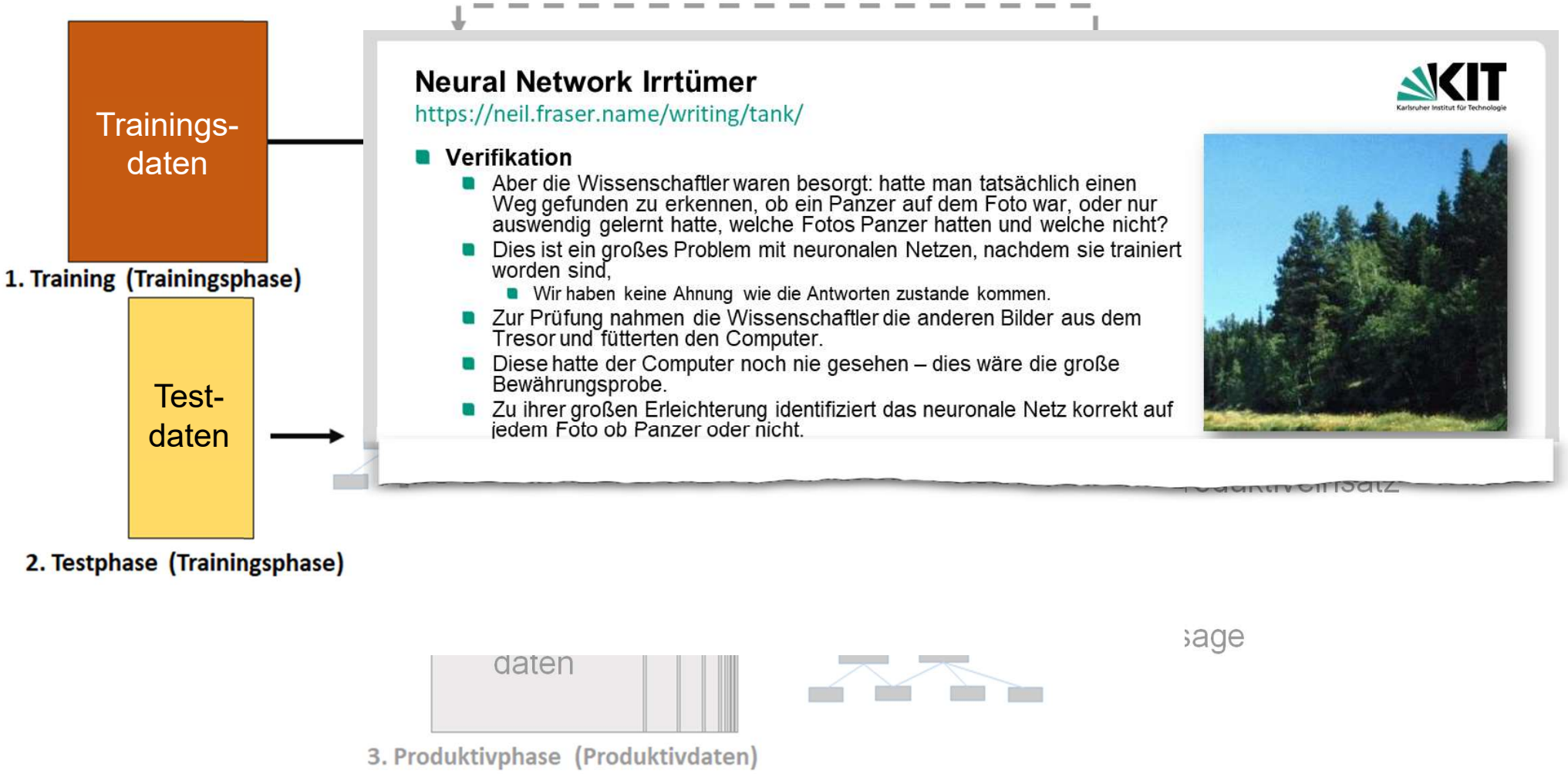


- Overfitting
  - Trainiertes Modell ist "zu gut" trainiert und nun zu eng mit dem Trainingsdatensatz übereinstimmend.
  - Dies geschieht, wenn das Modell zu komplex ist (d.h. zu viele Merkmale/Variablen im Vergleich zur Anzahl der Beobachtungen).
  - Das Modell wird sehr genau auf die Trainingsdaten sein, aber wahrscheinlich sehr ungenau auf untrainierte oder neue Daten.
- Underfitting
  - Im Gegensatz dazu bedeutet Underfitting eines Modells, dass das Modell nicht zu den Trainingsdaten passt und somit die Muster in den Daten nicht erkennt.
  - Es kann nicht auf neue Daten verallgemeinert werden kann.

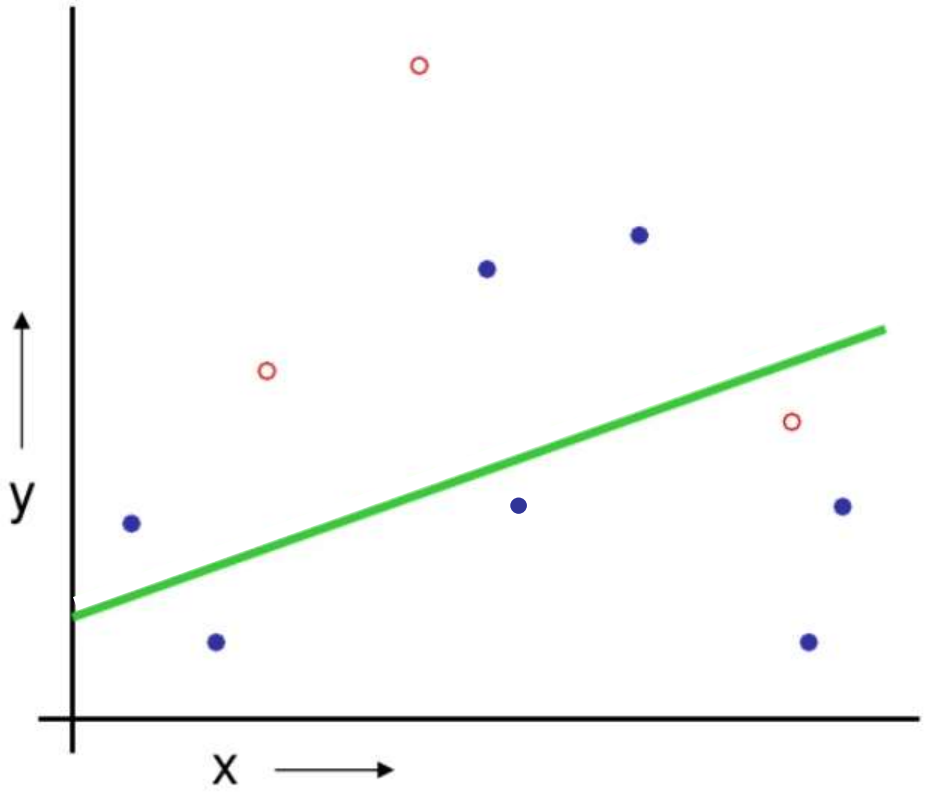


Test trainierter Modelle

Aufteilung der Daten in Training und Test notwendig, um Überanpassung zu vermeiden

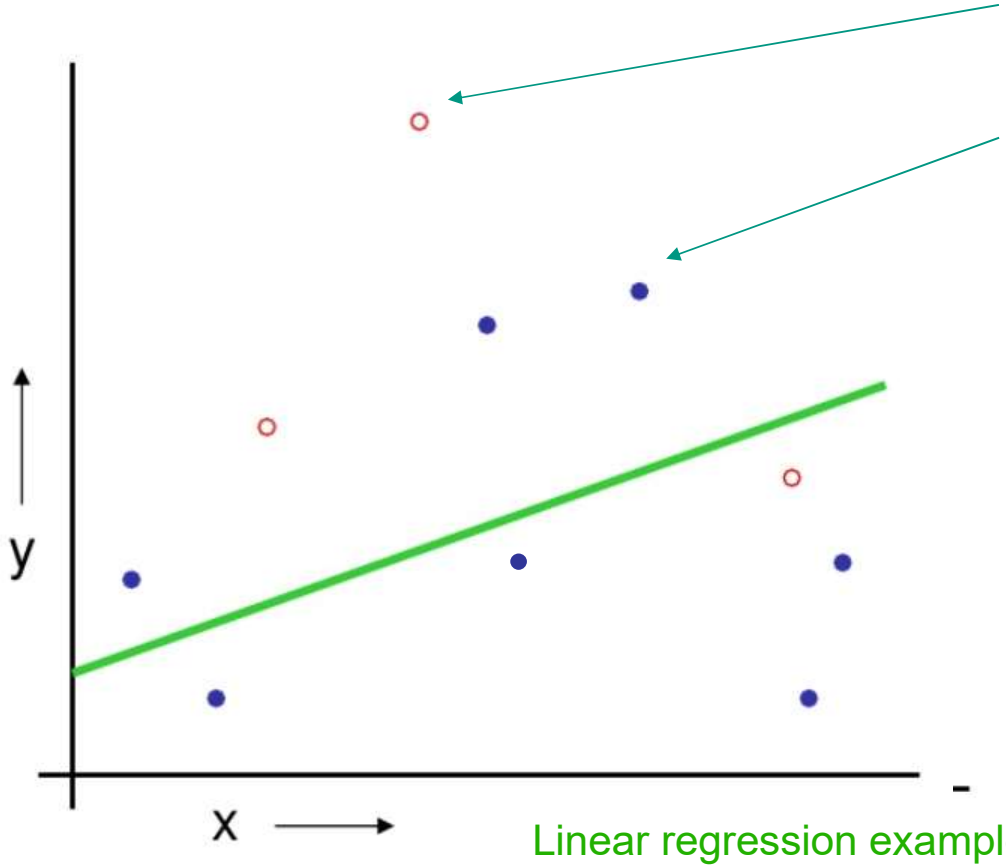


Train/Test Split



1. Wähle zufällig 30% der Daten als Testset aus
2. Die restlichen 70% der Daten ist das Trainingset

Train/Test Split



Linear regression example

1. Wähle zufällig 30% der Daten als Testset aus
2. Die restlichen 70% der Daten ist das Trainingset
3. Führe Regression anhand des Trainingssets aus

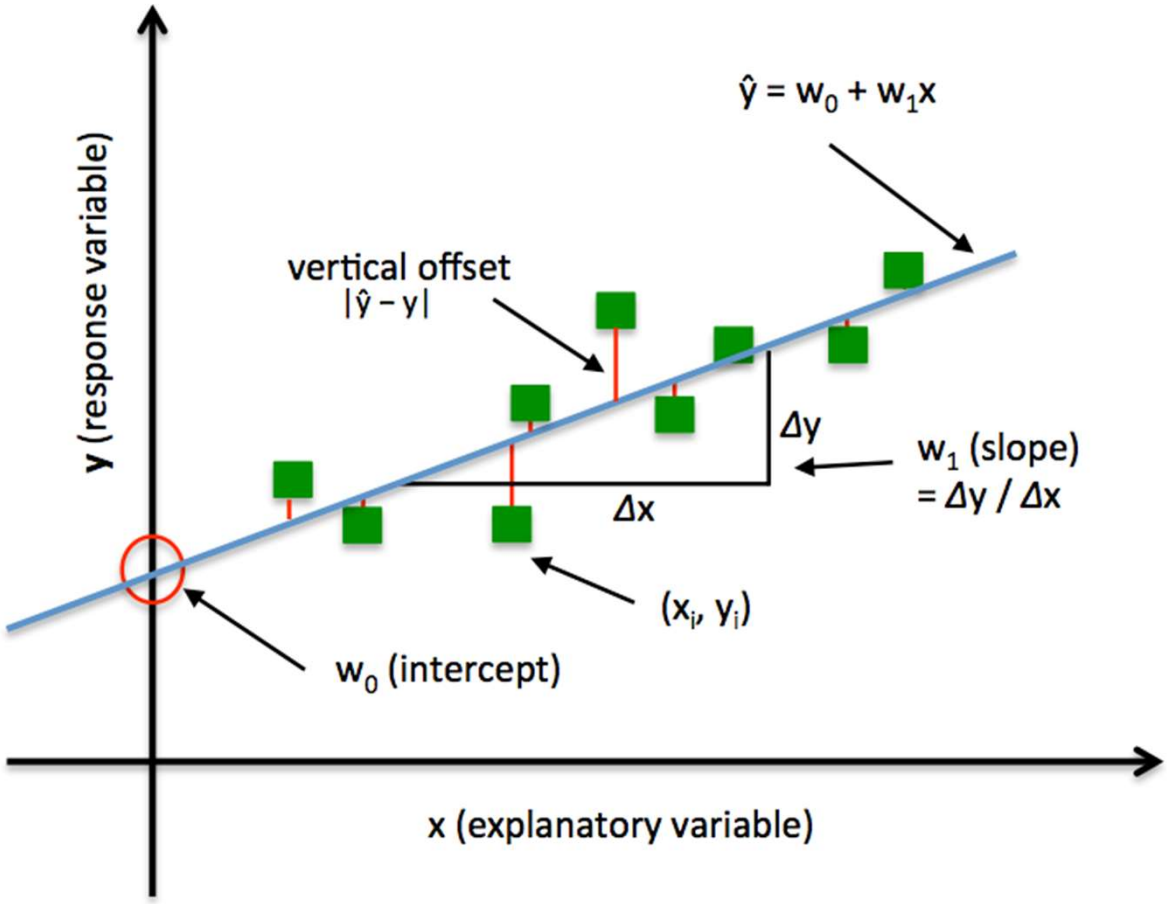
Gütemaße für die Evaluation von Modellen

Regression

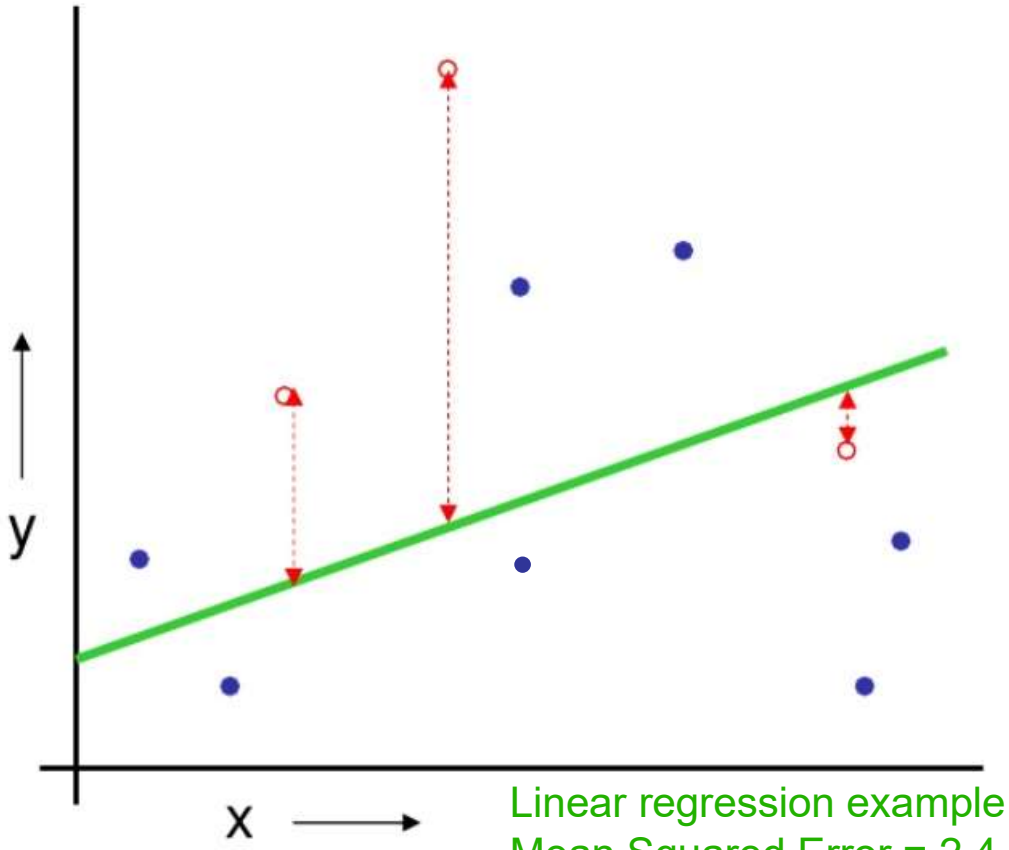
- **Mittlerer Quadratischer Fehler (MQF) (Mean Squared Error** MSE)
  - $MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$
- **Root Mean Squared Error** (RMSE)
  - $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}\, y$
- **Mean Absolut Error** (MAE)
  - $MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$

$y_i = tatsächlicher\ Werte$

$\hat{y}_i = korrekte\ Werte$



Train/Test Split

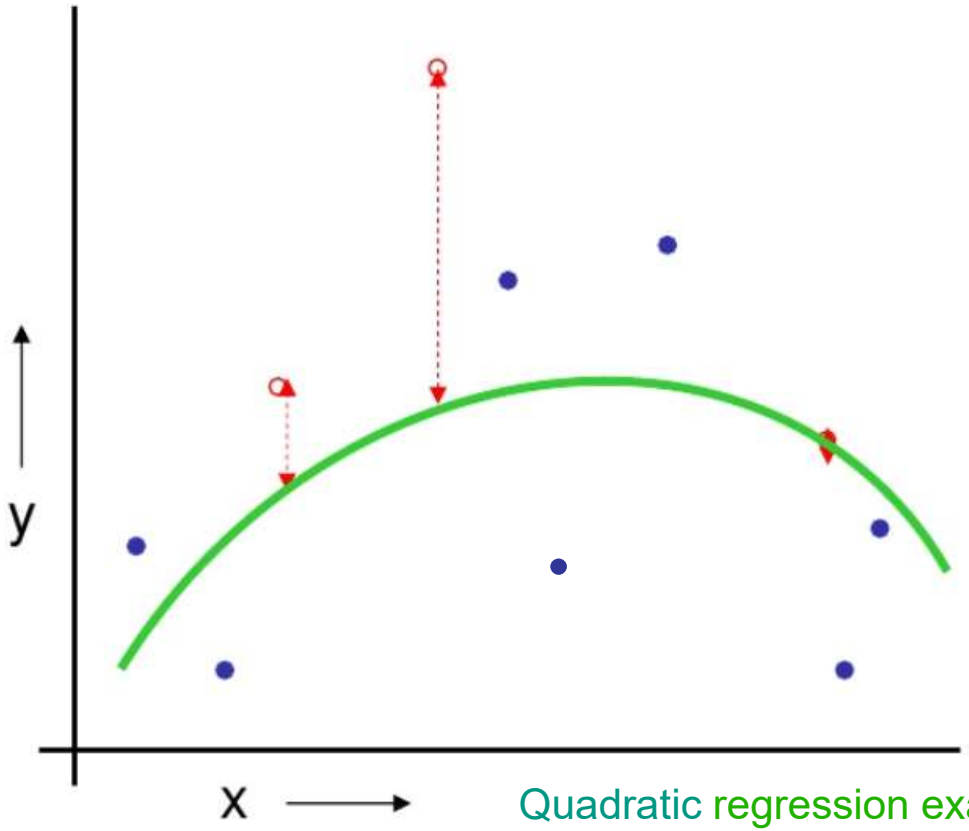


Linear regression example
Mean Squared Error = 2.4

1. Wähle zufällig 30% der Daten als Testset aus
2. Die restlichen 70% der Daten ist das Trainingset
3. Führe Regression anhand des Trainingssets aus
4. Schätze zukünftige Performance anhand des Testsets

Train/Test Split

Quadratische Regression

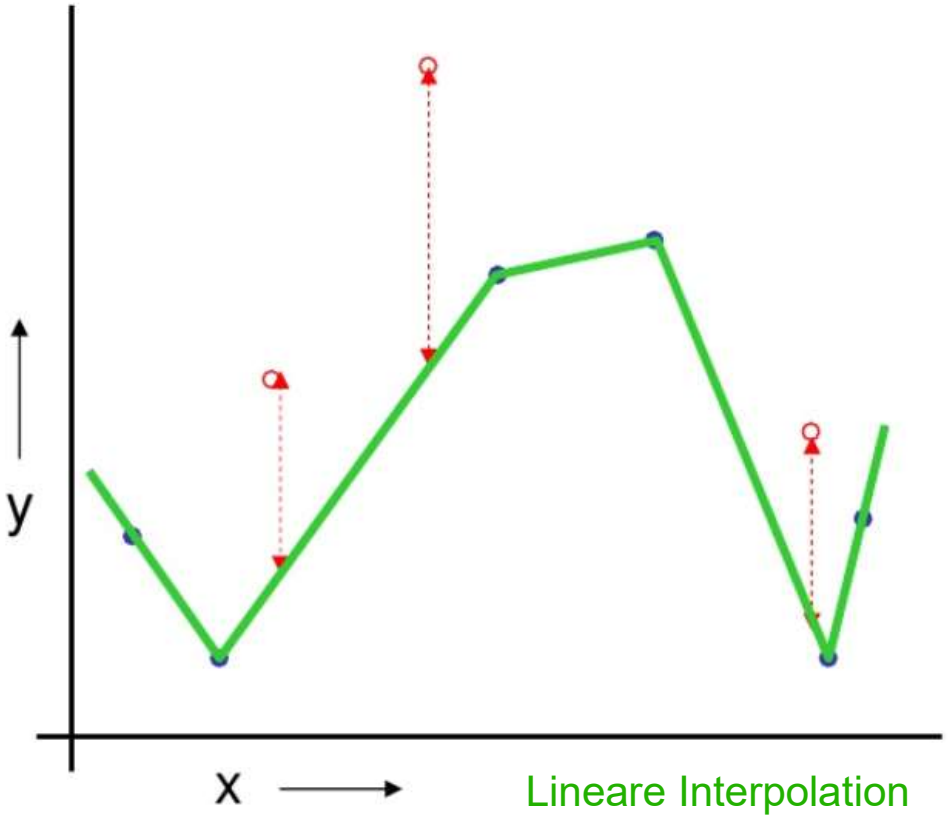


Quadratic regression example
Mean Squared Error = 0.9

1. Wähle zufällig 30% der Daten als Testset aus
2. Die restlichen 70% der Daten ist das Trainingset
3. Führe Regression anhand des Trainingssets aus
4. Schätze zukünftige Performance anhand des Testsets

Train/Test Split

„Connect the Dots“ → Ein Verfahren welches jeden Datenpunkt linear verbindet)

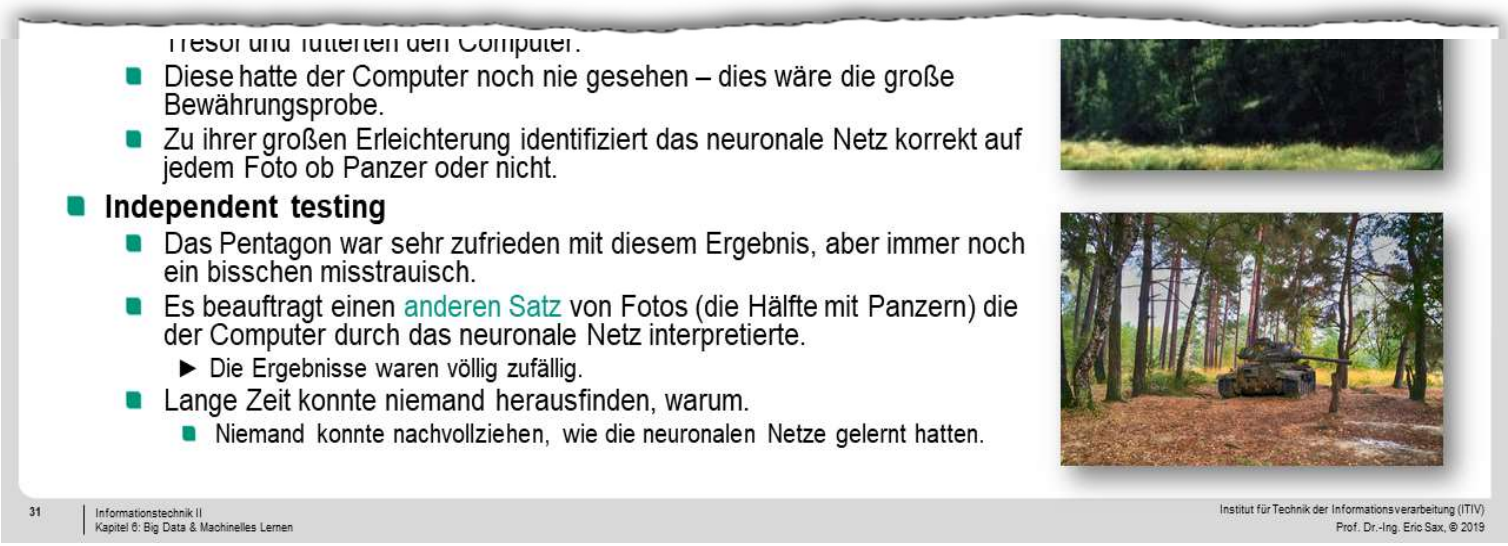


Lineare Interpolation
Mean Squared Error = 2.2

1. Wähle zufällig 30% der Daten als Testset aus
2. Die restlichen 70% der Daten ist das Trainingset
3. Führe Regression anhand des Trainingssets aus
4. Schätze zukünftige Performance anhand des Testsets

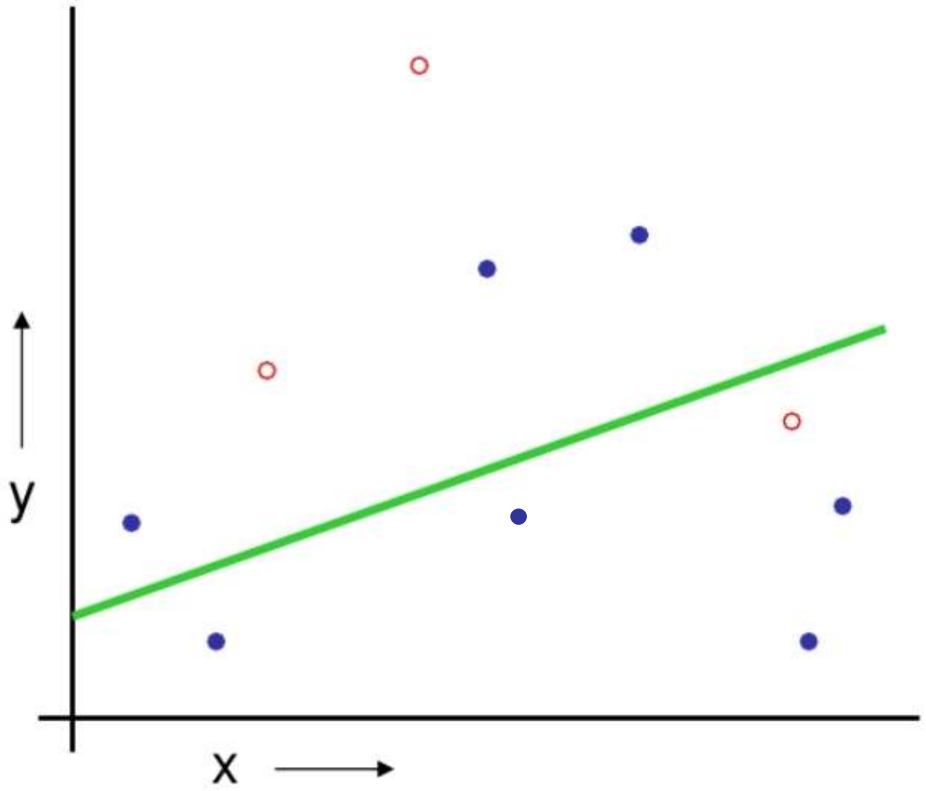
Train/Test Split

- Vorteile
  - Sehr einfach
  - Einfache Auswahl der Methode anhand des besten test-set score (= Mean Squared Error)
- Nachteile
  - Datenverschwendung
    - Wir benutzen 30% weniger Daten zu Erzeugung des Modells
  - Falls nur wenige Daten zur Verfügung stehen, besitzt das Testset nur geringe Varianz gegenüber Trainingsdaten



LOOCV- Leave-one-out Cross Validation

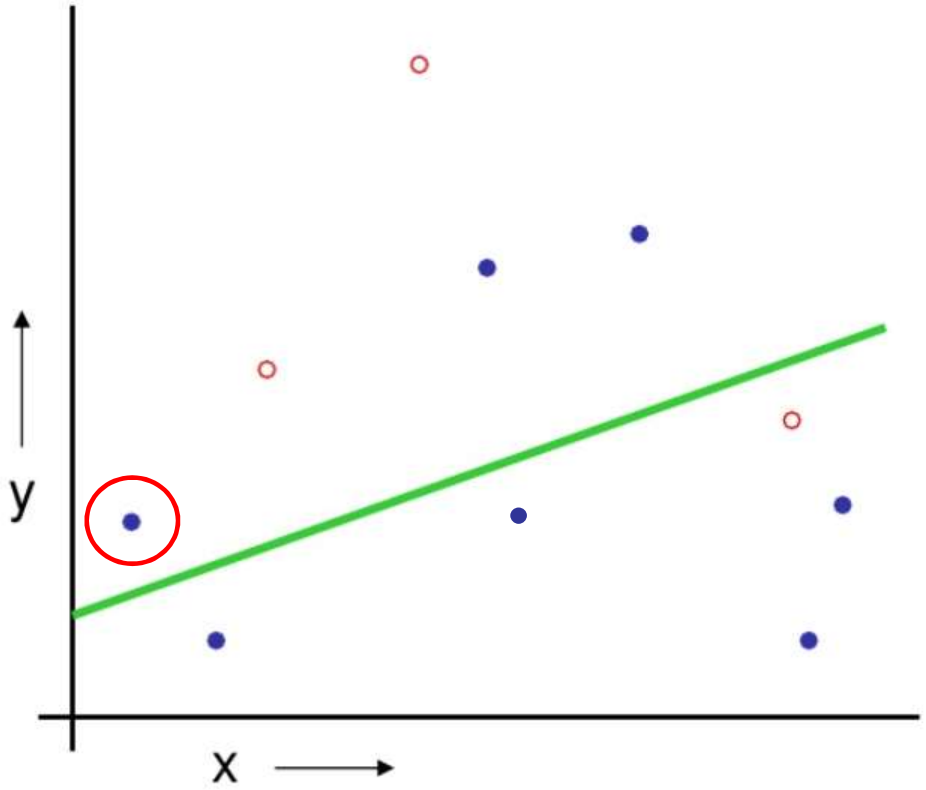
Kreuzvalidierungsverfahren



- Es wird ein Modell durch lineare Regression erstellt und dann der MSE (mean squared error) über alle Folds für dieses Modell berechnet.

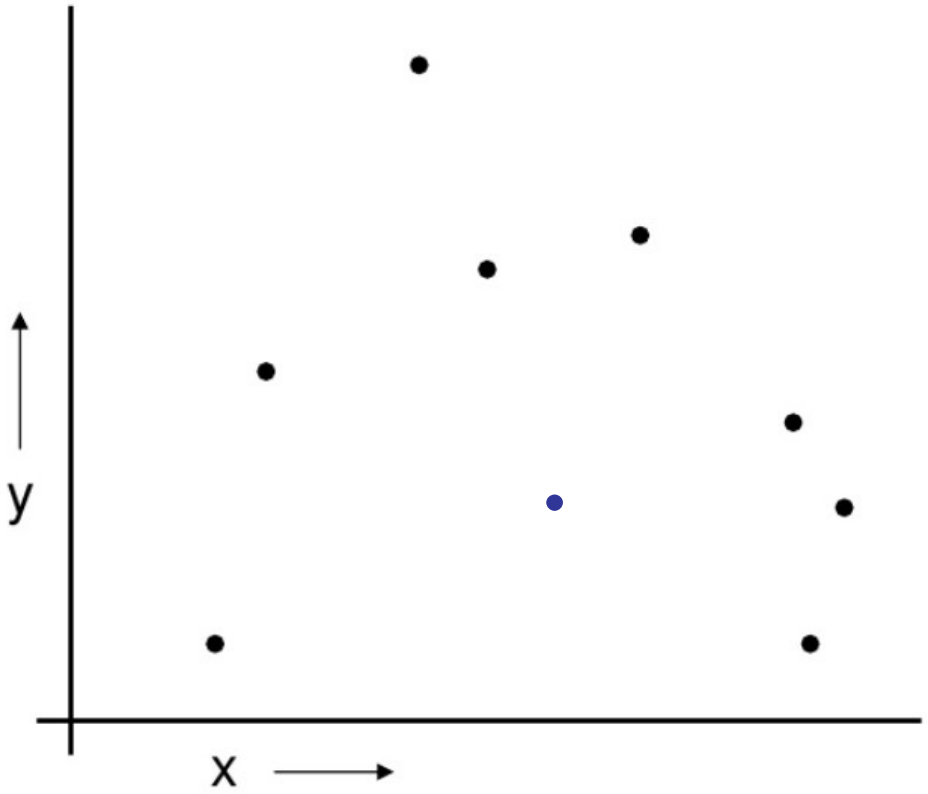
LOOCV- Leave-one-out Cross Validation

Kreuzvalidierungsverfahren



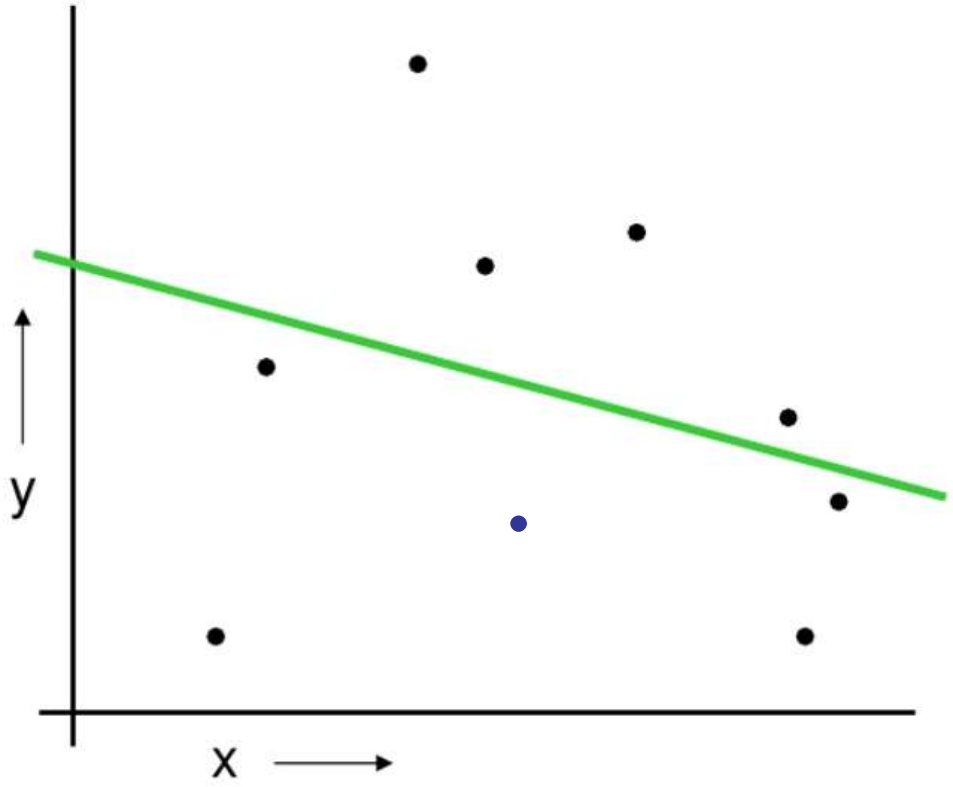
- Für k=1 bis n Datenpunkte
  1. Wähle (x_k, y_k) aus

LOOCV- Leave-one-out Cross Validation



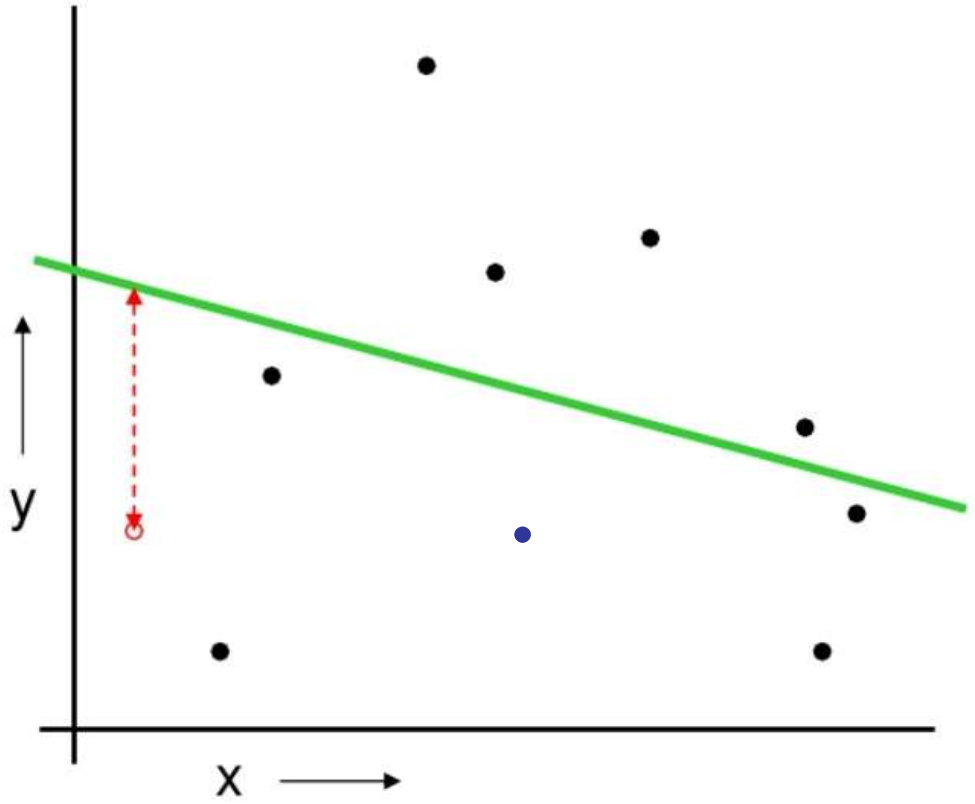
- Für k=1 bis n Datenpunkte
  1. Wähle (x_k, y_k) aus
  2. Entferne Datenpunkt (x_k, y_k) temporär aus Dataset

LOOCV- Leave-one-out Cross Validation



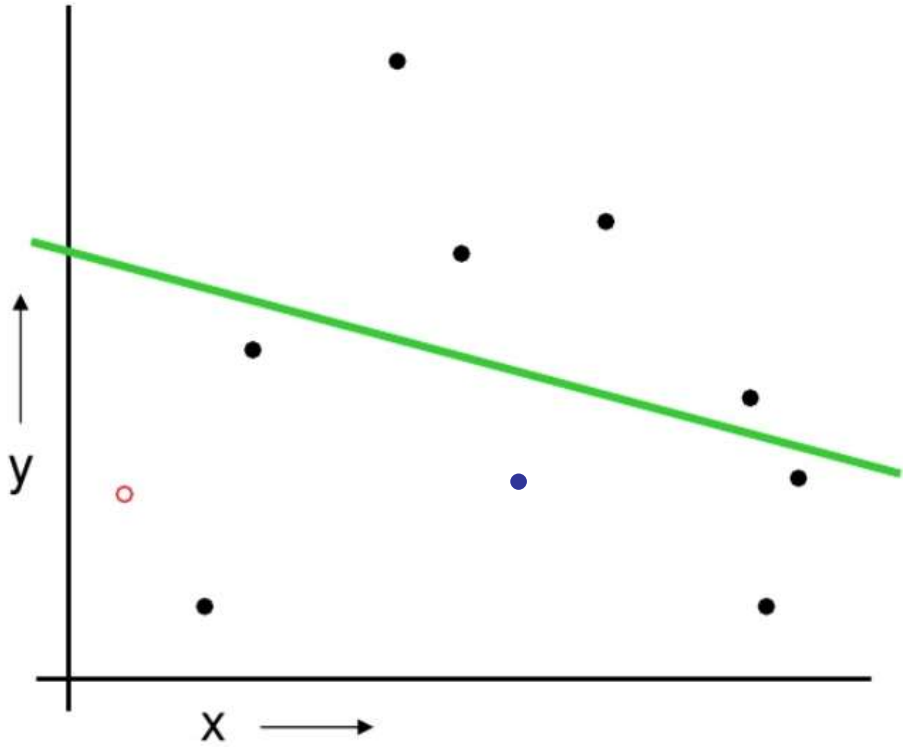
- Für k=1 bis n Datenpunkte
  1. Wähle (x_k, y_k) aus
  2. Entferne Datenpunkt (x_k, y_k) temporär aus Dataset
  3. Trainiere anhand der verbliebenen Datenpunkten

LOOCV- Leave-one-out Cross Validation



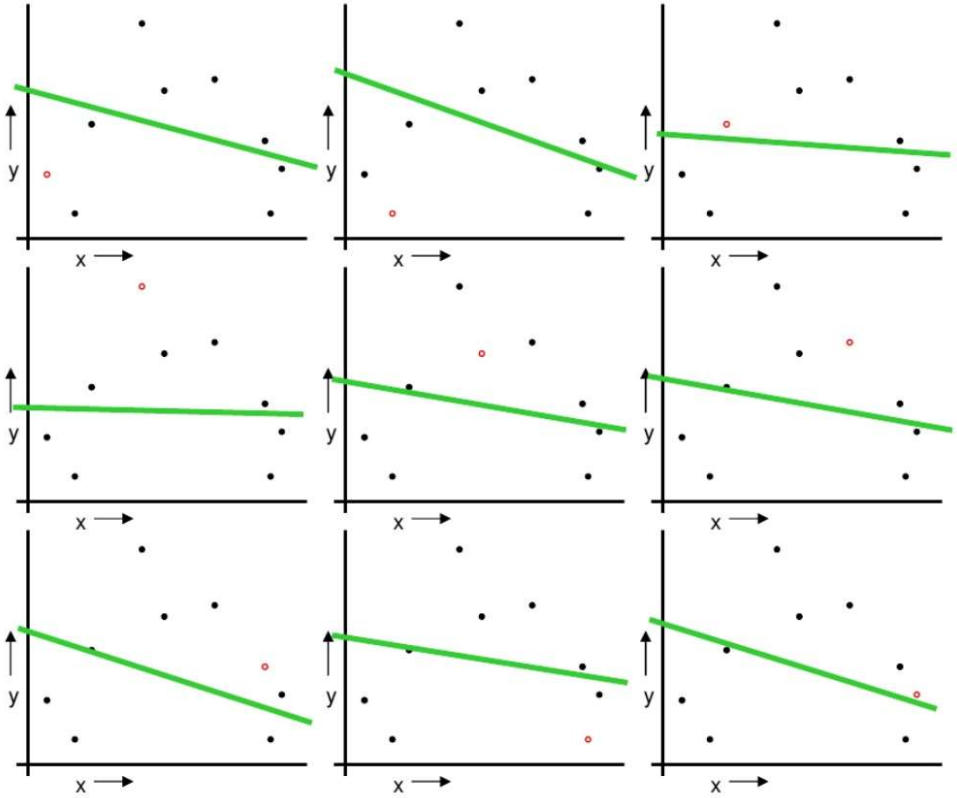
- Für k=1 bis n Datenpunkte
  1. Wähle (x_k, y_k) aus
  2. Entferne Datenpunkt (x_k, y_k) temporär aus Dataset
  3. Trainiere anhand der verbliebenen Datenpunkten
  4. Berechne Fehler von (x_k, y_k)

LOOCV- Leave-one-out Cross Validation



- Für k=1 bis n Datenpunkte
  1. Wähle (x_k, y_k) aus
  2. Entferne Datenpunkt (x_k, y_k) temporär aus Dataset
  3. Trainiere anhand der verbliebenen Datenpunkten
  4. Berechne Fehler von (x_k, y_k)
  5. Wiederhole Vorgang für alle Punkte

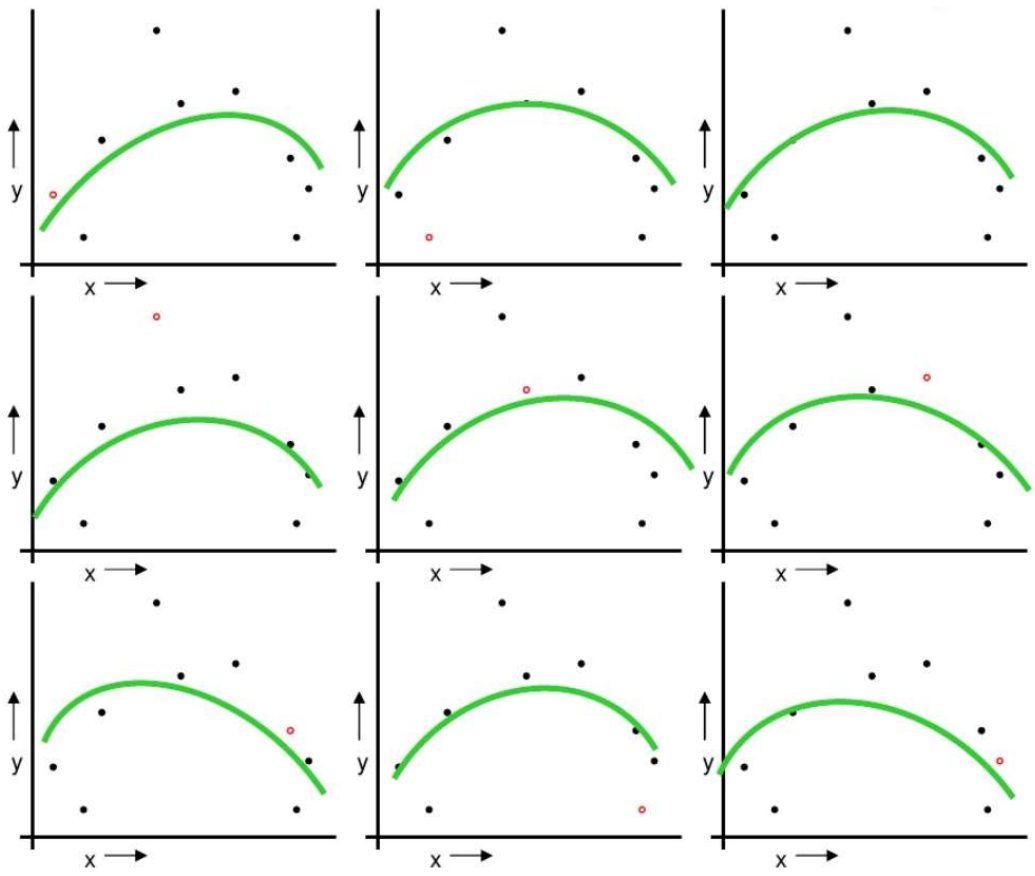
LOOCV- Leave-one-out Cross Validation



- Für k=1 bis n Datenpunkte
  1. Wähle (x_k, y_k) aus
  2. Entferne Datenpunkt (x_k, y_k) temporär aus Dataset
  3. Trainiere anhand der verbliebenen Datenpunkten
  4. Berechne Fehler von (x_k, y_k)
  5. Wiederhole Vorgang für alle Punkte
  6. Bilde durchschnittlichen Fehler für alle Punkte

     $MSE_{LOOCV} = 2.12$

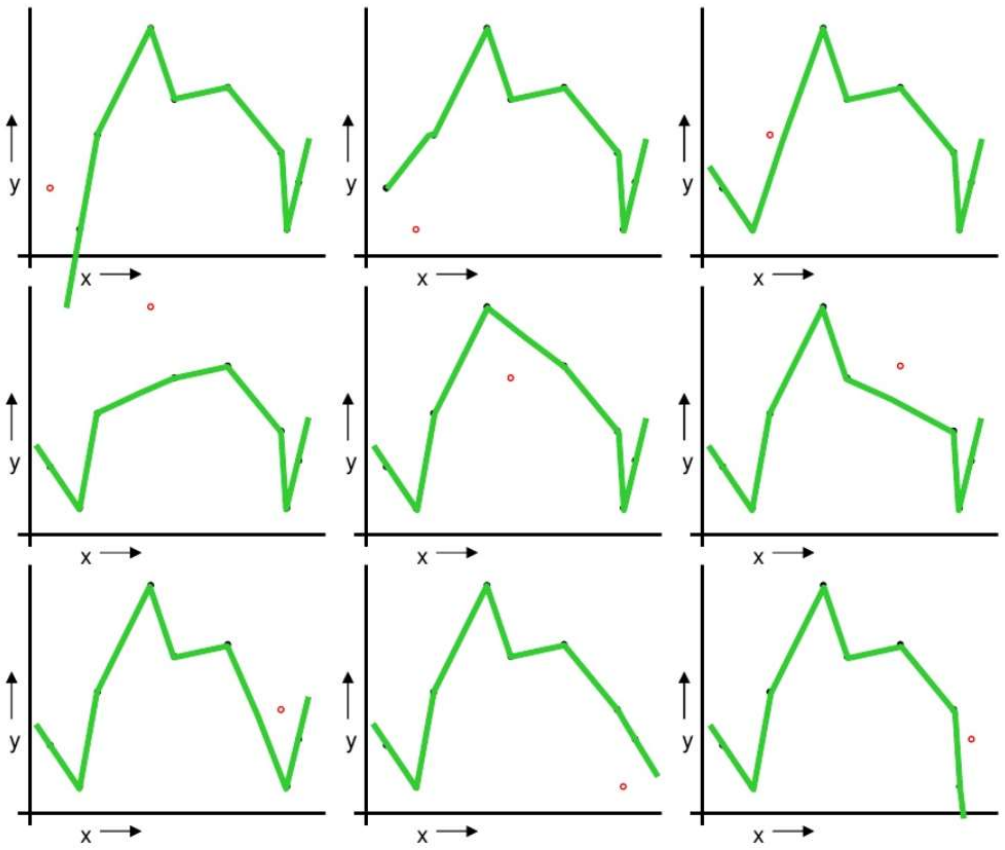
LOOCV- Leave-one-out Cross Validation



- Für k=1 bis #Datenpunkte
  1. Wähle (x_k, y_k) aus
  2. Entferne Datenpunkt (x_k, y_k) temporär aus Dataset
  3. Trainiere anhand der verbliebenen Datenpunkten
  4. Berechne Fehler von (x_k, y_k)
  5. Wiederhole Vorgang für alle Punkte
  6. Bilde durchschnittlichen Fehler für alle Punkte

  $MSE_{LOOCV} = 0.962$

LOOCV- Leave-one-out Cross Validation



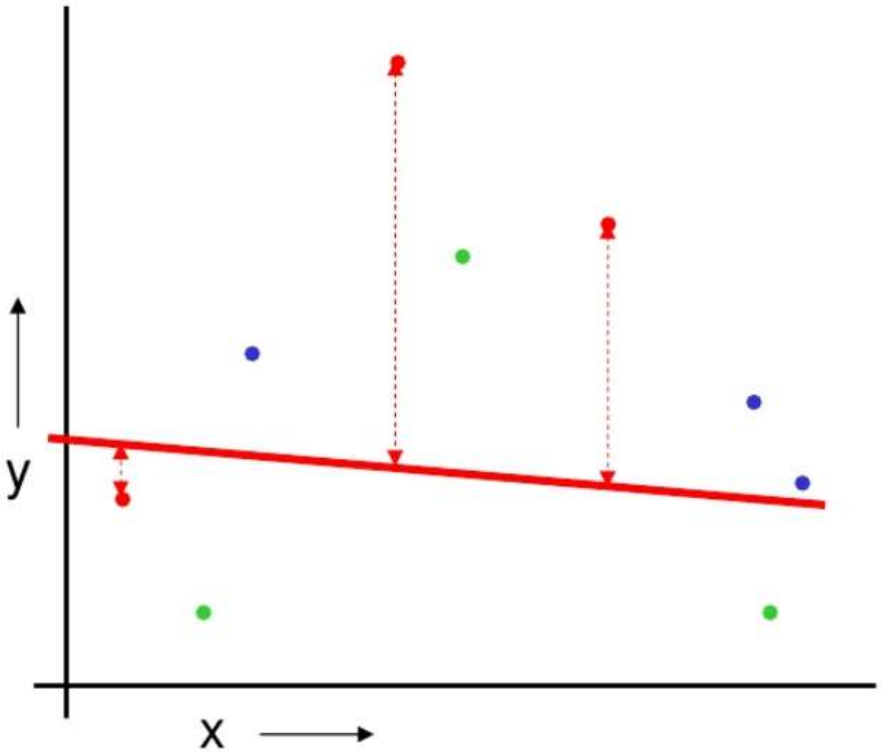
- Für k=1 bis #Datenpunkte
  1. Wähle (x_k, y_k) aus
  2. Entferne Datenpunkt (x_k, y_k) temporär aus Dataset
  3. Trainiere anhand der verbliebenen Datenpunkten
  4. Berechne Fehler von (x_k, y_k)
  5. Wiederhole Vorgang für alle Punkte
  6. Bilde durchschnittlichen Fehler für alle Punkte

  $MSE_{LOOCV} = 3.33$

k-fold Cross Validation

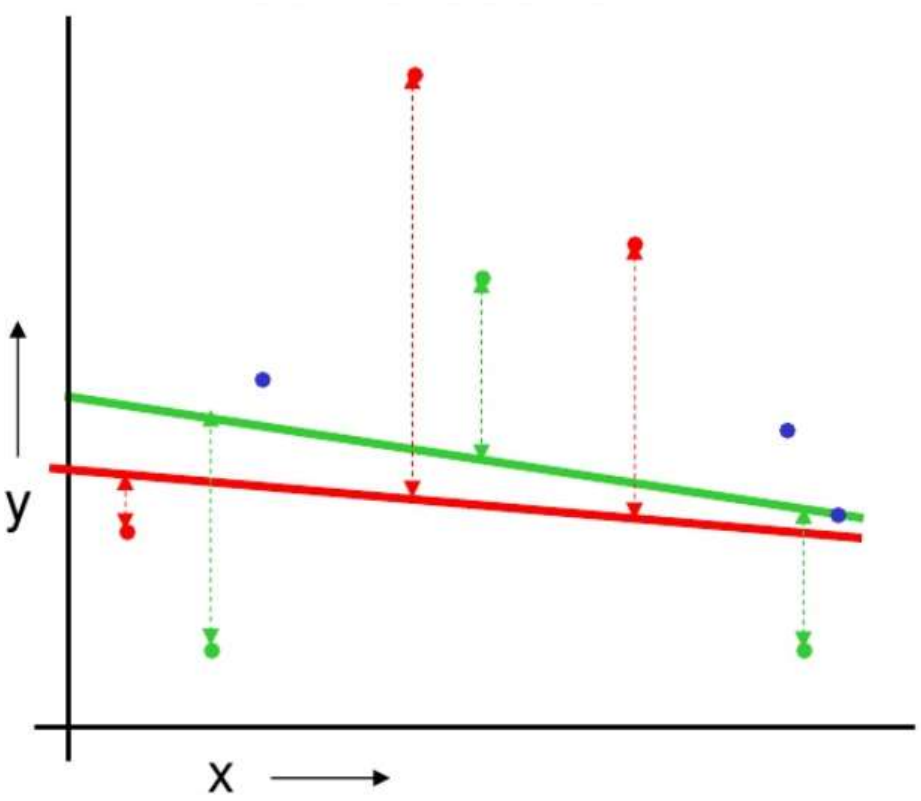
- Zur Bewertung eines Machine Learning Models wird normalerweise der Datensatz (zufällig) in Trainings- und Test-Daten aufgeteilt.
  - Man erstellt das Modell anhand von Trainingsdaten und bewertet das Modell anhand der Testdaten.
  - Allerdings ist dieses Vorgehen nicht sehr verlässlich, da es nur gegen einen (1!) Testdatensatz validiert wird.
  - K-fold Cross Validierung löst dieses Problem indem es den Datensatz in k Teile (*folds*) aufteilt und jeder Fold einmal zum Testdatensatz wird.
- Es wird im folgenden 3-fold cross validation vorgestellt, d.h. 2 folds sind Trainingsdaten, 1 fold sind Testdaten.
- Diese Aufteilung wird im folgenden für das Training von 3 Modellen verwendet.
  - Es wird ein Modell durch lineare Regression erstellt und dann der MSE (mean squared error) über alle Folds für dieses Modell berechnet
  - Dann wird ein Modell durch quadratische Regression erstellt und auch der MSE berechnet
  - Dann wird „Connect the dots“ angewandt
- Ziel:
  - ► Man hat 3 Modelle erstellt und diese mithilfe des k-fold cross validation validiert.
  - ► Man ist sich also sicher(er) dass, das jeweilige Modell generell gute Ergebnisse liefert.

k-fold Cross Validation



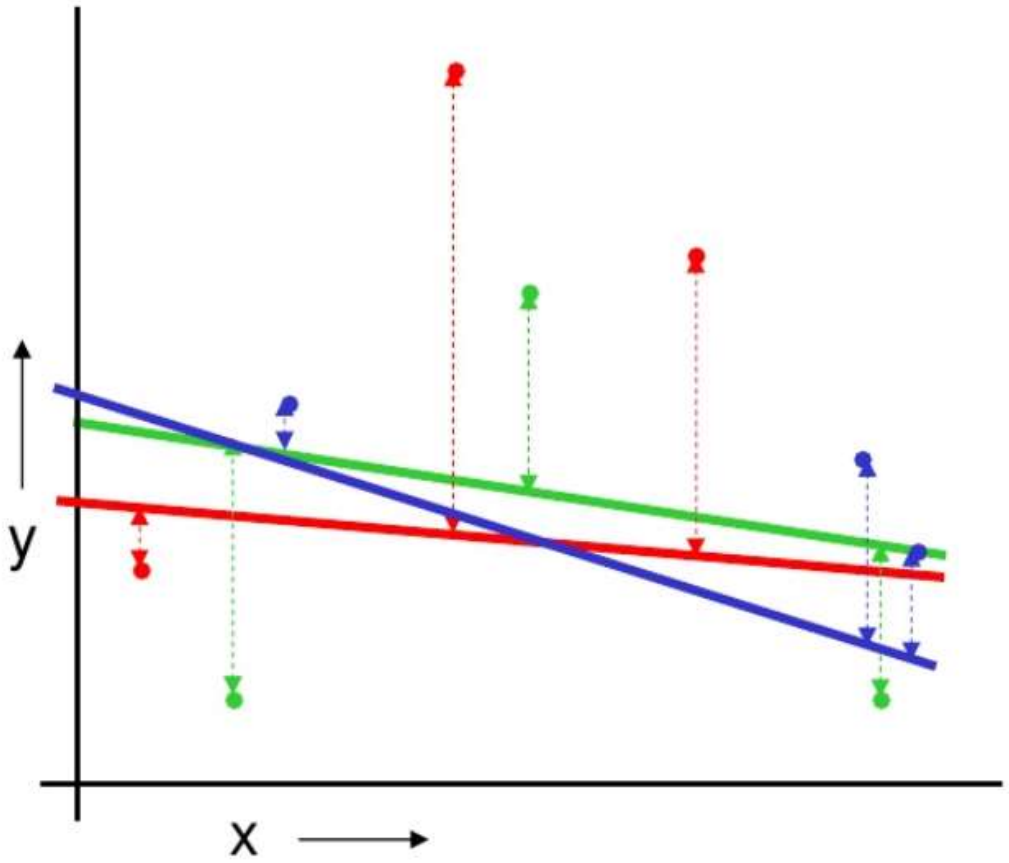
- Teile den Datensatz zufällig in k gleiche Partitionen auf (hier: k=3, rot, blau, grün)
- Rote Partition: Trainiere anhand der Datenpunkte, die in blauer und grüner Partition sind und berechne Fehler der Regression zu roten Punkten

k-fold Cross Validation



- Teile den Datensatz zufällig in k gleiche Partitionen auf (hier: k=3, rot, blau, grün)
- Rote Partition: Trainiere anhand der Datenpunkte, die in blauer und grüner Partition sind und berechne Fehler der Regression zu roten Punkten
- Grüne Partition: Trainiere anhand der Datenpunkte, die in roter und blauer Partition sind und berechne Fehler der Regression zu grünen Punkten

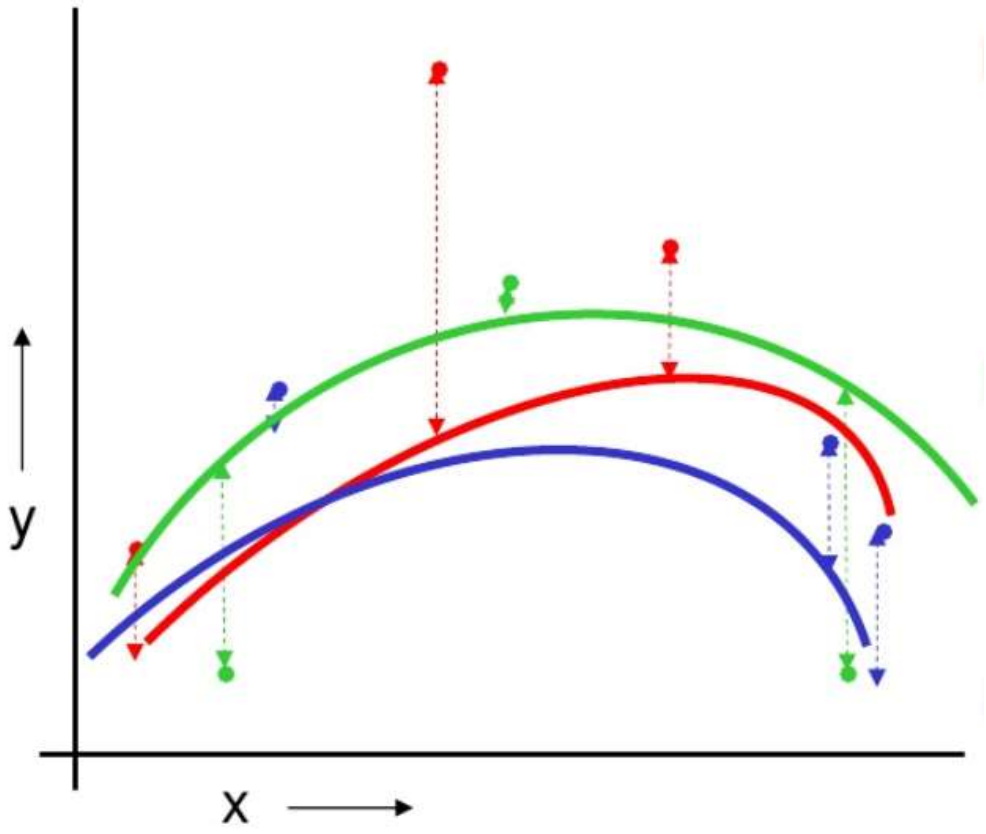
k-fold Cross Validation



- Teile den Datensatz zufällig in k gleiche Partitionen auf (hier: k=3, rot, blau, grün)
- Rote Partition: Trainiere anhand der Datenpunkte, die in blauer und grüner Partition und berechne Fehler der Regression zu roten Punkten
- Grüne Partition: Trainiere anhand der Datenpunkte, die in roter und blauer Partition sind und berechne Fehler der Regression zu grünen Punkten
- Blaue Partition: Trainiere anhand der Datenpunkte, die in roter und grüner Partition sind und berechne Fehler der Regression zu blauen Punkten
- Berechne durchschnittlichen Fehler:
  - Lineare Regression
  - MSE_{3Fold}= 2.05 (mean squared error)

k-fold Cross Validation

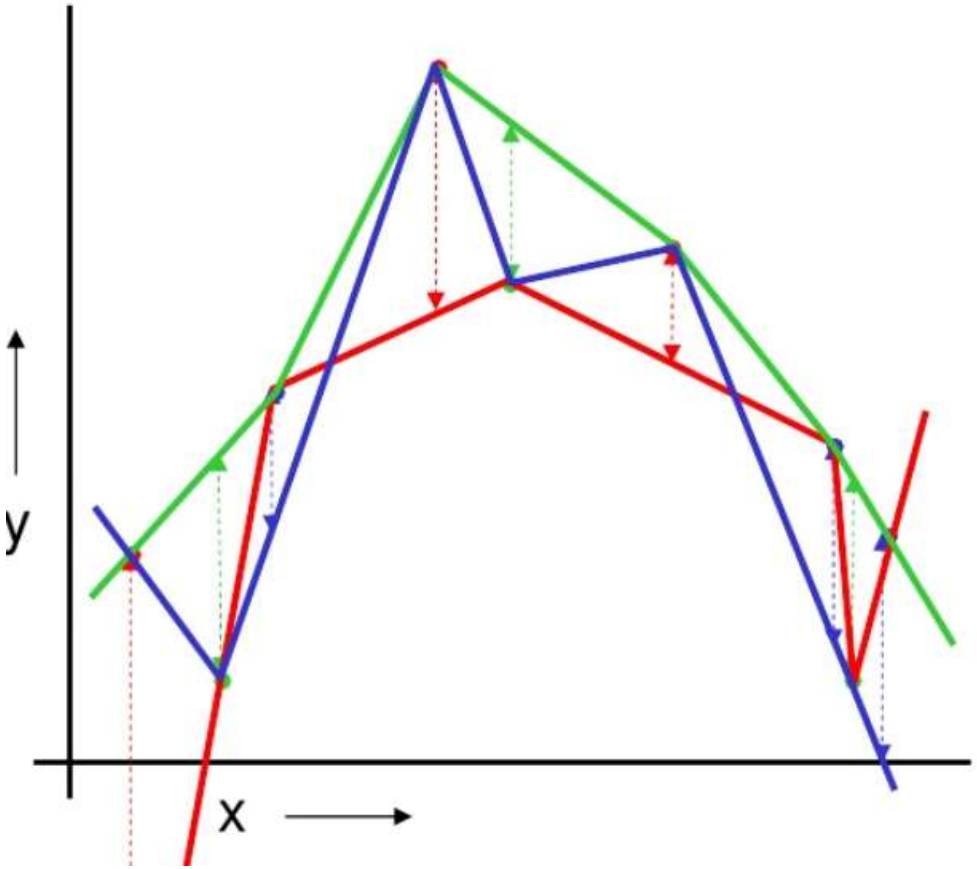
Quadratische Regression



- Teile den Datensatz zufällig in k gleiche Partitionen auf (hier: k=3, rot, blau, grün)
- Rote Partition: Trainiere anhand der Datenpunkte, die in blauer und grüner Partition und berechne Fehler der Regression zu roten Punkten
- Grüne Partition: Trainiere anhand der Datenpunkte, die in roter und blauer Partition sind und berechne Fehler der Regression zu grünen Punkten
- Blaue Partition: Trainiere anhand der Datenpunkte, die in roter und grüner Partition sind und berechne Fehler der Regression zu blauen Punkten
- Berechne durchschnittlichen Fehler:
  - Quadratische Regression
  - MSE_{3Fold}= 1.11 (mean squared error)

k-fold Cross Validation

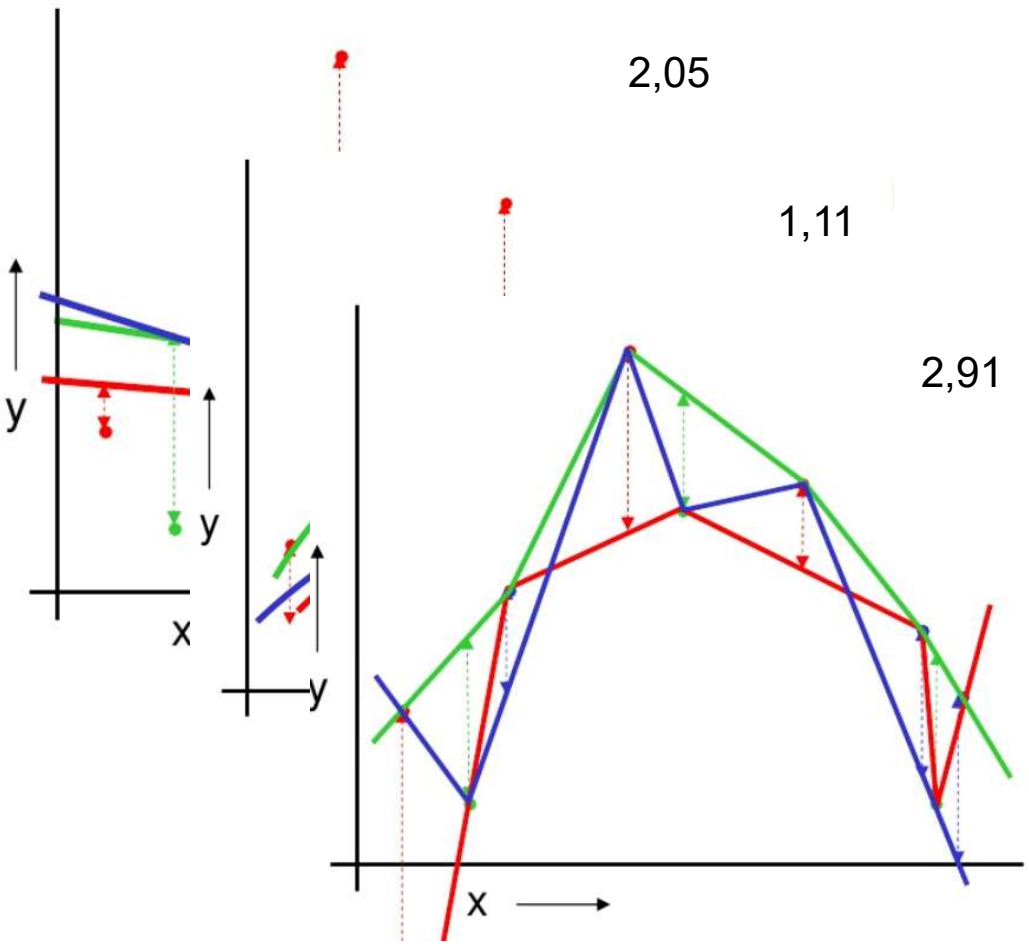
Connect the dots



- Teile den Datensatz zufällig in k gleiche Partitionen auf (hier: k=3, rot, blau, grün)
- Rote Partition: Trainiere anhand der Datenpunkte, die in blauer und grüner Partition und berechne Fehler der Regression zu roten Punkten
- Grüne Partition: Trainiere anhand der Datenpunkte, die in roter und blauer Partition sind und berechne Fehler der Regression zu grünen Punkten
- Blaue Partition: Trainiere anhand der Datenpunkte, die in roter und grüner Partition sind und berechne Fehler der Regression zu blauen Punkten
- Berechne durchschnittlichen Fehler:
  - „Connect the dots“ Interpolation
  - MSE_{3Fold}= 2.91 (mean squared error)

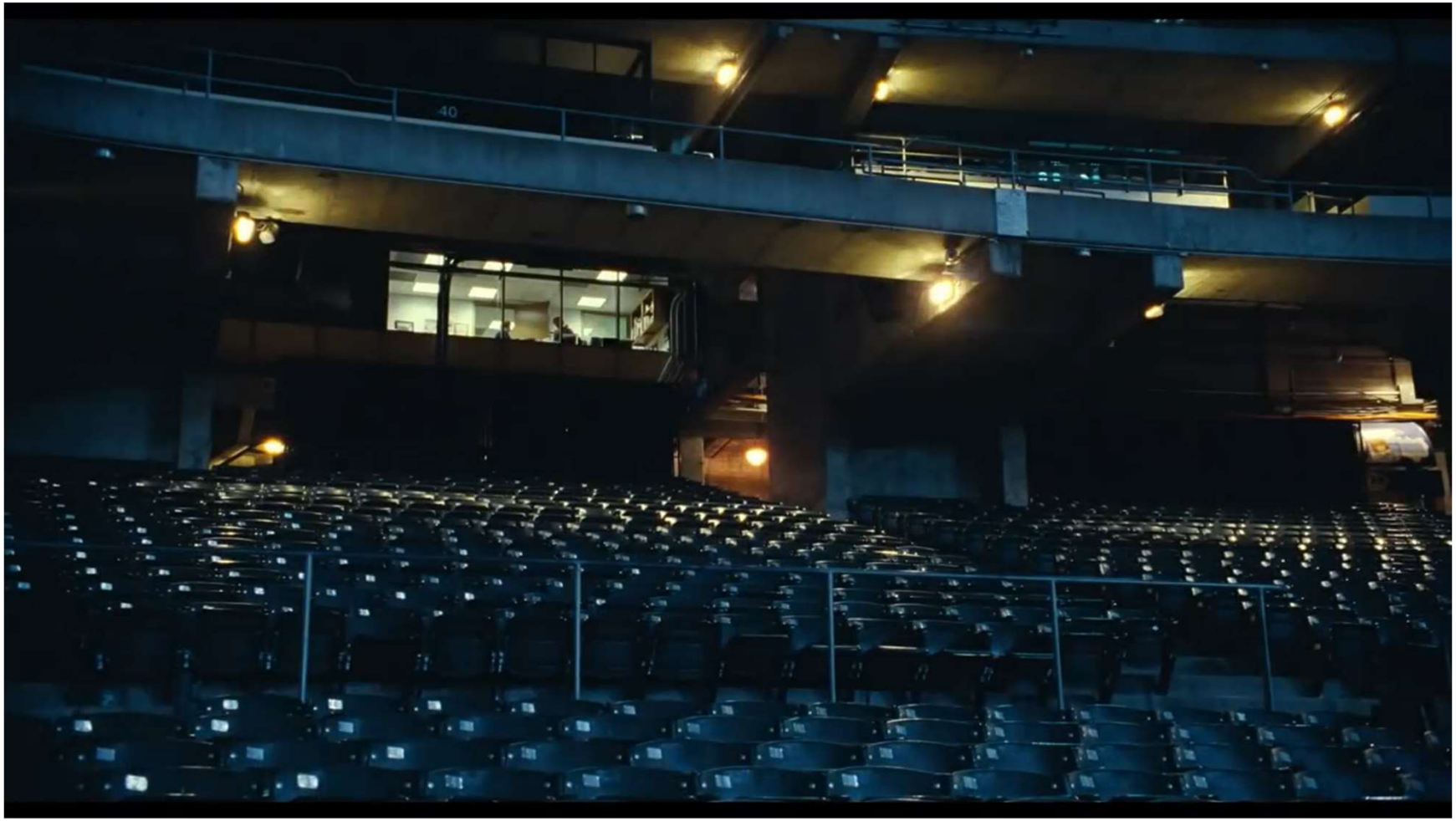
k-fold Cross Validation

Methodenauswahl



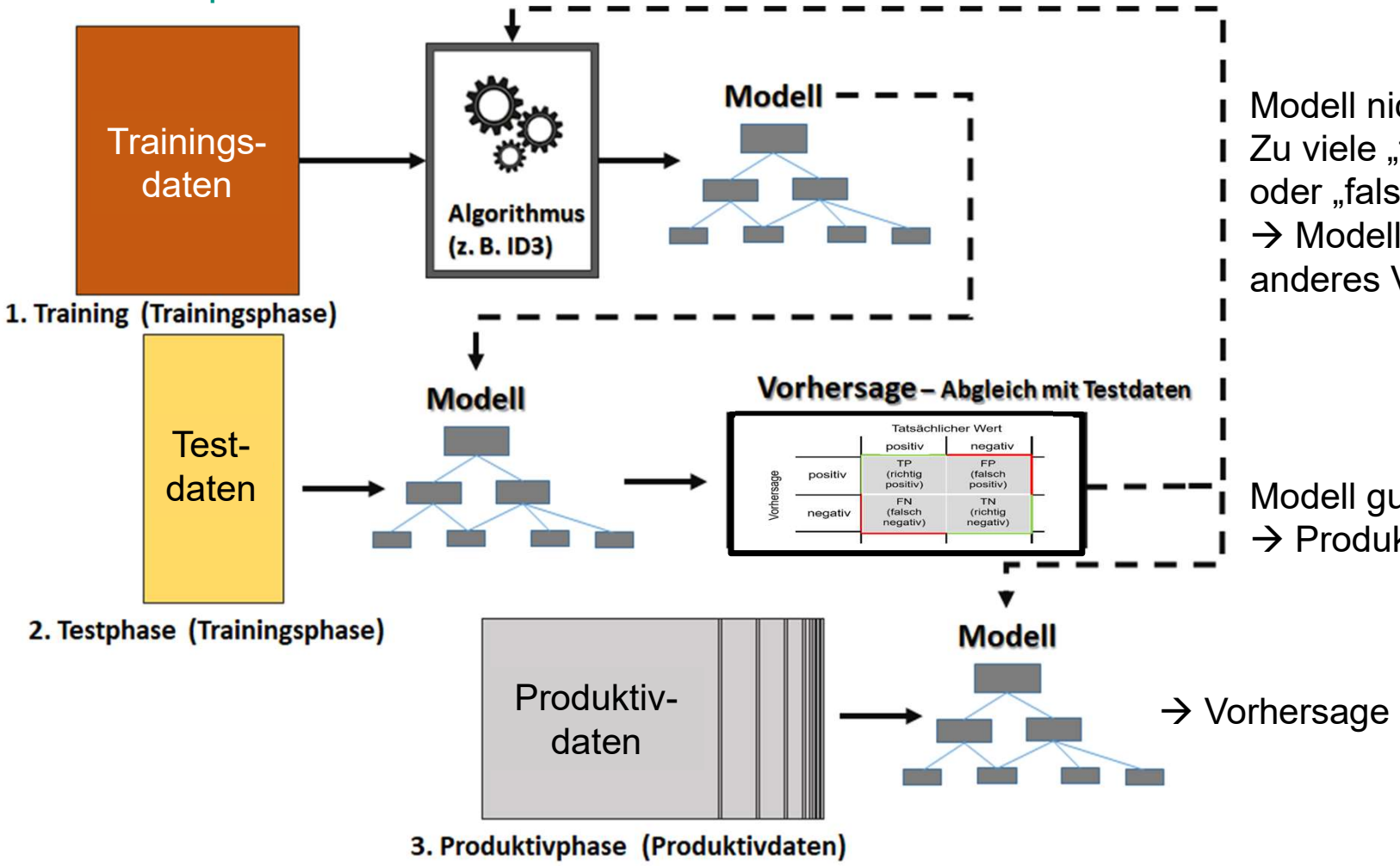
- Am Ende vergleicht man alle mittleren MSE der unterschiedlichen Methoden und entscheidet sich für die Methode mit dem kleinsten mittleren MSE (im Beispiel hier: quadratische Regression)
- Man bildet nun ein neues Modell mit allen Datenpunkten als Trainingsdatensatz.
- Im Idealfall hat man nun noch „unangetastete“ Daten zum Testen oder gar Validieren des endgültigen Modells.
- Es geht also letztlich um die Methodenauswahl.

Moneyball



Test trainierter Modelle

Train/Test-Split



Modell nicht gut?
Zu viele „falsch negativ“
oder „falsch positiv“
→ Modell verbessern,
anderes Verfahren etc.

Modell gut?
→ Produktiveinsatz

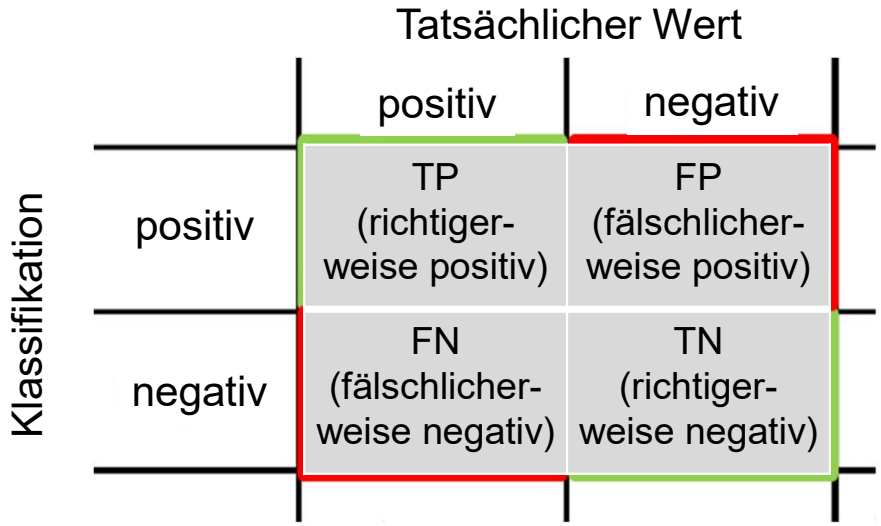
→ Vorhersage

Konfusionsmatrix

Es werden 4 Kategorien für die Bewertung der Klassifikationsgüte eingeführt

- Richtig positiv (engl.: true positives):
  - Das Endprodukt entspricht den Qualitätsansprüchen, und das Klassifikationsmodell hat dies korrekt mit gut klassifiziert/vorhergesagt.
- Richtig negativ (engl.: true negatives):
  - Das Endprodukt entspricht nicht den Qualitätsansprüchen, und das Klassifikationsmodell hat dies korrekt mit schlecht klassifiziert/vorhergesagt.

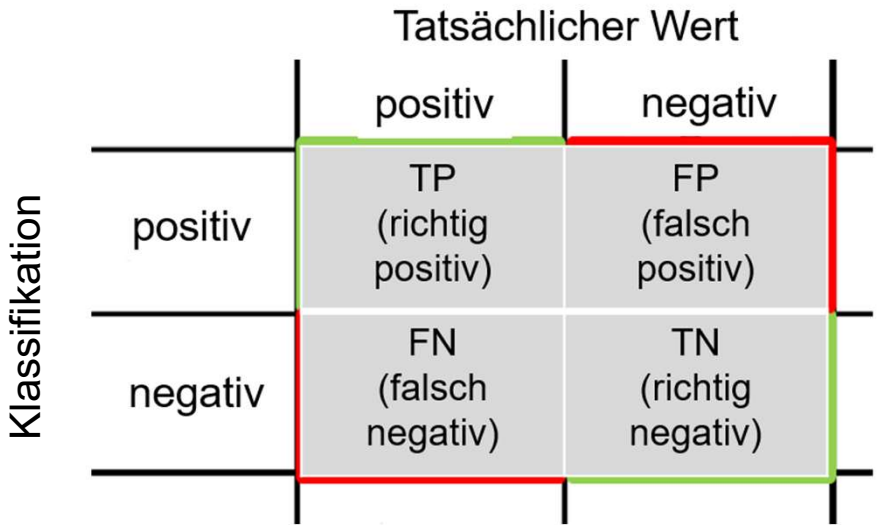
- Falsch positiv (engl.: false positives):
  - Das Endprodukt entspricht nicht den Qualitätsansprüchen, aber das Klassifikationsmodell hat dies fälschlich mit gut klassifiziert.
- Falsch negativ (engl.: false negatives):
  - Das Endprodukt entspricht den Qualitätsansprüchen, aber das Klassifikationsmodell hat dies fälschlich mit schlecht klassifiziert.



Konfusionsmatrix

Es werden 4 Kategorien für die Bewertung der Klassifikationsgüte eingeführt

- Diese vier Bewertungskategorien werden in einer Konfusionsmatrix (engl.: *confusion matrix*) dargestellt.
  - Die Konfusionsmatrix besteht aus den Spalten, die das Klassifikationsmodell vorhersagt.
  - Die Zeilen repräsentieren die tatsächlichen Werte, die in den Testdaten vorhanden sind.
  - Die Zellen «richtig positiv» (TP) und «richtig negativ» (TN) geben die Anzahl der richtig vorhergesagten Gut- und Schlecht-Werte an.
  - Die Zellen «falsch positiv» (FP) und «falsch negativ» (FN) geben die Anzahl der falschen Vorhersagen an
- Werden TP und FN miteinander summiert, so ergibt die Summe die Anzahl der tatsächlich verfügbaren Gut-Werte in dem Testdatensatz.
- Gleiches gilt für FP und TN, und die Summe entspricht der Anzahl der Schlecht-Werte im Testdatensatz.



Konfusions-Matrix

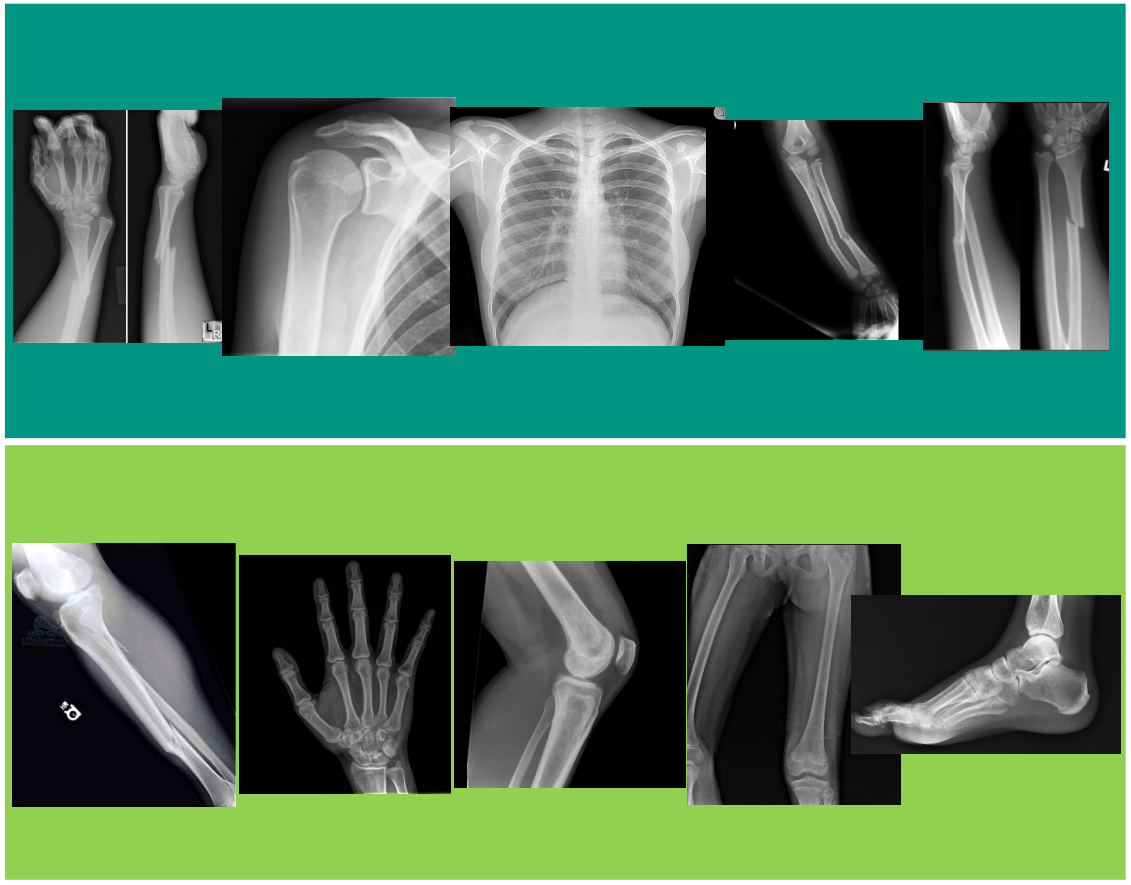
Beispiel: Ziel ist es die gebrochenen von den gesunden Knochen zu unterscheiden.

Knochenbruch-Datenbank





Vorhergesagte Situation



Konfusions-Matrix

Beispiel: Jetzt schaut der Arzt drauf.

Knochenbruch-Datenbank





Vorhergesagte Situation

Wahre Situation



	gebrochen	gesund
gebrochen	 3 True Positives (richtig erkannt)	2 False Positives (fälschlicherweise gebrochen → Fehldiagnose)
gesund	1 False Negative (fälschlicherweise gesund → Fehldiagnose	 4 True Negatives (richtig erkannt

Konfusions-Matrix

Knochenbruch-Datenbank



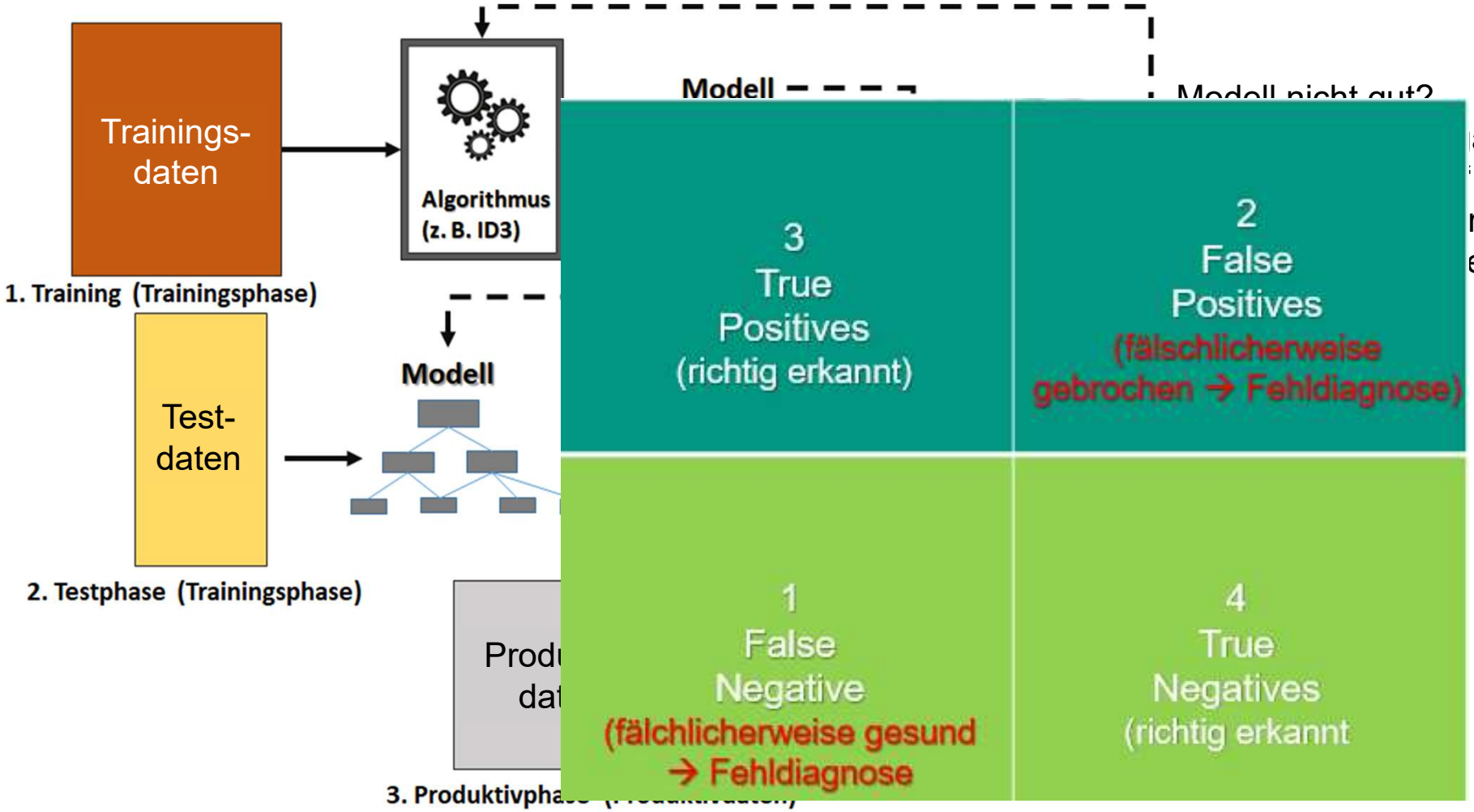


Vorhergesagte Situation

Wahre Situation



Vorhergesagte Situation \ Wahre Situation	gebrochen	gesund
gebrochen	3 True Positives (richtig erkannt)	2 False Positives (fälschlicherweise gebrochen → Fehldiagnose)
gesund	1 False Negative (fälschlicherweise gesund → Fehldiagnose	4 True Negatives (richtig erkannt


Trainings-
daten
Algorithmus
(z. B. ID3)
Modell
1. Training (Trainingsphase)
Test-
daten
Modell
2. Testphase (Trainingsphase)
Produ
dat
3. Produktivpha
3
True
Positives
(richtig erkannt)
2
False
Positives
(fälschlicherweise
gebrochen → Fehldiagnose)
1
False
Negative
(fälschlicherweise gesund
→ Fehldiagnose
4
True
Negatives
(richtig erkannt

Metriken

- Die Richtig-positiv-Rate (auch Sensitivität, Empfindlichkeit oder Trefferquote) gibt den Anteil der korrekt als positiv klassifizierten Objekte an der Gesamtheit der tatsächlich positiven Objekte an (True Positive Rate TPR).
- Die Falsch-negativ-Rate (englisch *false negative rate* oder *miss rate*) gibt den Anteil der fälschlich als negativ klassifizierten Objekte an der Gesamtheit der tatsächlich positiven Objekte an (False negative rate FNR).
- Übrig bleiben
  - Specificity, Selectivity (True negative rate TNR).
  - Fall-out (False positive rate FPR).

$$\mathrm{TPR} = \frac{\sum \text{True positives}}{\sum \text{Real positive}}$$

$$\mathrm{FNR} = \frac{\sum \text{False negatives}}{\sum \text{Real positive}}$$

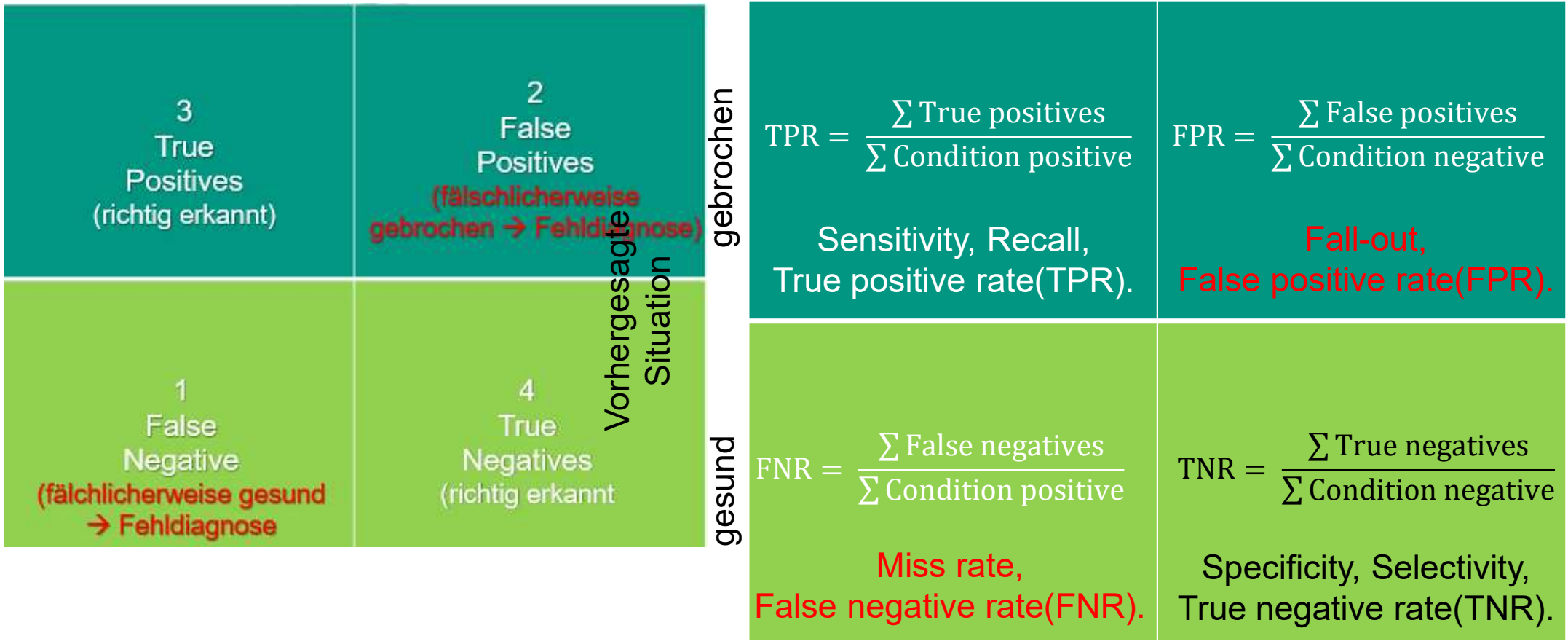
$$\mathrm{TNR} = \frac{\sum \text{True negatives}}{\sum \text{Real negative}}$$

$$\mathrm{FPR} = \frac{\sum \text{False positives}}{\sum \text{Real negative}}$$

- Da sich beide Maße auf den Fall beziehen, dass in Wirklichkeit die positive Kategorie vorliegt (erste Spalte der Wahrheitsmatrix), addieren sich die Sensitivität und die Falsch-negativ-Rate zu 1 bzw. 100 %.

Konfusions-Matrix

„Knochenbrüche“



Konfusions-Matrix

Beispiel „165 vermeintliche Knochenbrüche“

- Untersuchte Fälle: 165
- Wirklich gebrochen: 105
- Wirklich nicht gebrochen: 60
- Vorhergesagt gebrochen: 110
- Vorhergesagt nicht gebrochen: 55

Vorhergesagte Situation \ Wahre Situation	gebrochen	gesund
gebrochen	$\mathrm{TPR} = \frac{\sum \text{True positives}}{\sum \text{Condition positive}}$ Sensitivity, Recall, True positive rate(TPR).	$\mathrm{FPR} = \frac{\sum \text{False positives}}{\sum \text{Condition negative}}$ Fall-out, False positive rate(FPR).
gesund	$\mathrm{FNR} = \frac{\sum \text{False negatives}}{\sum \text{Condition positive}}$ Miss rate, False negative rate(FNR).	$\mathrm{TNR} = \frac{\sum \text{True negatives}}{\sum \text{Condition negative}}$ Specificity, Selectivity, True negative rate(TNR).

Konfusions-Matrix

Beispiel „Knochenbrüche“

- Untersuchte Fälle: 165
- Wirklich gebrochen: 105
- Wirklich nicht gebrochen: 60
- Vorhergesagt gebrochen: 110
- Vorhergesagt nicht gebrochen: 55

- TPR: 100/105 = 0,952
- FNR: 5/105 = 0,048

= 1

- FPR: 10/60 = 0,167
- TNR: 50/60 = 0,833

= 1

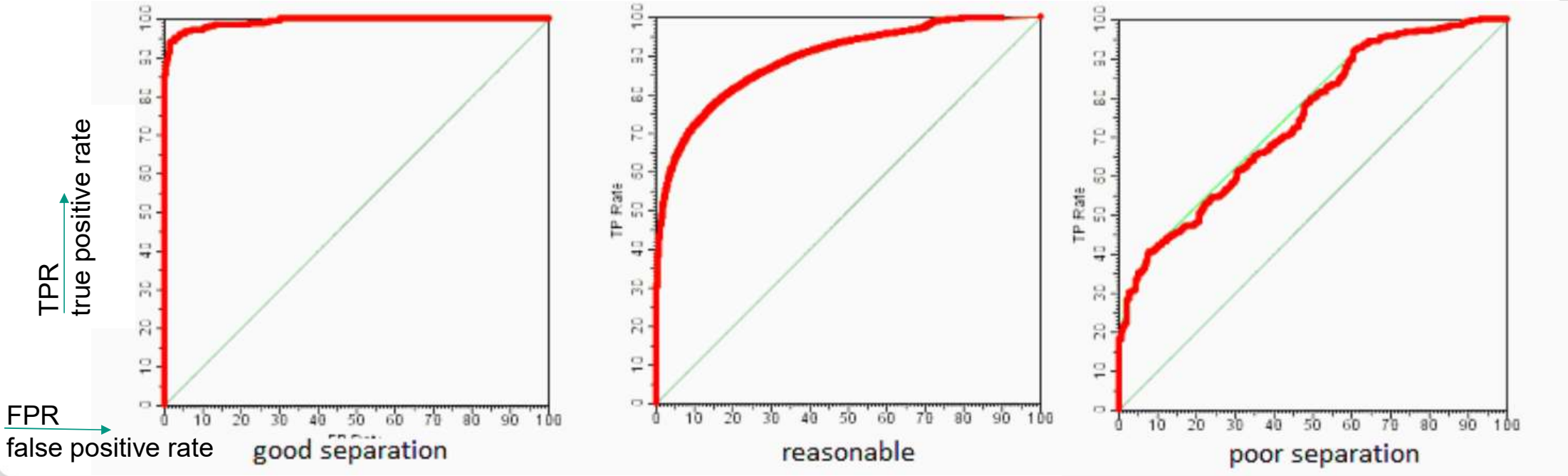
Vorhergesagte Situation \ Wahre Situation	gebrochen	gesund
gebrochen	$\text{TPR} = \frac{\sum \text{True positives}}{\sum \text{Condition positive}}$ 100	$\text{FPR} = \frac{\sum \text{False positives}}{\sum \text{Condition negative}}$ 10
gesund	$\text{FNR} = \frac{\sum \text{False negatives}}{\sum \text{Condition positive}}$ 5	$\text{TNR} = \frac{\sum \text{True negatives}}{\sum \text{Condition negative}}$ 50

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Gütemaße für die Evaluation von Modellen

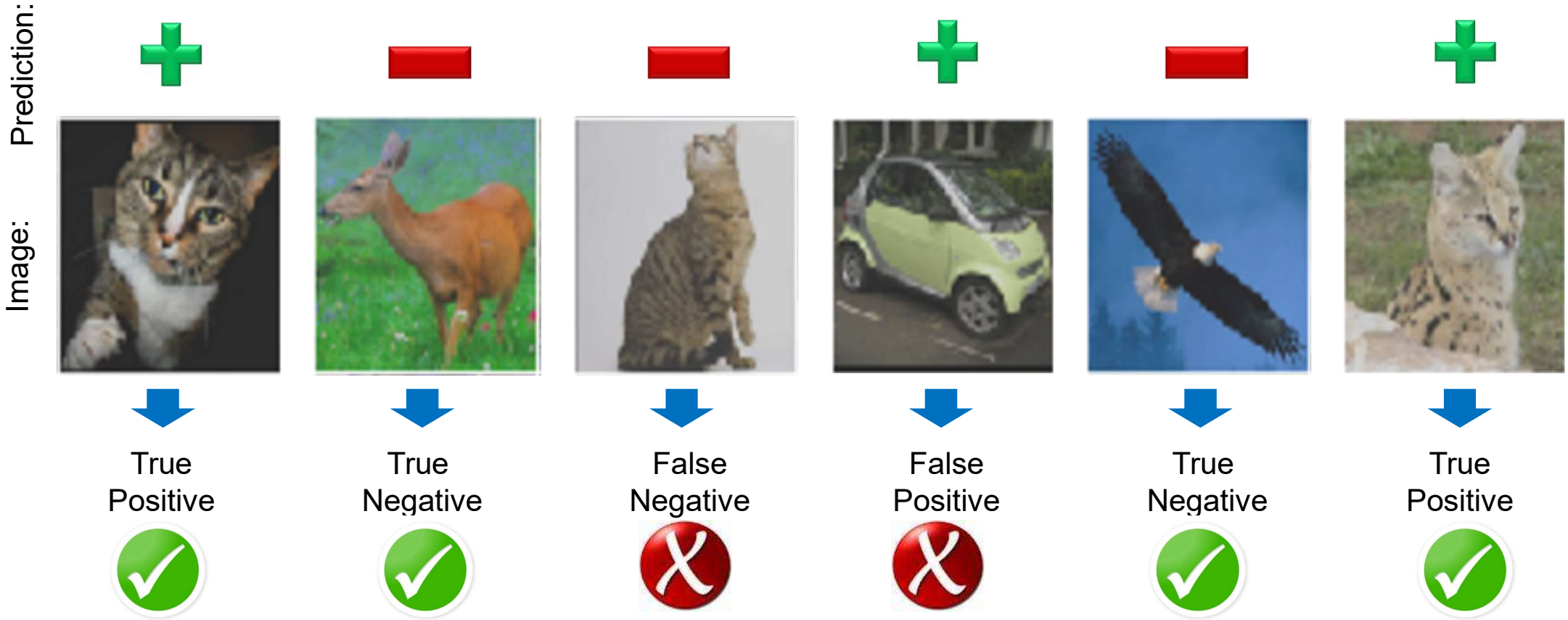
ROC (Receiver Operating Characterstic)

- Die **Receiver-Operating-Characteristic-Kurve** (**ROC-Kurve** o. Grenzwertoptimierungskurve) ist eine Methode zur Bewertung und Optimierung von Analyse-Strategien.
- Die **ROC-Kurve** stellt visuell die Abhängigkeit der Effizienz mit der Fehlerrate für verschiedene Parameterwerte dar.
- ROC plottet die *True Positives* (auf der y-Achse) gegen die *False Positives* (auf der x-Achse)



Gütemaße für die Evaluation von Modellen

Klassifikation am Beispiel „Katzen“



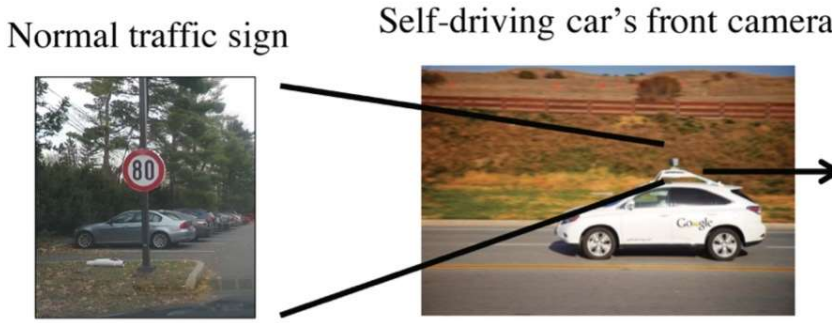
Images from the STL-10 dataset

Beispiel
„False Prediction“

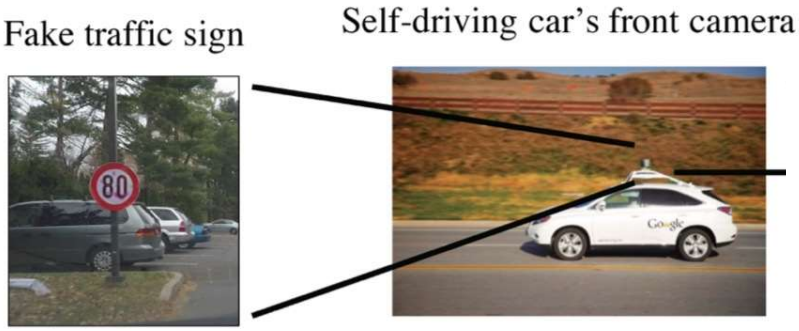
- Autonome Autos orientieren sich unter anderem ebenso wie menschliche Fahrer an Verkehrszeichen.
  - Dabei können sie in böswilliger Absicht in die Irre geführt werden, haben Forscher der Universitäten Princeton und Purdue herausgefunden.
  - Das von ihnen so genannte *Deceiving Autonomous caRs with Toxic Signs (DARTS)* könne zu katastrophalen Folgen führen.
- Augenscheinlich unverdächtig
  - Dabei arbeiteten die Forscher mit Verkehrs- oder anderen Schildern, die für normale Beobachter unverdächtig aussehen, aus der Sicht eines automatischen Bilderkennungsystems aber eine andere Bedeutung haben können.
  - So haben die Forscher beispielsweise das Verkehrszeichen für ein Tempolimit von 80 km/h mit Hilfe eines Musters aus dem ersten Anschein nach unverdächtigen Flecken so modifiziert, dass es ein Zeichenerkennungssystem als Stoppschild interpretieren würde.
  - Auf offener Strecke könnte ein autonomes Auto plötzlich stoppen und so einen Auffahrunfall provozieren.

Augenscheinlich unverdächtig

https://www.heise.de/newsticker/meldung/Autonome-Autos-Forscher-fuehren-Bilderkennung-mit-manipulierten-Schildern-in-die-Irre-3974483.html

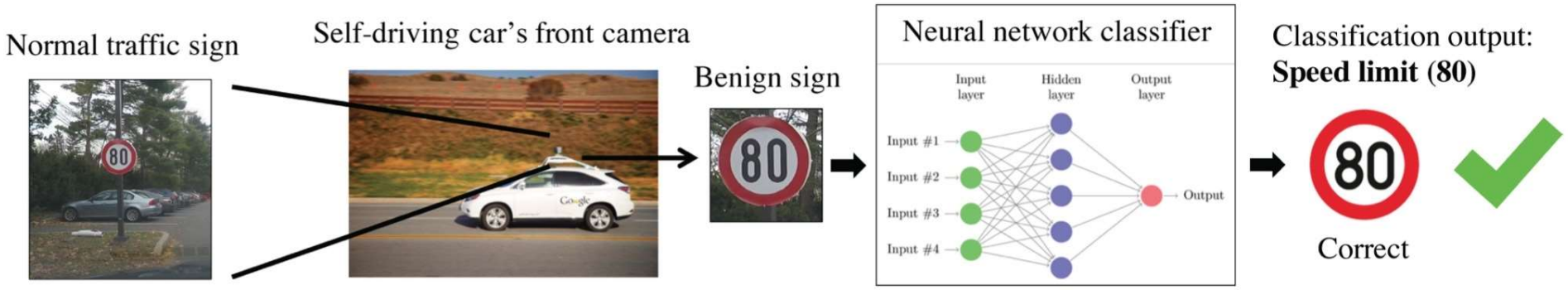


(a) Operation of the computer vision subsystem of an AV under *benign conditions*

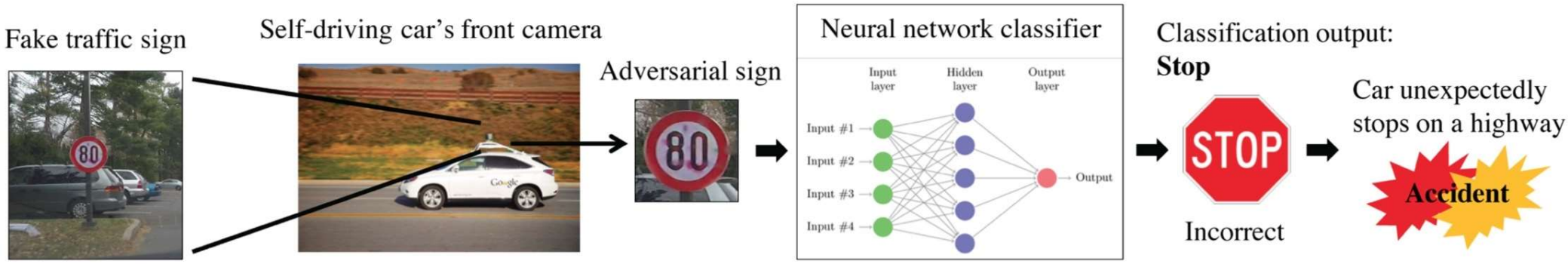


Augenscheinlich unverdächtig

https://www.heise.de/newsticker/meldung/Autonome-Autos-Forscher-fuehren-Bilderkennung-mit-manipulierten-Schildern-in-die-Irre-3974483.html

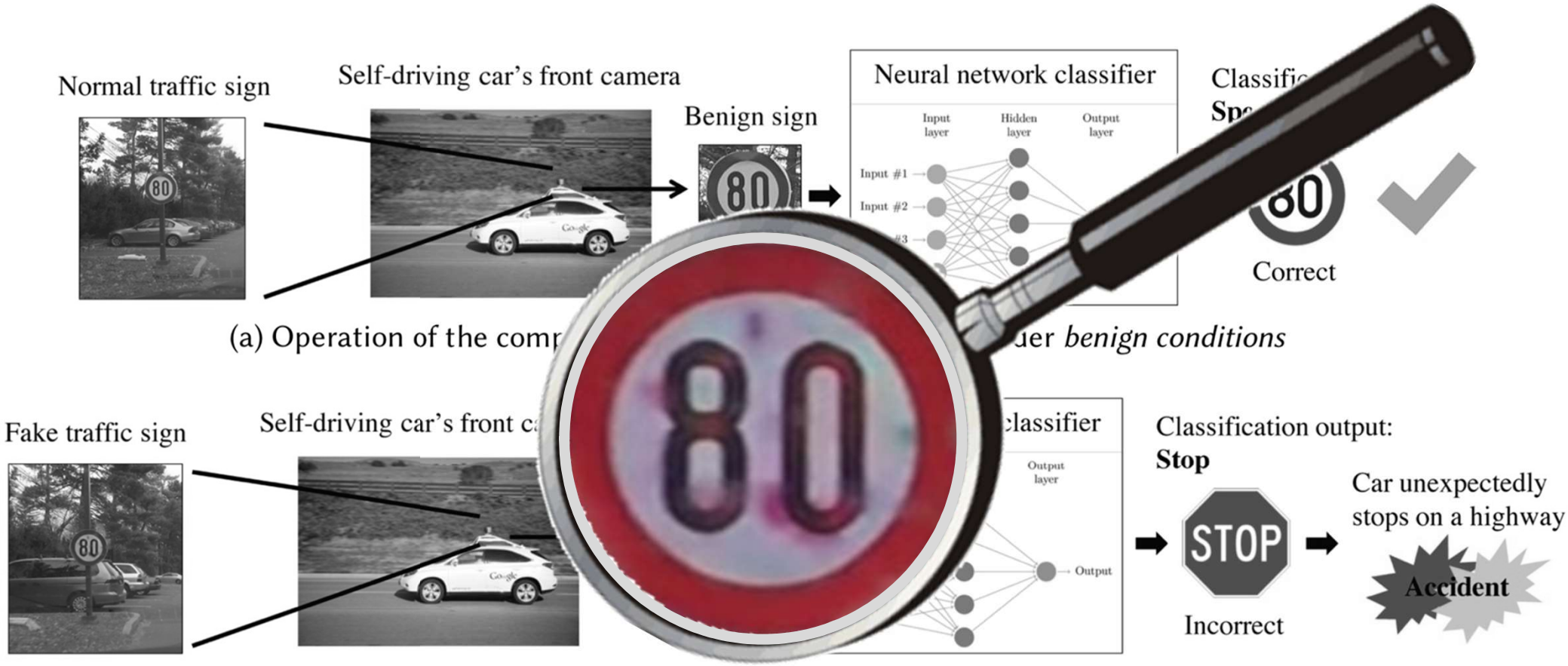


(a) Operation of the computer vision subsystem of an AV under *benign conditions*



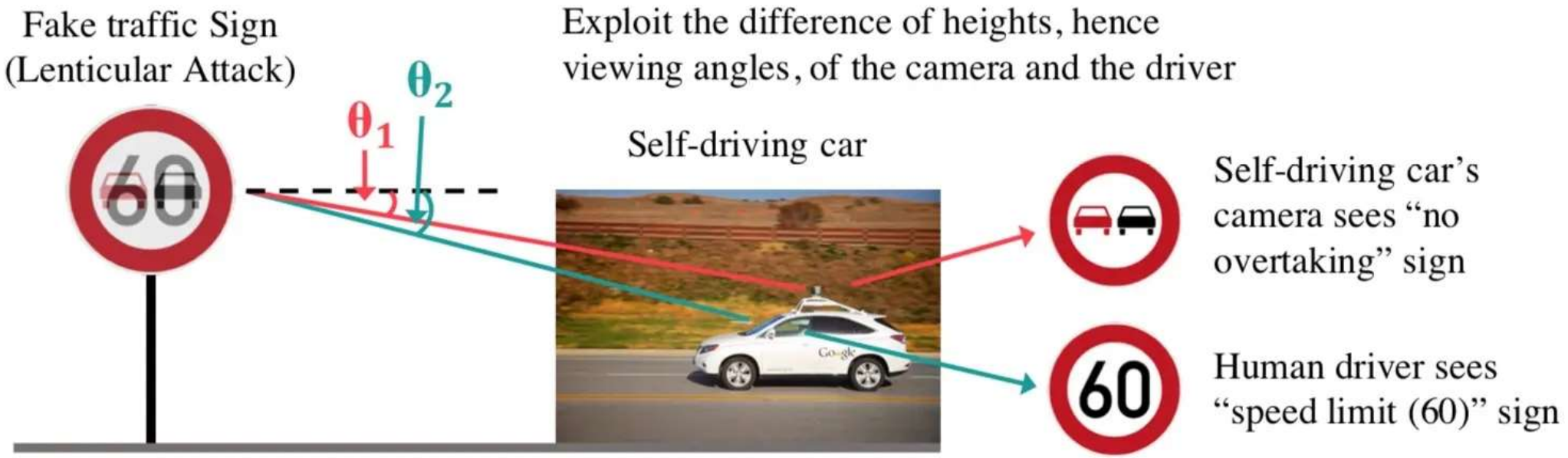
Augenscheinlich unverdächtig

https://www.heise.de/newsticker/meldung/Autonome-Autos-Forscher-fuehren-Bilderkennung-mit-manipulierten-Schildern-in-die-Irre-3974483.html



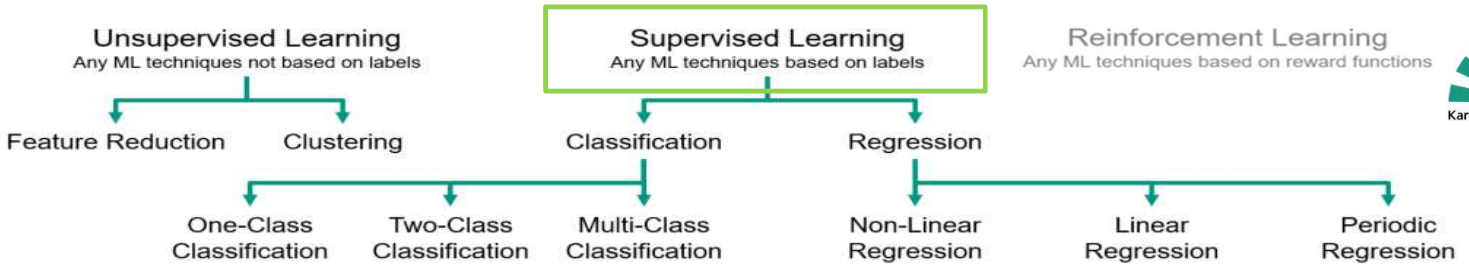
Überholverbot statt Tempolimit

Irreführung von Fahrer und Kamera mit Hilfe einer Linsenrasterfolie



(b) Principle behind an attack based on lenticular printing.

Machinelles Lernen



Algorithm ↓	Abk.	Feature Reduction	Clustering	One-Class Classification	Two-Class Classification	Multi-Class Classification	Non-Linear Regression	Linear Regression	Periodic Regression
Faktorenanalyse		X							
Principal Component Analysis	PCA	X		X					
	K-means		X						
hierarchical cluster analysis	HCA		X						
DBSCAN			X						
One Class Support Vector Machine	OCSVM			X					
Isolation Forest				X					
	LODA			X					
(künstliche) Neuronale Netze	NN			X (Autoencoder)	X	X	X	X	X
Support Vector Machine	SVM				X				
Decision Tree					X	X			
Bayes-Klassifikation					X	X			
Random Forest						X			
Diskriminanzanalyse				x	x	x			
Logistic Regression							X	X	
Linear Regression								X	
Harmonic Regression									X
Nächste-Nachbar-Klassifikation	K-NN	x	x						

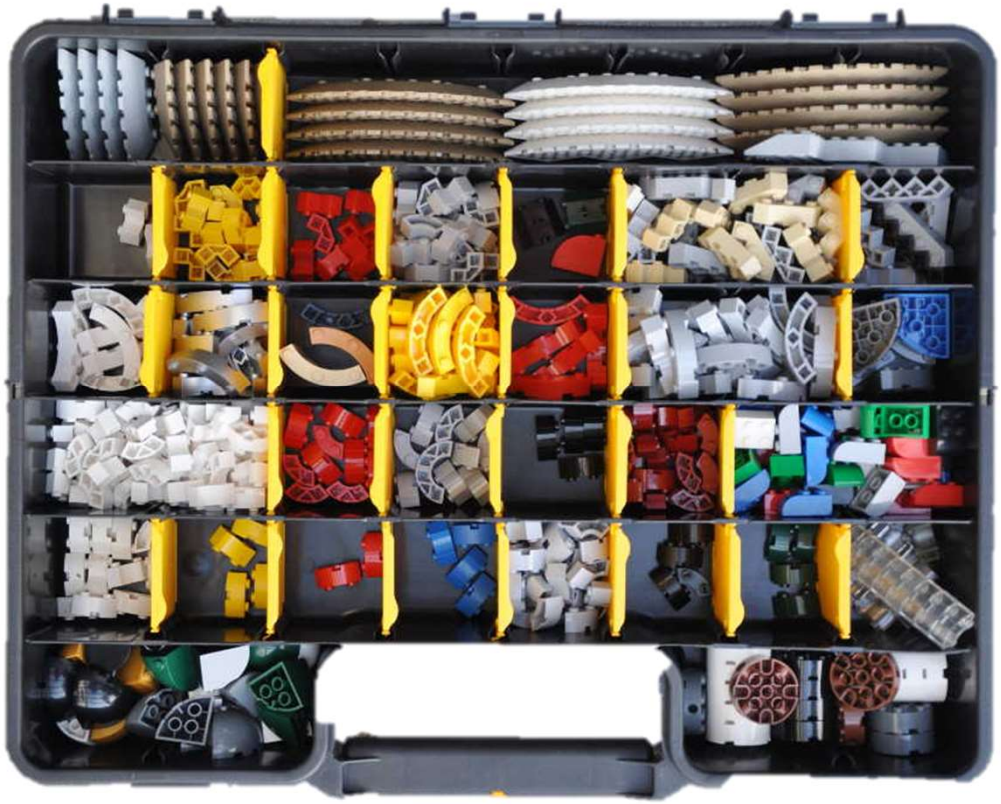
Klassifizierung – Algorithmen

- Im Gegensatz zu Regressionsproblemen erkennt man Klassifikationsprobleme daran, dass der Output y nur wenige diskrete Werte annehmen kann.
  - Meistens liegen diese Werte in qualitativer Form vor, beispielsweise, wenn es darum geht auf der Basis von mehreren erklärenden Variablen zu bestimmen, ob es sich bei einer E-Mail um Spam oder keinen Spam handelt.
  - In diesem Beispiel wären die erklärenden Variablen dann x und der Output y wäre 1, wenn es sich um eine Spam E-Mail handelt und 0, wenn keine Spam E-Mail vorliegt.
- Man unterscheidet zudem zwischen *one-class*, *two-class* und Klassifikationsproblemen, bei denen *multiple* Klassen vorliegen.
  - Ein Beispiel hierfür wäre zu klassifizieren von welcher Farben ein Bleistift ist.
  - Die Klassen sind in diesem Fall die rechts stehenden Farben.



Diskriminanzanalyse
Klassifikation

- Die Diskriminanzanalyse ist ein Verfahren zur Analyse von Gruppenunterschieden.
  - Es werden die „diskriminierenden“ Variablen erkannt, die für die Gruppenzugehörigkeit sorgen.
  - Dadurch kann anhand der vorliegenden Ausprägung der unabhängigen Variablen auf die abhängigen Variable geschlossen werden (z. B. bei der Kreditwürdigkeitsprüfung).
- Die Diskriminanzanalyse erstellt ein Vorhersagemodell für Gruppenzugehörigkeiten.



Diskriminanzanalyse
Klassifikation

- Das Vorhersagemodell besteht aus einer Diskriminanzfunktion (oder bei mehr als zwei Gruppen einem Set von Diskriminanzfunktionen), die auf der Grundlage derjenigen linearen Kombinationen der Prädiktorvariablen bestimmt wird, die die beste Diskriminanz zwischen den Gruppen ergeben.
  - *Die zu untersuchenden Variablen in einem Experiment (die gemessen und beobachtet werden) werden als Antwortvariablen bzw. abhängige Variablen bezeichnet.*
  - *Andere Variablen im Experiment, die die Antwortvariable beeinflussen und vom Versuchsleiter festgelegt oder gemessen werden können, werden als Prädiktorvariablen, erklärende Variablen oder unabhängige Variablen bezeichnet.*
- Die Funktionen werden aus einer Stichprobe der Fälle generiert, bei denen die Gruppenzugehörigkeit bekannt ist.
  - Diese Funktionen können dann auf neue Fälle mit Werten für die Prädiktorvariablen zur Bestimmung der Gruppenzugehörigkeit angewandt werden.

Diskriminanzanalyse

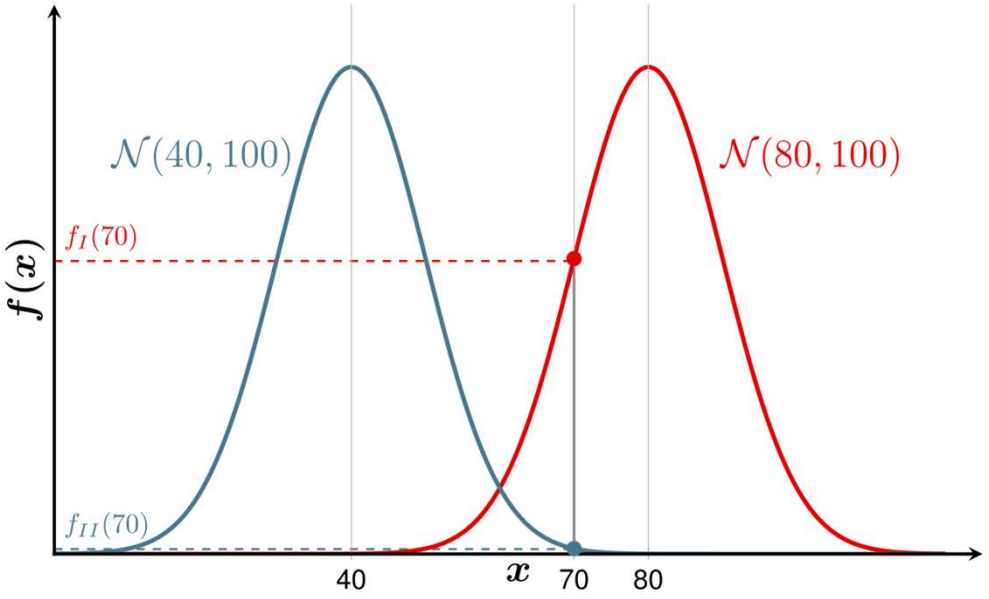
Beispiel

- Eine Gärtnerei hat die Möglichkeit, eine größere Menge Samen einer bestimmten Sorte Nelken günstig zu erwerben.
  - Um den Verdacht auszuräumen, dass es sich dabei um alte, überlagerte Samen handelt, wird eine Keimprobe gemacht.
  - Man sät also 1g Samen aus und zählt, wie viele dieser Samen keimen.
  - Aus Erfahrung ist bekannt,
    - dass die Zahl der keimenden Samen pro 1g Saatgut annähernd normalverteilt ist.
    - Bei frischem Saatgut (Population I) keimen im Durchschnitt 80 Samen, bei altem (Population II) sind es nur 40 Samen.

Diskriminanzanalyse

Beispiel: Set von Diskriminanzfunktionen durch Normalverteilung

- Eine Gärtnerei hat die Möglichkeit, eine größere Menge Samen einer bestimmten Sorte Nelken günstig zu erwerben.
  - Um den Verdacht auszuräumen, dass es sich dabei um alte, überlagerte Samen handelt, wird eine Keimprobe gemacht.
  - Man sät also 1g Samen aus und zählt, wie viele dieser Samen keimen.
  - Aus Erfahrung ist bekannt,
    - dass die Zahl der keimenden Samen pro 1g Saatgut annähernd normalverteilt ist.
    - Bei frischem Saatgut (Population I) keimen im Durchschnitt 80 Samen, bei altem (Population II) sind es nur 40 Samen.
- Die Keimprobe hat nun x=70 ergeben
- Die Grafik zeigt, dass bei dieser Probe die „Likelihood“ der Population I am größten ist.
  - ► Man ordnet also diese Keimprobe als *frisch* ein.

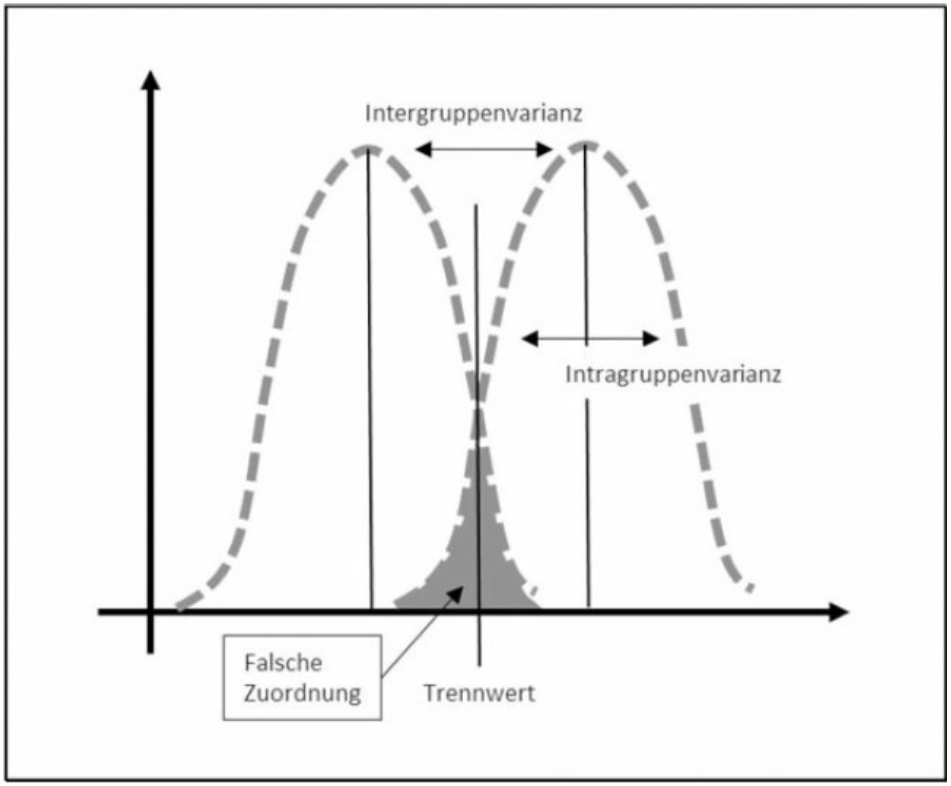


- Population I: Die Zahl der frischen Samen, die keimen, ist verteilt als $X_I \sim N(80; 10^2)$,
- Population II: Die Zahl der alten Samen, die keimen, ist verteilt als $X_{II} \sim N(40; 10^2)$,

- Der Schnittpunkt der Verteilungsdichten (bei x=60) entspricht der Entscheidungsgrenze.

Diskriminanzanalyse
Regeln

- Die Bildung der Diskriminanzfunktion erfolgt unter folgenden Bedingungen:
  - Die Varianz zwischen den Gruppenmittelwerten (Intergruppenvarianz) sollte möglichst groß sein.
  - Die Varianz innerhalb einer Gruppe (Intragruppenvarianz) sollte möglichst klein sein.
  - Die sich „überlappende“ Fläche (die eine Falschklassifikation bedeutet) sollte möglichst klein sein.

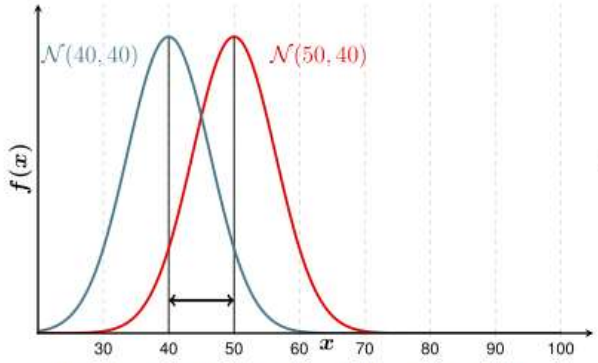


„Standardmodell“ der Diskriminanzanalyse

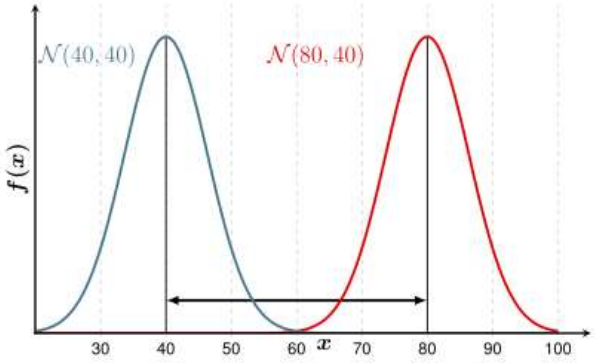
Es wird von gleichen Varianzen und Kovarianzen ausgegangen.

- Große Intergruppenvarianz

Die Varianz **zwischen** den Gruppenmittelwerten, die **Intergruppenvarianz**, sollte groß sein, weil sich dann die Verteilungen nicht durchmischen: Die Trennung der Gruppen ist schärfer.



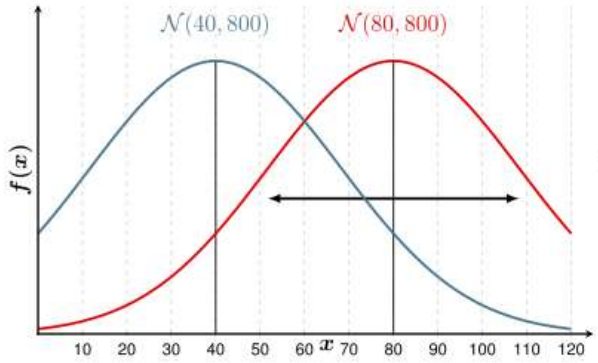
Schlechter: Kleine Varianz zwischen den Gruppen



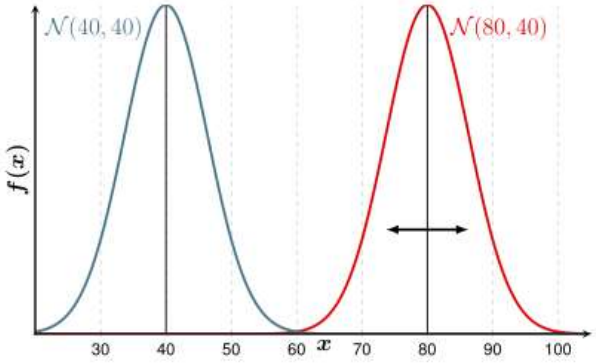
Besser: Große Varianz zwischen den Gruppen

- Kleine Intragruppenvarianz

Die Varianz **innerhalb** einer Gruppe, die **Intragruppenvarianz**, sollte möglichst klein sein, dann durchmischen sich die Verteilungen nicht, die Trennung ist besser.



Schlechter: Große Varianz in einer Gruppe

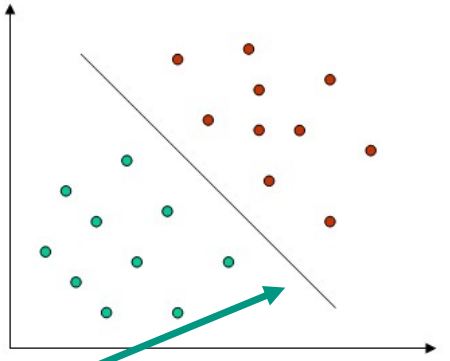


Besser: Kleine Varianz in einer Gruppe

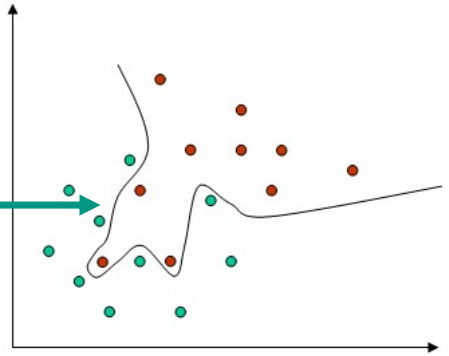
Support Vector Machine

Stützvektormethode: statistisches Verfahren, zur Klassifizierung von Datenobjekten

- Jedes Objekt wird durch einen Vektor in einem Vektorraum repräsentiert.
  - Aufgabe der Support Vector Machine ist es, in diesen Raum eine Hyperebene einzupassen, die als Trennfläche fungiert und die Trainingsobjekte in zwei Klassen teilt.
  - Der Abstand derjenigen Vektoren, die der Hyperebene am nächsten liegen, wird dabei maximiert.
  - Dieser breite, leere Rand soll später dafür sorgen, dass auch Objekte, die nicht genau den Trainingsobjekten entsprechen, möglichst zuverlässig klassifiziert werden.
  - Bei dem Verfahren geht es darum, eine Grenzfläche zu finden, die einen möglichst breiten Bereich zwischen den Grenzen der einzelnen Klassen bildet.
- Die Grenzfläche kann entweder einer linearen Funktion folgen, oder auch von nicht linearem Charakter sein.
- Dazu wird eine Trainingsdatenmenge benötigt, bei der die Klassenzugehörigkeit bekannt ist.



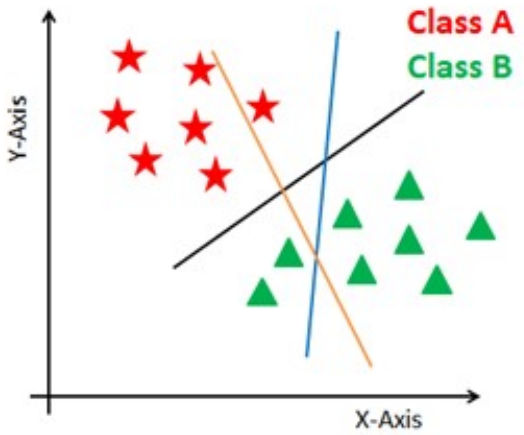
linear trennbar

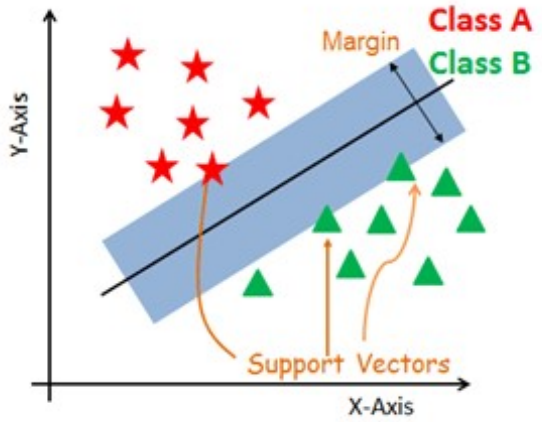


nicht linear trennbar

Support Vector Machine
Stützvektormethode

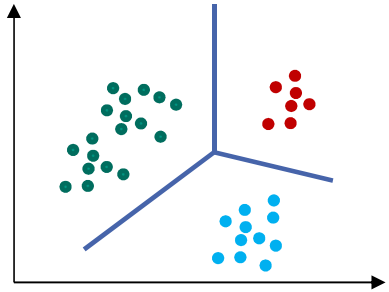
- Dazu werden die Daten solange in einen höherdimensionalen Raum transformiert, bis eine lineare Trennung möglich ist.
- Diese lineare Trennung aus einem höherdimensionalen Raum wird dann wieder in die Ausgangsdimension zurücktransformiert und erscheint dann als „krumme Linie“, die die Gruppen trennt.
- Die Hyperebene ist nur von den ihr am nächsten liegenden Vektoren abhängig – und auch nur diese werden benötigt, um die Ebene mathematisch exakt zu beschreiben.
  - Diese nächstliegenden Vektoren werden nach ihrer Funktion Stützvektoren (engl. *support vectors*) genannt und verhalfen den Support Vector Machines zu ihrem Namen.





Support Vector Machine (überwacht)

- Gruppierung/ Separierung von vorsortierten Daten





- Vorteil:
  - Zielgerichtet durch Vorsortierung
  - Evaluation durch Testdaten möglich
    - Vergleich der entdeckten Muster (Optimierung)
    - Vergleich erleichtert Merkmalselektion
- Nachteil:
  - Nur anwendbar auf vorsortierte Daten
  - Neues Wissen beschränkt auf die vorgegeben Klassen

Bayes-Klassifikation (überwacht)

- Die Bayes-Klassifikation ist eine statistische Klassifikationsmethode, die die Wahrscheinlichkeit vorhersagt, mit der ein Objekt zu einer bestimmten Gruppe gehört.
  - Basiert auf der Formel von Bayes, mit der die bedingte Wahrscheinlichkeit eines Ereignisses berechnet werden kann.
- Bayes' Klassifikationsverfahren gehören zu den überwachten Klassifikatoren, da sie erst durch Trainingsdaten mit bekannter Klassifikation trainiert und dann auf neue Instanzen angewendet werden können.

Bayes-Klassifikation (überwacht)

- Die Entscheidungsregel funktioniert nach dem Prinzip, eine neue Instanz der Klasse zuzuordnen, bei der die berechnete Wahrscheinlichkeit für diese Klasse am größten ist.
  - Dabei wird eine A-priori-Wahrscheinlichkeit mit den gewichteten bedingten Wahrscheinlichkeiten berechnet.
- Bei vielen Variablen wird die Berechnung der (optimalen) Bayes-Klassifikation sehr aufwendig, sodass als Näherung die *Naive* Bayes- Klassifikation als Verfahren angewendet werden kann.
  - Alle Attribute werden dabei so behandelt als wären sie statistisch unabhängig.
  - Damit entfällt die Notwendigkeit der Berechnung der bedingten „Kreuzwahrscheinlichkeiten“ und der Berechnungsaufwand steigt nicht exponentiell mit der Anzahl der Variablen.
  - Obwohl die Annahme der Unabhängigkeit der Variablen in der Praxis häufig verletzt wird, liefert die Naive Bayes-Klassifikation trotzdem gute Ergebnisse (zumindest für den Fall, dass sich die Korrelationen in Grenzen halten).

Naive Bayes-Klassifikation

Beispiel

- Wir haben Früchte, die mit drei Variablen beschrieben werden
  - Länge
  - Geschmack
  - Farbe
- In der Tabelle sind die 1.000 Trainingsdaten dargestellt, mit denen das Klassifizierungsmodell erstellt werden soll.

	Länge		Geschmack		Farbe		Gesamt
	lang	kurz	süß	nicht süß	gelb	and. Farbe	
Banane	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Sonst. Frucht	100	100	150	50	50	150	200
Gesamt	**500**	**500**	**650**	**350**	**800**	**200**	**1000**

Die A-priori-Wahrscheinlichkeiten sind:

$P_{(Banane)} = 0{,}5$ $P_{(Orange)} = 0{,}3$ $P_{(Sonst)} = 0{,}2$

$P_{(lang)} = 0{,}5$ $P_{(süß)} = 0{,}65$ $P_{(gelb)} = 0{,}8$

Likelihood – Ausprägungswahrscheinlichkeiten:

$P_{(lang \mid Banane)} = 0{,}8$ (400/500)

$P_{(lang \mid Orange)} = 0$ (es gibt keine langen Orangen)

.....

$P_{(and.\ Farbe \mid Orange)} = 0$

$P_{(and.\ Farbe \mid Sonst.)} = 0{,}75$ (150/200)

Beispiel: Lang, süß, gelb

Es soll nun eine neue Frucht klassifiziert werden, die lang, süß und gelb ist:

- Man berechnet zuerst die bedingte Wahrscheinlichkeit, dass die Frucht eine Banane, eine Orange oder eine andere Frucht ist.
- Die Formel für die Berechnung ist das Produkt aus den Ausprägungs-wahrscheinlichkeiten mit der A-priori-Wahrscheinlichkeit, geteilt durch die A-priori-Wahrscheinlichkeiten der Variablen.

- Sieht z. B. für die Banane so aus:

$$P(\text{Banana}|\text{Long, Sweet and Yellow}) = \frac{P(\text{Long}|\text{Banana}) * P(\text{Sweet}|\text{Banana}) * P(\text{Yellow}|\text{Banana}) * P(\text{banana})}{P(\text{Long}) * P(\text{Sweet}) * P(\text{Yellow})}$$

$$= 0.8 * 0.7 * 0.9 * 0.5 \,/P(\text{evidence})$$

$$= 0.252 \,/P(\text{evidence})$$

$$P(\text{Orange}|\text{Long, Sweet and Yellow}) = 0$$

$$P(\text{Other Fruit}|\text{Long, Sweet and Yellow}) = \frac{P(\text{Long}|\text{Other fruit}) * P(\text{Sweet}|\text{Other fruit}) * P(\text{Yellow}|\text{Other fruit}) * P(\text{Other Fruit})}{P(\text{evidence})}$$

$$= (100/200 * 150/200 * 50/200 * 200/1000) \,/P(\text{evidence})$$

$$= 0.01875 \,/P(\text{evidence})$$

	Länge		Geschmack		Farbe		
	lang	kurz	süß	nicht süß	gelb	and. Farbe	
Banane	400	100	350	150	450	50	**500**
Orange	0	300	150	150	300	0	**300**
Sonst. Frucht	100	100	150	50	50	150	**200**
Gesamt	**500**	**500**	**650**	**350**	**800**	**200**	**1000**

Beispiel: Lang, süß, gelb

- Man berechnet zuerst die bedingte Wahrscheinlichkeit, dass die Frucht eine Banane, eine Orange oder eine andere Frucht ist.
- Die Formel für die Berechnung ist das Produkt aus den Ausprägungs-wahrscheinlichkeiten mit der A-priori-Wahrscheinlichkeit, geteilt durch die A-priori-Wahrscheinlichkeiten der Variablen.

$$P(\text{Banana}|\text{Long, Sweet and Yellow}) = \frac{P(\text{Long}|\text{Banana}) * P(\text{Sweet}|\text{Banana}) * P(\text{Yellow}|\text{Banana}) * P(\text{banana})}{P(\text{Long}) * P(\text{Sweet}) * P(\text{Yellow})}$$

$$= 0.8 * 0.7 * 0.9 * 0.5 \,/P(\text{evidence})$$

$$0.252 \,/P(\text{evidence})$$

$$P(\text{Orange}|\text{Long, Sweet and Yellow}) = 0$$

$$P(\text{Other Fruit}|\text{Long, Sweet and Yellow}) = \frac{P(\text{Long}|\text{Other fruit}) * P(\text{Sweet}|\text{Other fruit}) * P(\text{Yellow}|\text{Other fruit}) * P(\text{Other Fruit})}{P(\text{evidence})}$$

$$= (100/200 * 150/200 * 50/200 * 200/1000) \,/P(\text{evidence})$$

$$0.01875 \,/P(\text{evidence})$$

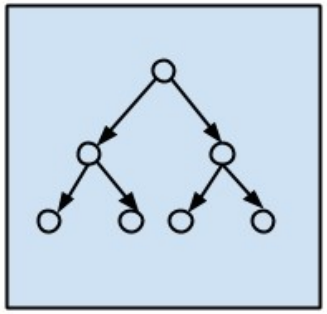
- Danach wählt man den wahrscheinlichsten Wert aus, sofern man mit dem Unterschied der Wahrscheinlichkeit zufrieden ist.
  - Da die lange, süße, gelbe Frucht mit dem Wert 0,252 mehr als 10-fach so wahrscheinlich eine Banane als eine sonstige Frucht ist (0,01875), kann man die unbekannte neue Frucht „guten Gewissens“ als Banane klassifizieren.
- Die naive Bayes-Klassifikation wird häufig für die Klassifikation von Texten verwendet, beispielsweise in Spam-Filter, die E-Mails als Spam bzw. kein Spam kategorisieren.

Decision Tree Algorithms

Entscheidungsbäume

- Entscheidungsbäume sind Baumstrukturen, die der Darstellung von Entscheidungsregeln dienen.
  - Sie veranschaulichen hierarchisch aufeinanderfolgende Entscheidungen.
  - Ihre Anwendungsgebiete umfassen dabei die automatische Klassifizierung aus Erfahrungswissen.
  - Ein Entscheidungsbaum besteht immer aus einem Wurzelknoten und beliebig vielen inneren Knoten sowie mindestens zwei Blättern.
  - Dabei repräsentiert jeder Knoten eine Regel und jedes Blatt eine Antwort auf das Entscheidungsproblem.
- Grundsätzlich lassen sich Entscheidungsbäume in zwei Varianten unterteilen:
  - **Klassifikationsbäume** zeigen eine Auswahl von **diskreten** Klassen und deren Beziehungen untereinander.
  - **Regressionsbäume** dienen der Prognose eines **stetigen** Wertes der abhängigen Variable.

- Entscheidungsbäume können entweder von Experten manuell aufgestellt werden oder sie werden mit Techniken des maschinellen Lernens automatisch aus gesammelten Daten erstellt.
  - Bei letztgenanntem besteht die Möglichkeit, dem Algorithmus die Zahl der Ebenen vorzugeben.
- Die beiden am weitesten verbreiteten Verfahren sind:
  - *Classification and Regression Tree (CART)*
  - *Iterative Dichotomiser 3 (ID3)*

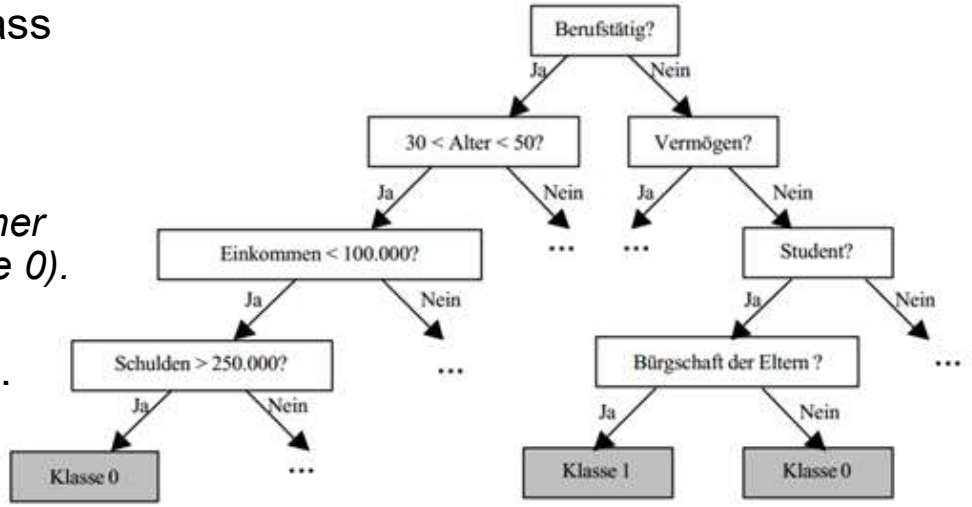


Decision Tree Algorithms

Entscheidungsbäume

Beispiel: Kreditvergabe

- Die Vorteile der Entscheidungsbäume liegen darin, dass sie i.d.R. gut zu verstehen und zu interpretieren sind.
  - Beispielsweise kann der Entscheidungsbaum zu einer Kreditentscheidung in Fließtext interpretiert werden:
  - *Einem mittellosen Studenten, der keine Bürgschaft seiner Eltern vorweisen kann, wird kein Kredit gewährt (Klasse 0).*
- Entscheidungsbäume führen häufig zu schlechteren Klassifikationsergebnissen als andere Verfahren (z. B. neuronale Netzwerke).
  - Die Verständlichkeit der Regeln geht auf Kosten der Klassifikationsgüte.
  - Außerdem besteht – je nach Verfahren und Art der Daten – die Gefahr, dass die Bäume zu groß und damit schwerer verständlich bzw. weniger aussagekräftig werden.



- Remark: *Eine Erweiterung der Entscheidungsbäume sind die Entscheidungswälder (Decision Forests).*
  - *Dabei handelt es sich um den kombinierten Einsatz mehrerer Entscheidungsbäume.*
  - *Über eine Kombination der Entscheidungsbäume anhand von Mehrheitsentscheidungen soll die Klassifikationsgüte erhöht werden.*

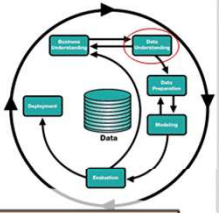
Decision Tree

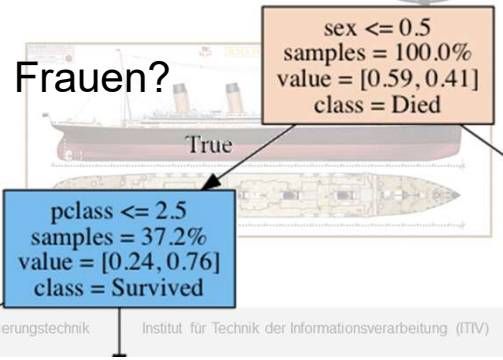
Beispiel: Titanic Dataset

CRISP-DM im Detail: Data Understanding

Merkmale (Features)

- Beispiel: Titanic-Datensatz
  - 11 Features (Name, Alter, Geschlecht, Ticketklasse, Überlebt,…)
    - Survival
    - Pclass
    - Name
    - Sex
    - Age
    - SibSp
    - Parch
    - Ticket
    - Fare
    - Cabin
    - Embarked





Frauen?

Männer?

1. Und 2. Klasse?

1. Klasse?

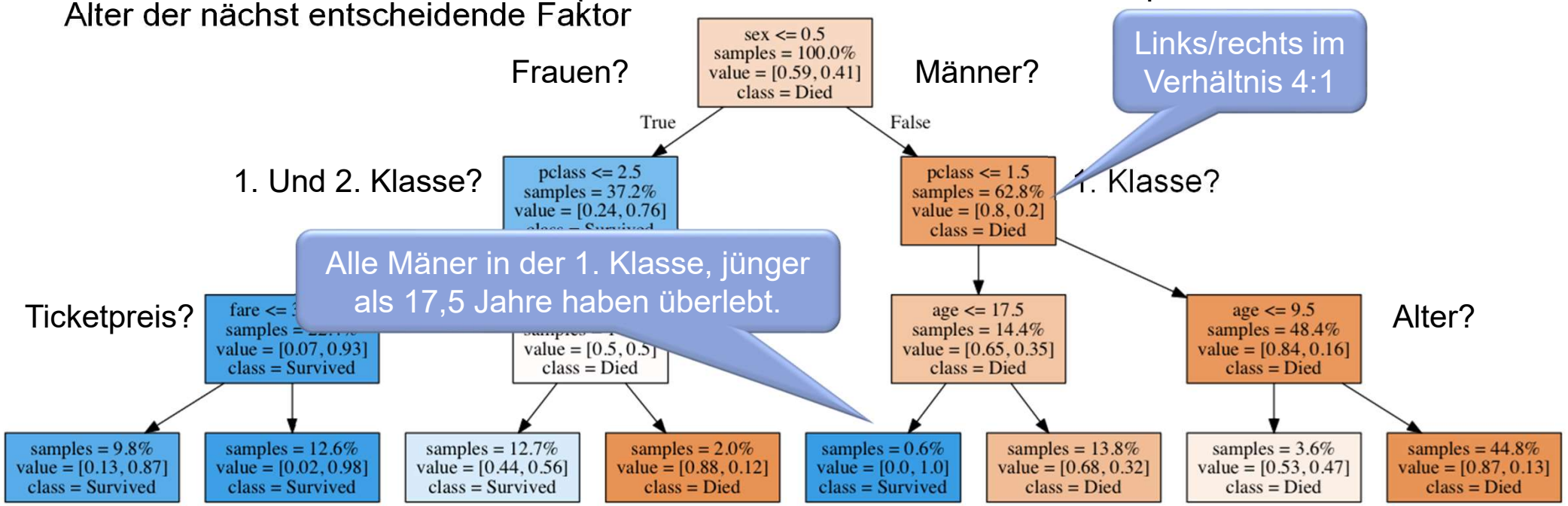
Ticketpreis?

Alter?

Decision Tree

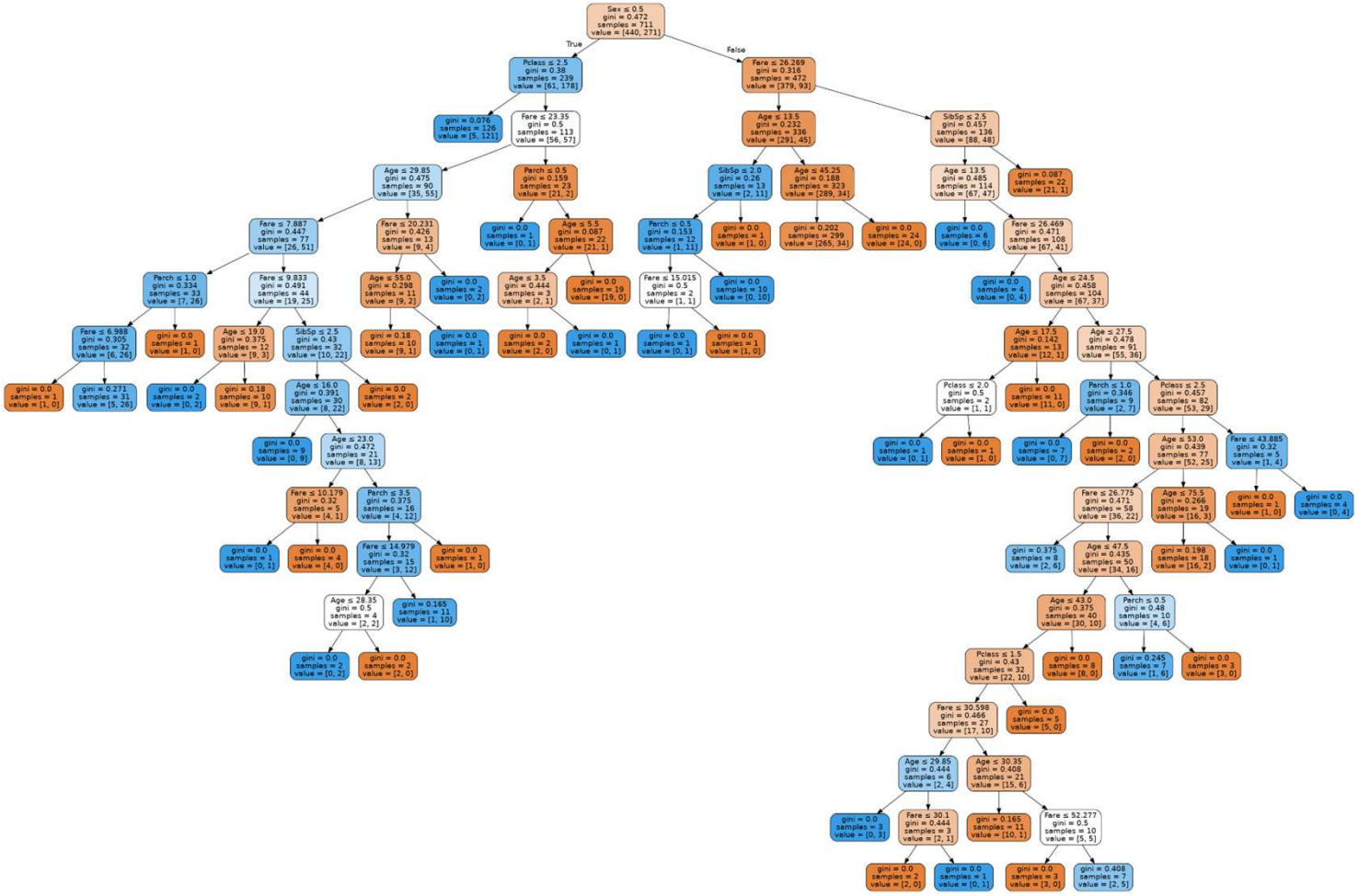
http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

- Es wird dem Algorithmus eine max. Ebenenzahl von 3 (+ 1 Wurzelknoten) vorgegeben.
- Der Algorithmus teilt zunächst nach “sex” und dann “class”, da in der Traingingsphase die beiden Parameter als am einflussreichsten identifiziert wurden, um über Leben und Tod zu entscheiden.
  - Je dunkler das Blau, umso höher die Wahrscheinlichkeit zu überleben
  - Je dunkler das Orange, umso höher die Wahrschieinlichkeit zu sterben.
- Interessanterweise ist nach dem Split nach “class” bei Frauen der Tickettpreis und bei Männern das Alter der nächst entscheidende Faktor



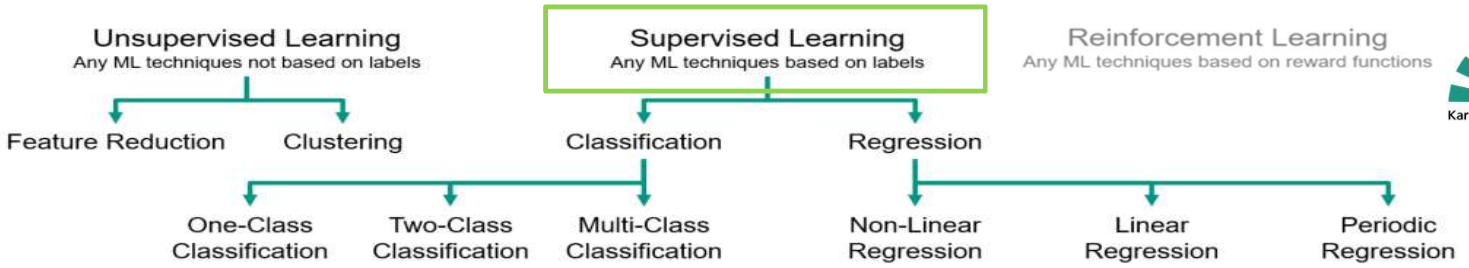
… und falls man das Verfahren nicht auf 3 Ebenen beschränkt …

Die „Blätter“ sind nun eindeutig „survived“ und „died“




5 59

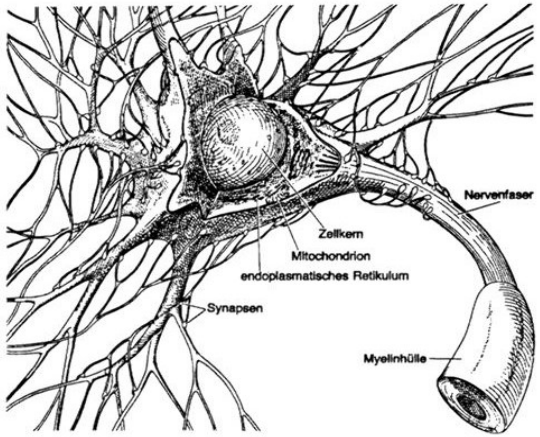
Machinelles Lernen



Algorithm ↓	Abk.	Feature Reduction	Clustering	One-Class Classification	Two-Class Classification	Multi-Class Classification	Non-Linear Regression	Linear Regression	Periodic Regression
Faktorenanalyse		X							
Principal Component Analysis	PCA	X		X					
	K-means		X						
hierarchical cluster analysis	HCA		X						
DBSCAN			X						
One Class Support Vector Machine	OCSVM			X					
Isolation Forest				X					
	LODA			X					
(künstliche) Neuronale Netze	NN			X (Autoencoder)	X	X	X	X	X
Support Vector Machine	SVM				X				
Decision Tree					X	X			
Bayes-Klassifikation					X	X			
Random Forest						X			
Diskriminanzanalyse				x	x	x			
Logistic Regression							X	X	
Linear Regression								X	
Harmonic Regression									X
Nächste-Nachbar-Klassifikation	K-NN	x	x						

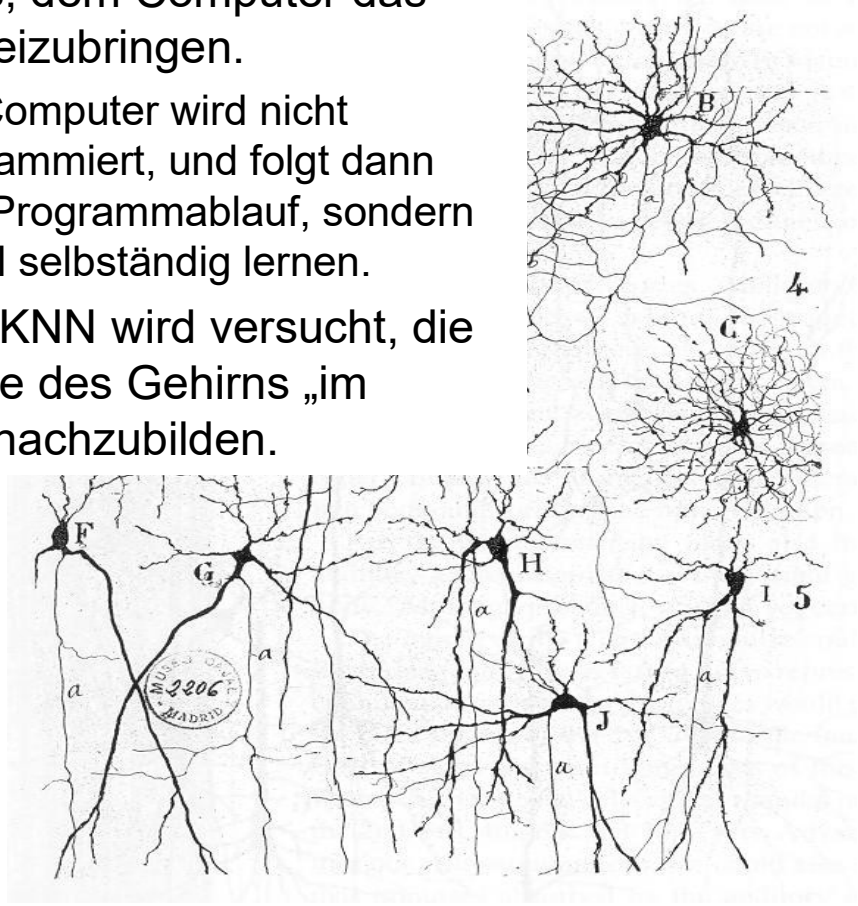
Neuron, (künstliches) Neuronales Netz / KNN

- Kombination aus mathematischer Matrizenrechnung und einem Ablaufschema (Lernalgorithmus), mit dem iterativ Verbesserungen ermöglicht werden „sollen“.
- Keine „Rocket Science“ und keine „Frankenstein’sche Alchemie“
- Aufbau von Neuronen
  - Gehirn besteht aus 100 Milliarden Neuronen



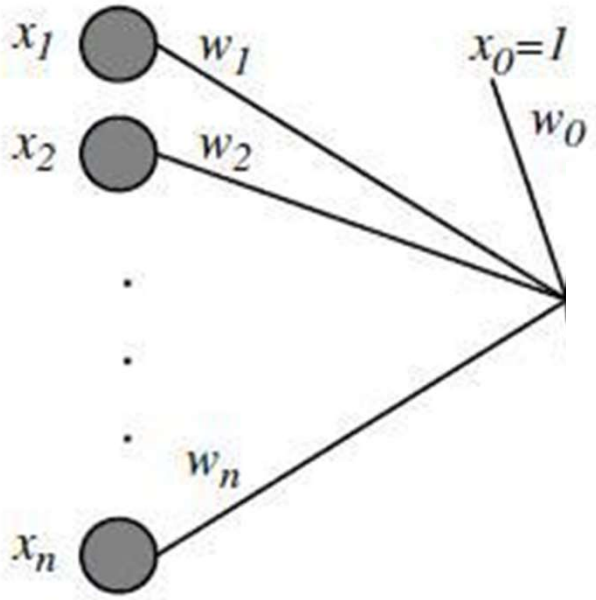
- Dendriten nehmen elektrische Impulse auf
- Zellkörper verarbeitet ankommende Impulse
- Nervenfaser (Axon) leitet gefeuerten Impuls weiter
- Synapsen sind die Endknöpfchen des Axons, übertragen Impulse an nächste Zelle

- Ziel ist es, dem Computer das Lernen beizubringen.
  - Der Computer wird nicht programmiert, und folgt dann dem Programmablauf, sondern er soll selbständig lernen.
- In einem KNN wird versucht, die Grundidee des Gehirns „im Kleinen“ nachzubilden.



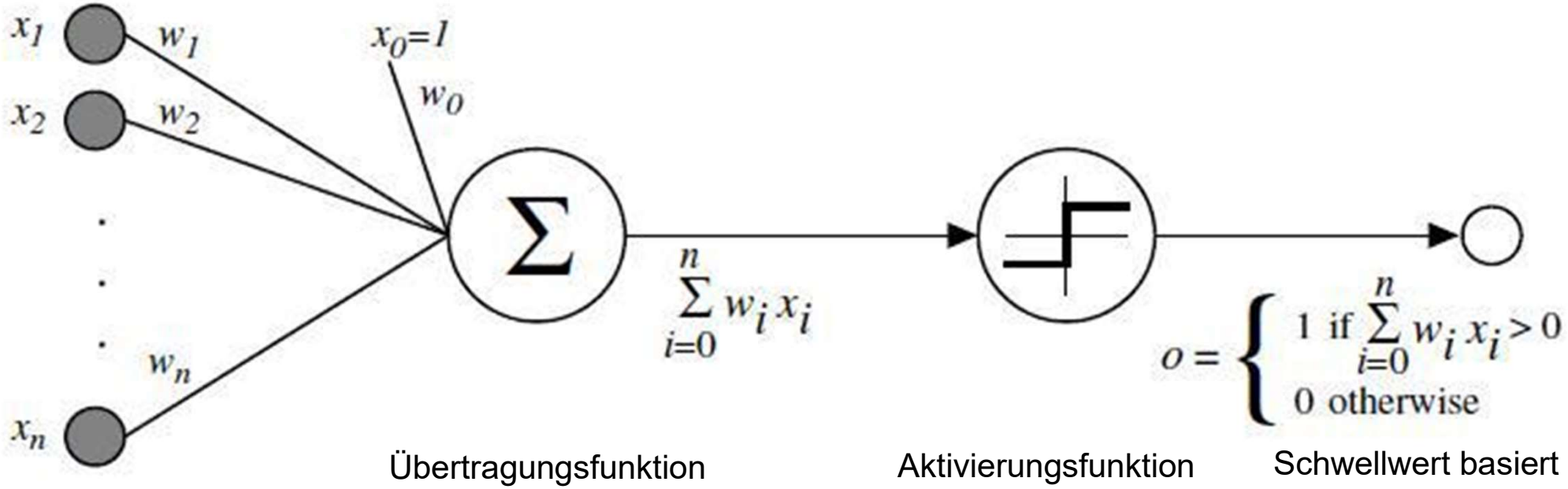
Perceptron

- Das Perzeptron ist ein vereinfachtes künstliches neuronales Netz, das zuerst von Frank Rosenblatt 1958 vorgestellt wurde.
  - Es besteht in der Grundversion aus 1 einzelnen künstlichen Neuron mit anpassbaren Gewichtungen.



Perceptron

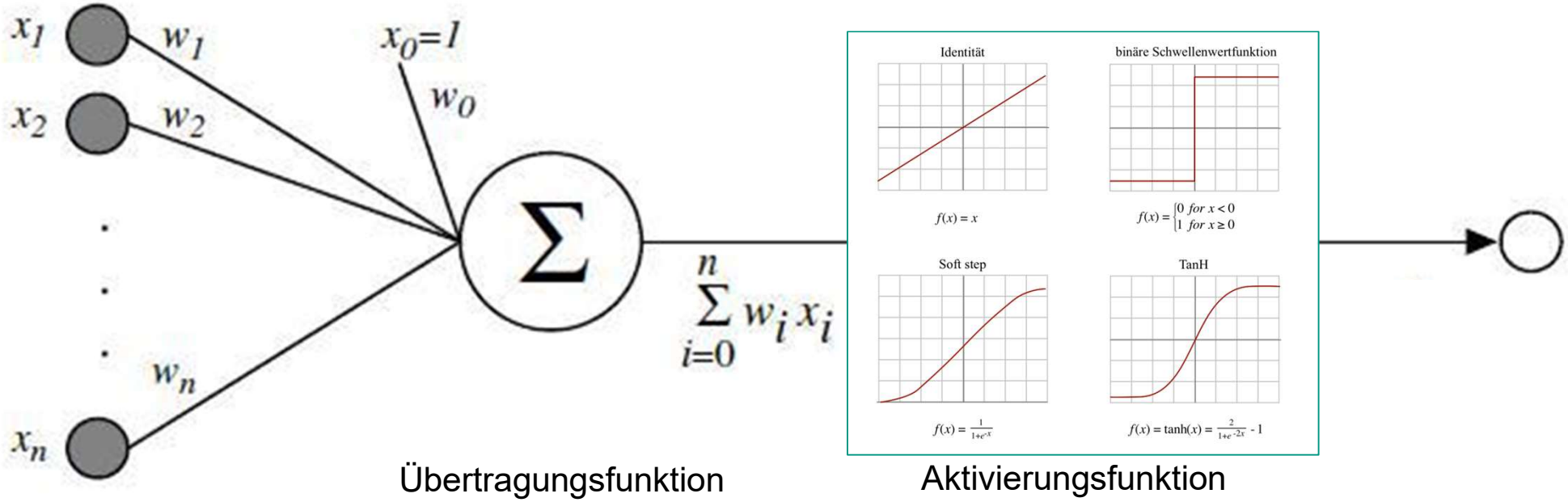
- Die Aktivierungsfunktion beschreibt den funktionalen Zusammenhang zwischen Input und Output eines Knotens.
- Der gesamte Input wird über die sog. Propagierungsfunktion bzw. Übertragungsfunktion bestimmt.



Quelle: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/

Perceptron

- Die verbreitetste Übertragungsfunktion ist eine Linearkombination, bei der sich der Netzinput additiv aus sämtlichen gewichteten Inputs zusammensetzt, die das Neuron von anderen Neuronen erhält.



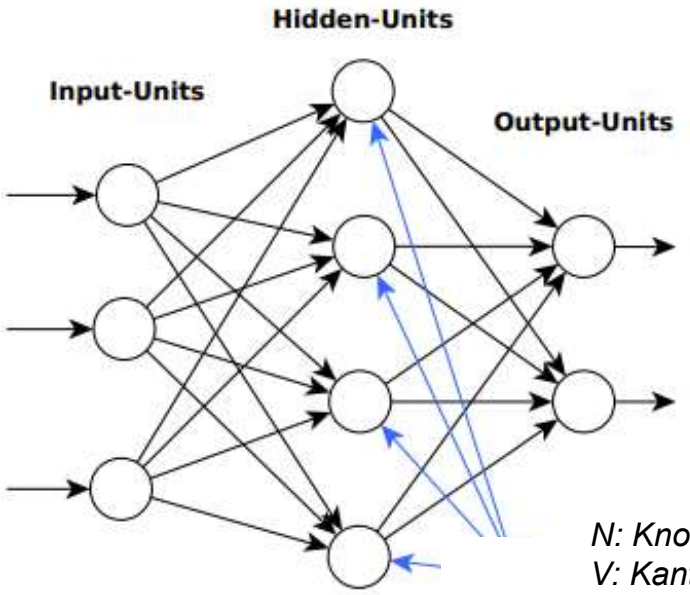
Übertragungsfunktion Aktivierungsfunktion

Quelle: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/

Definition des künstlichen neuronalen Netzes (KNN)

- Ein KNN besteht aus Knoten (Einheiten oder Neuronen N) und Kanten (Verbindungen V).
  - Die Stärke der Verbindung wird durch ein Gewicht w ausgedrückt.
- Die Knoten verfügen über einen Wert der Aktivierung, der beschreibt ob und wie aktiv er ist.
  - Das ist sozusagen der Schaltzustand.
- Die Struktur des Netzwerkes wird als „Topologie“ bezeichnet.
- Das Wissen des Netzwerks besteht aus der Gesamtheit der Gewichte der einzelnen Kanten.
  - Als Ausgangswerte werden i.d.R. Zufallswerte genommen.
  - Lernen erfolgt durch iteratives Anpassen der Gewichtungen nach unterschiedlichen Lernalgorithmen.

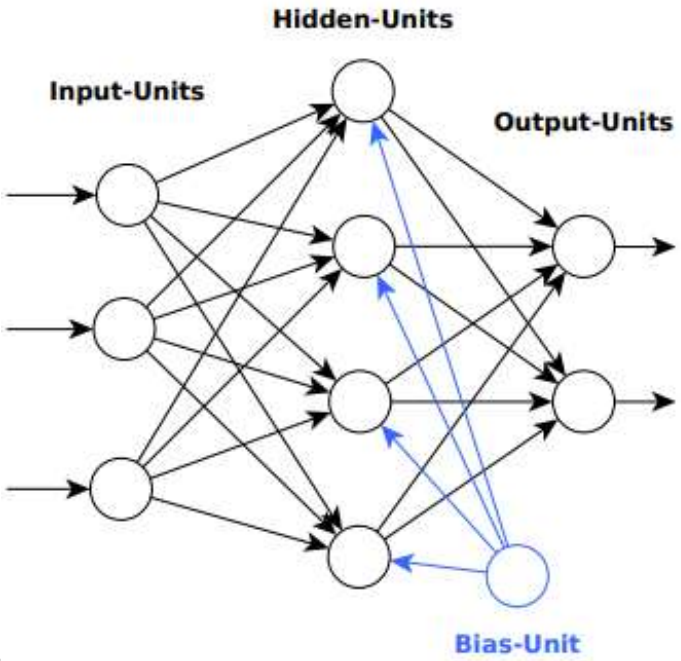
„Ein Neuronales Netz ist ein sortiertes Tripel (N, V, w) mit zwei Mengen N, V sowie einer Funktion w, wobei N die Menge der Neurone bezeichnet und V eine Menge $\{(i, j)|i, j \in N\}$ ist, deren Elemente Verbindungen von Neuron i zu Neuron j heißen. Die Funktion $w : V \rightarrow R$ definiert die Gewichte, wobei w((i, j)), das Gewicht der Verbindung von Neuron i zu Neuron j, kurz mit wi,j bezeichnet wird.“



N: Knoten
V: Kanten
w: Gewichte

Arten von Neuronen

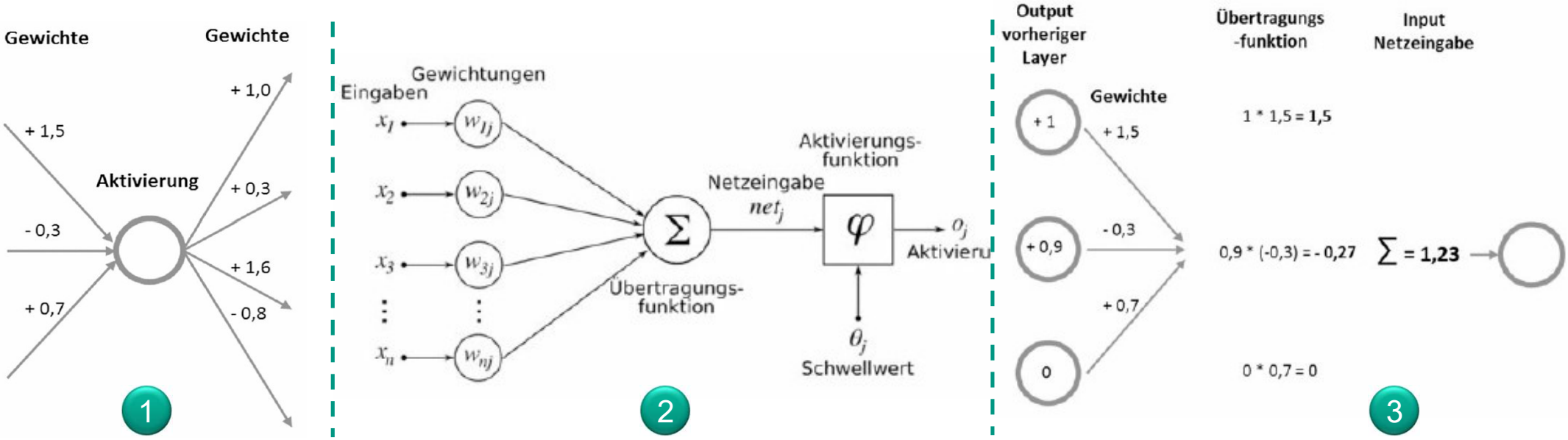
- Input-Units:
  - nur ein Eingang
  - Aktivierungsfunktion entfällt
- Hidden-Units:
  - Ein- und Ausgabe unbekannt
  - Deshalb „versteckte“ Knoten
- Output-Units:
  - Ausgabe des Systems
- Bias-Units:
  - Die Bias-Unit erhält selbst keinen Input,
  - ihr Aktivitätslevel beträgt immer +1.
  - Das Gewicht von der Bias-Unit zu einer anderen Unit kann positiv oder negativ sein.
  - Wenn kein starker Input von anderen Einheiten erfolgt, dann stellt die Bias-Unit sicher, dass die Einheit bei positivem Gewicht aktiv bleibt.
  - Bei negativem Gewicht sorgt die Bias-Einheit hingegen dafür, dass die Unit in ihrem inaktiven Zustand verharrt.
  - Funktion der Bias-Unit
    - kann nützlich sein, wenn man eine Schwelle benötigt (beim negativen Bias), die andere Input-Units erst überschreiten müssen.
    - Umgekehrt kann das Ziel auch sein, dass die Einheit sehr häufig feuern, also gewöhnlich aktiv sein soll. Dazu verwendet man einen positiven Bias.



Beispiel

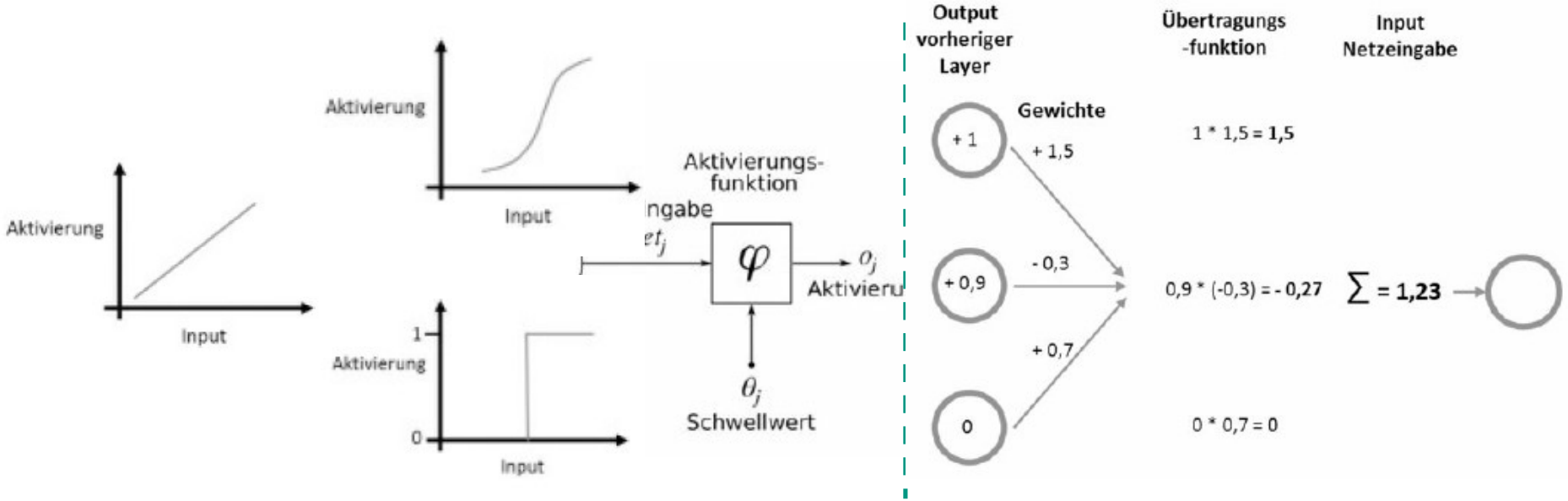
Ausgangspunkt sind gegebene, willkürliche Werte für die Gewichte der Kanten (1).

- Der Wert für die Aktivierung und den Output der Einheit in Abhängigkeit vom Input in das Netzwerk ergibt sich über Zusammenhang (2)
- Der Input x_1 bis x_n entspricht dem Output der entsprechenden vorangegangen Knoten.
- Dieser wird mit den Kantengewichten multipliziert und summiert (3)
  - Die Übertragungsfunktion ist also annahmegemäß die Summe der gewichteten Eingaben x_1 bis x_n.



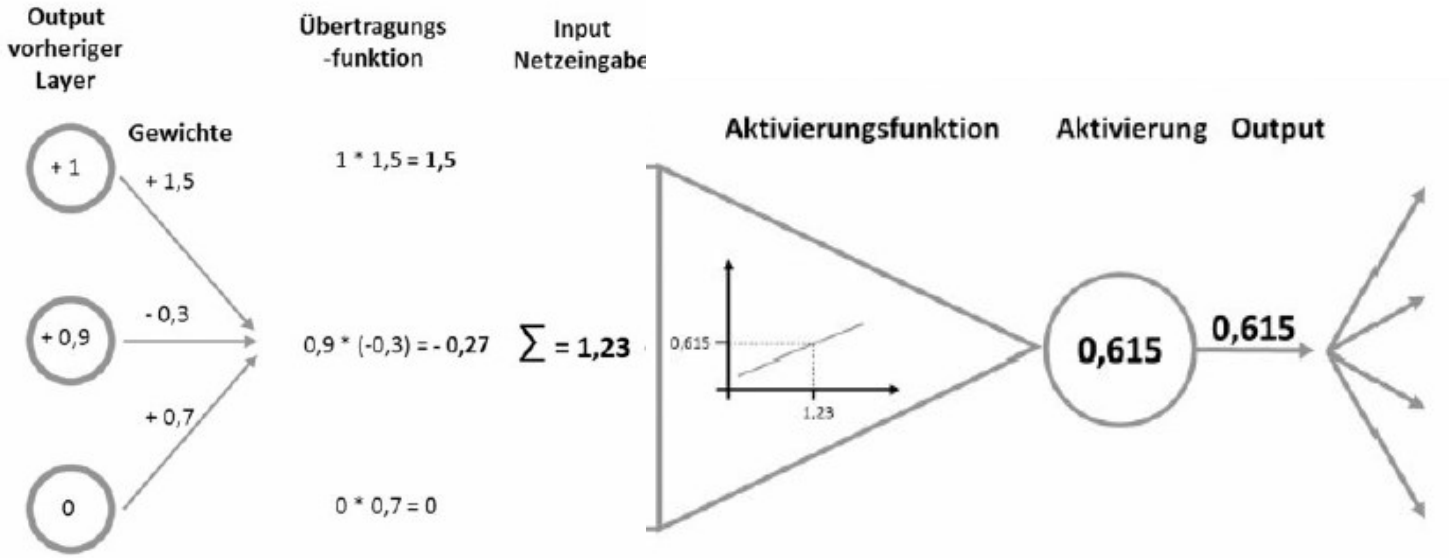
Aktivierungsfunktion

- Aus dem Input wird über die Aktivierungsfunktion der Aktivierungswert berechnet.
- Es handelt sich um eine funktionale Zuordnung des Inputs mit der Aktivierung der Zelle.
- Die Aktivierungsfunktion kann unterschiedliche Ausprägungen einnehmen:
  - Lineare Funktion ohne/mit Schwellenwert
  - Binäre Funktion
  - Sigmoide (z. B. logistische) Funktion



Aktivierungsfunktion

- Aus dem Input wird über die Aktivierungsfunktion der Aktivierungswert berechnet.
- Es handelt sich um eine funktionale Zuordnung des Inputs mit der Aktivierung der Zelle.
- Annahme: die Aktivierungsfunktion ist eine lineare Funktion ohne Schwellenwert mit der „Steigung“ 0,5 und die Aktivierung entspricht dem Output:
  - Aktivierung = 0,5 *
  - Input = Output = 0,615



- Berechnung einer Einheit bzw. eines Neurons!

Fragen bzw. Handlungsoptionen

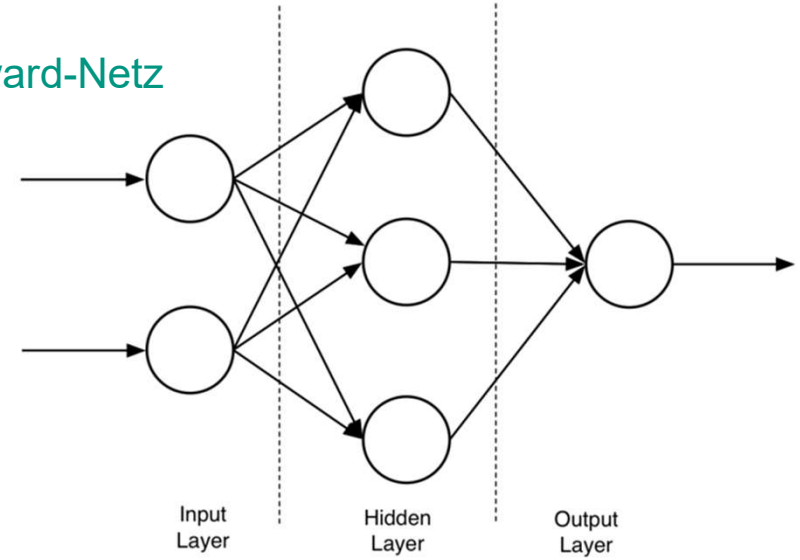
Neuronale Netze, Einstellgrößen (*Hyperparameter*)

- Soll das KNN für ein überwachtes oder unüberwachtes Lernen eingesetzt werden, oder anders ausgedrückt:
  Liegen ge-labelte Lerndaten vor (also Trainingsdaten, für die Input- und Output-Werte vorliegen) oder soll mit dem KNN eine Datenmenge mit nicht vorhandenen Output-Werten zur Erkennung von Mustern verwendet werden?
  - Im ersten Fall müssen über einen Lernalgorithmus die Gewichtungen der Kanten solange angepasst werden, bis das Netz die Trainingsdaten hinreichend gut wiedergibt.
  - Im zweiten Fall muss der Lernalgorithmus die Gewichtungen im Netz solange anpassen, bis das Lernziel (z. B. die Mustererkennung oder die Clusterbildung) hinreichend gut erfüllt ist.
    ► Es müssen also die für das Ziel des Lernens geeigneten Lernregeln ausgewählt werden.
- Welche Topologie soll mein KNN haben?
  - Wie viele Knoten und wie viele Hidden Layers soll das Netz haben?
- Wie ist die Richtung der Verbindungen?
  - Die Verbindungen können ausschließlich in eine Richtung (von „links nach rechts“) und immer nur zwischen zwei benachbarten Layers vorkommen (*Feedforward Netze*), oder aber es sind Rückkopplungen in die andere Richtung (indirekte Rückkopplung), im selben Layer (seitliche Rückkopplung), oder sogar mit derselben Unit (*direkte Rückkopplung*) möglich.
- Welche Aktivierungsfunktion soll ausgewählt werden?

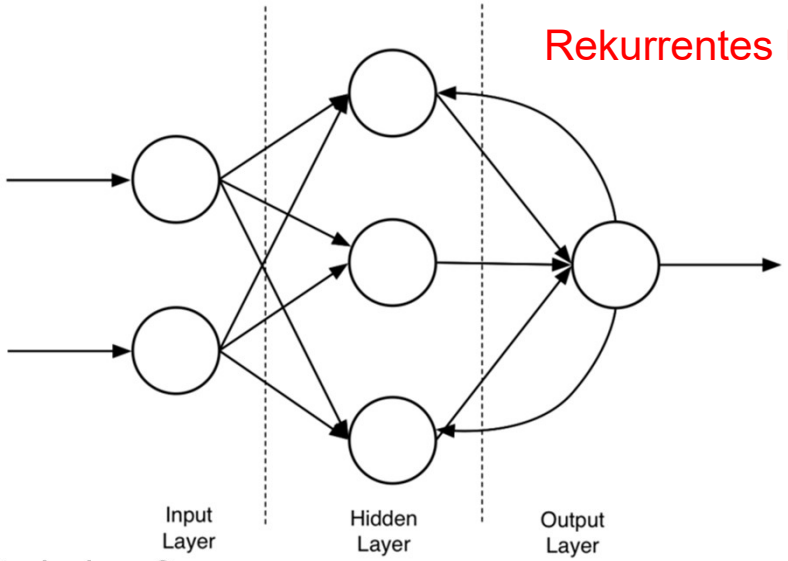
Fragen bzw. Handlungsoptionen

Neuronale Netze, Einstellgrößen (*Hyperparameter*)

Feedforward-Netz



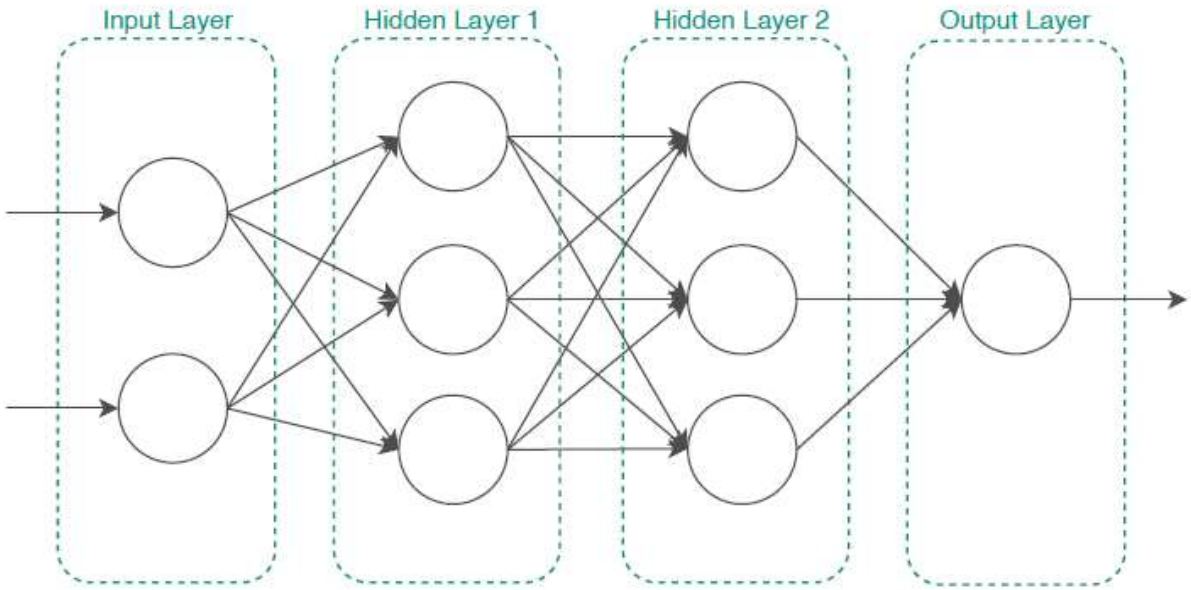
Rekurrentes Netze



- Wie viele Knoten und wie viele Hidden Layers soll das Netz haben?
- Wie ist die Richtung der Verbindungen?
  - Die Verbindungen können ausschließlich in eine Richtung (von „links nach rechts“) und immer nur zwischen zwei benachbarten Layers vorkommen (*Feedforward Netze*), oder aber es sind Rückkopplungen in die andere Richtung (*indirekte Rückkopplung*), im selben Layer (seitliche Rückkopplung), oder sogar mit derselben Unit (*direkte Rückkopplung*) möglich.
- Welche Aktivierungsfunktion soll ausgewählt werden?

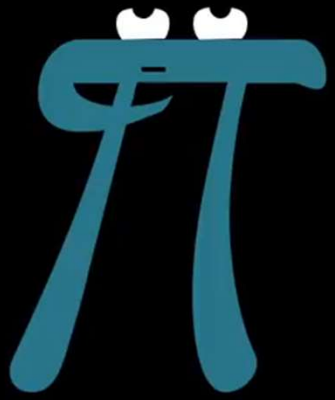
Deep Learning

- Es gibt unzählige Ausprägungen an Netzwerken.
- Ein wichtiges Merkmal ist die Frage, ob und wieviele „Hidden Layers“ vorhanden sind.
- Bei mehr als einem Hidden Layer spricht man von **Deep Learning**
- Die am weitesten verbreiteten sind:
  - Convolutional Neural Network (CNN)
  - Stacked Auto-Encoders



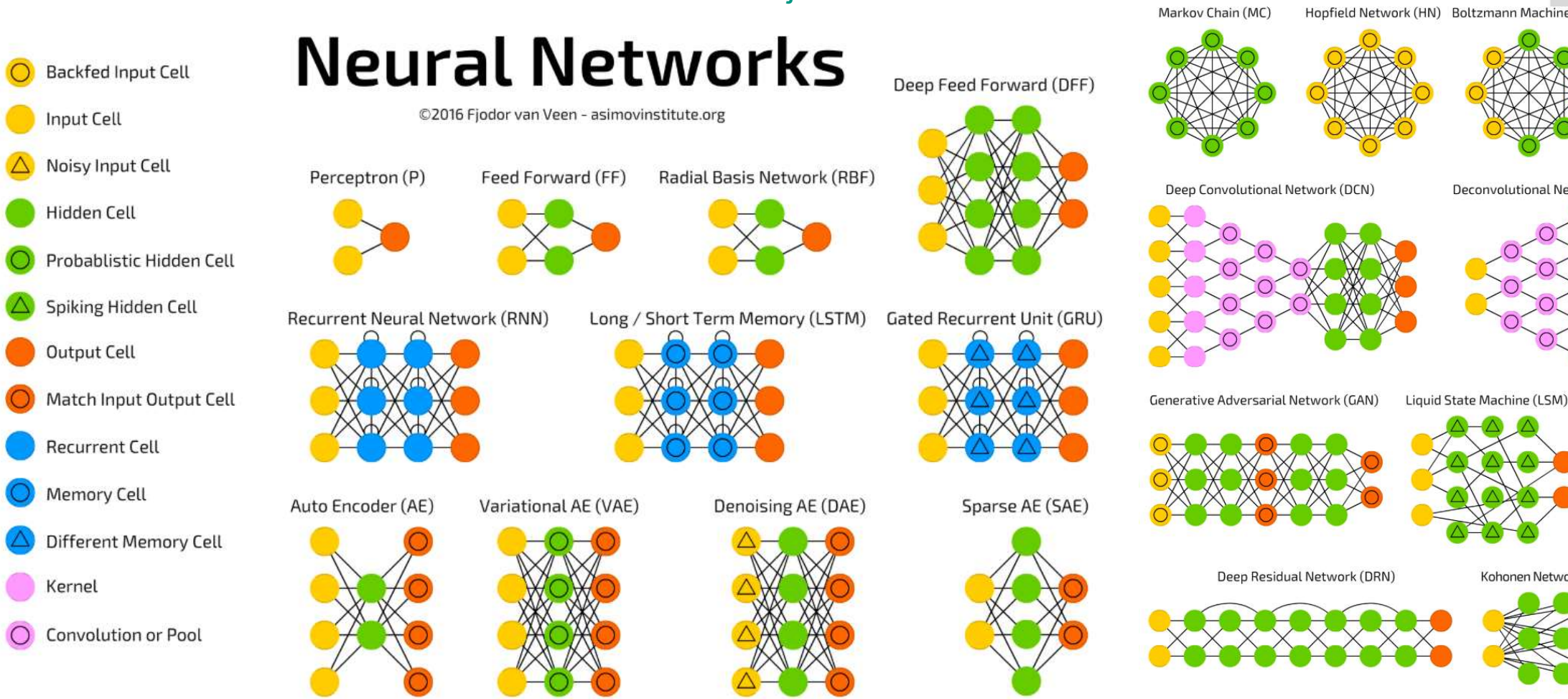
Neural network





Übersicht Architekturen

Verschiedene Arten von neuronalen Netzen nach Fjodor van Veen



Convolutional Neural Networks (CNN)

„faltendes Neuronales Netz“ → Stand der Technik für Bilderkennung

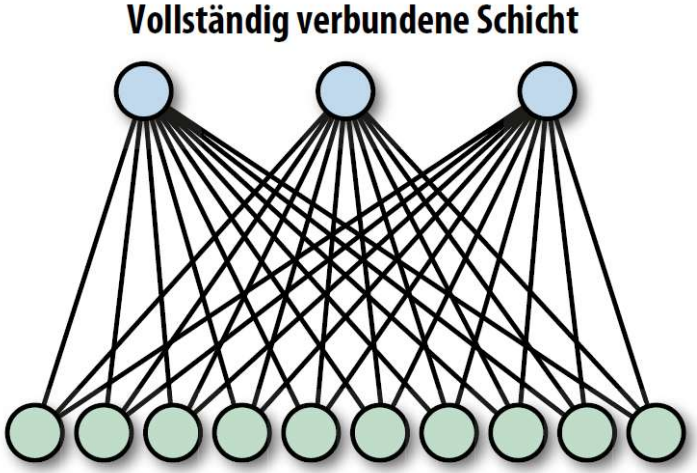
- Der grundlegende Unterschied zwischen *vollständig verbundenen Netzen* und *Konvolutionsnetzen* ist das Verbindungsmuster aufeinanderfolgender Schichten.
  - In einer vollständig verbundenen Schicht ist jede Einheit mit allen Einheiten der vorigen Schicht verbunden
  - In der Konvolutionsschicht eines CNN dagegen ist jede Einheit mit wenigen räumlich nahen Einheiten der vorigen Schicht verknüpft (passend z.B. bei Bildern, wo Zusammenhänge zwischen weit entfernten Bildpunkten nicht in Betracht gezogen werden)
  - Außerdem sind sämtliche Einheiten mit der vorigen Schicht auf dieselbe Weise, mit genau denselben Gewichten und derselben Struktur verbunden.

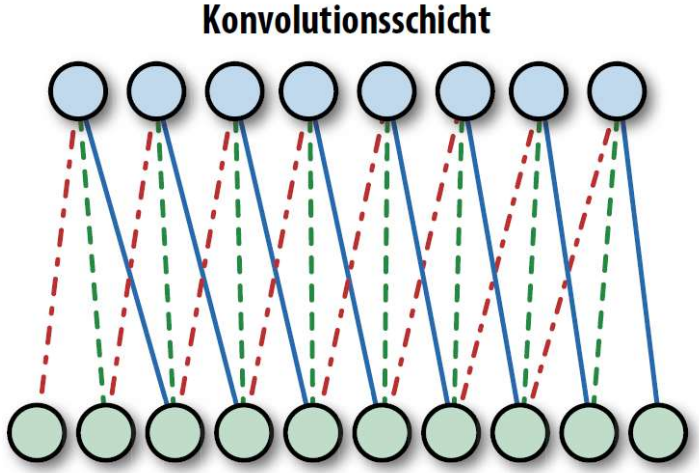
Convolutional Neural Networks (CNN)

„faltendes Neuronales Netz“

- Der grundlegende Unterschied zwischen *vollständig verbundenen Netzen* und *Konvolutionsnetzen* ist das Verbindungsmuster aufeinanderfolgender Schichten.
  - In einer vollständig verbundenen Schicht ist jede Einheit mit allen Einheiten der vorigen Schicht verbunden.
  - In der Konvolutionsschicht eines CNN dagegen ist jede Einheit mit wenigen räumlich nahen Einheiten der vorigen Schicht verknüpft (passend z.B. bei Bildern, wo Zusammenhänge zwischen weit entfernten Bildpunkten nicht in Betracht gezogen werden)
  - Außerdem sind sämtliche Einheiten mit der vorigen Schicht auf dieselbe Weise, mit genau denselben Gewichten und derselben Struktur verbunden.

- Es werden kleine »Fenster« von Gewichten (sogenannte *Filter)* über Eingangsdaten (z.B. ein Bild) hinweg angewendet.
  - Dadurch wird der als *Konvolution* bezeichnete Vorgang ermöglicht.

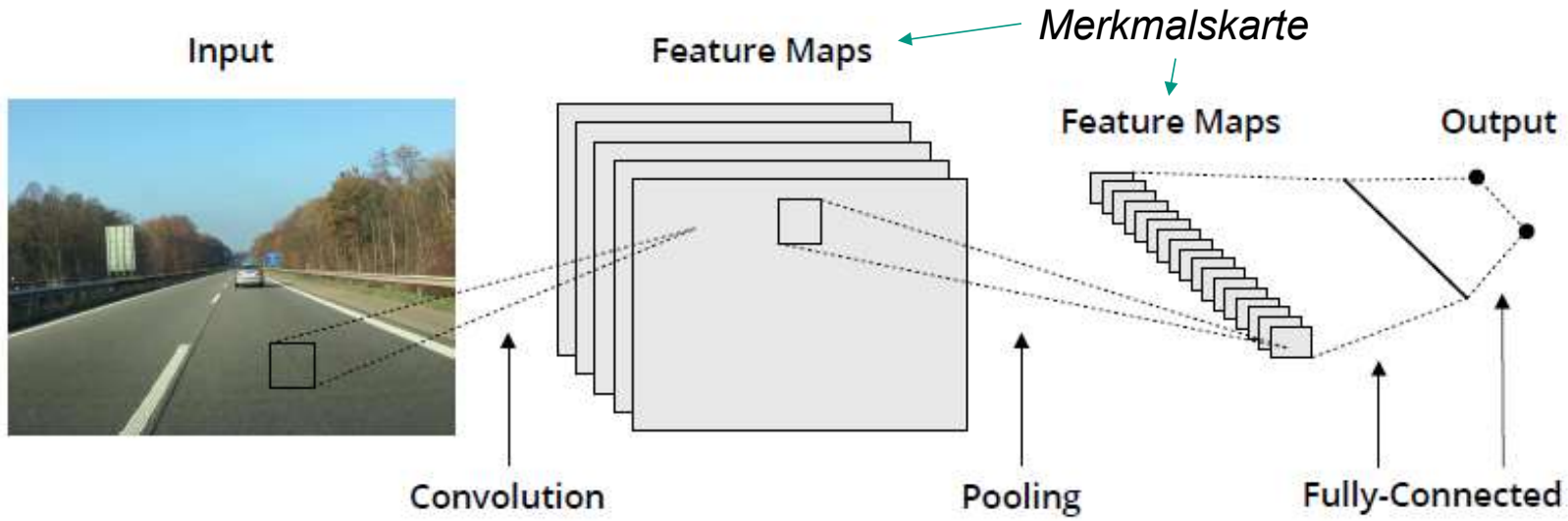




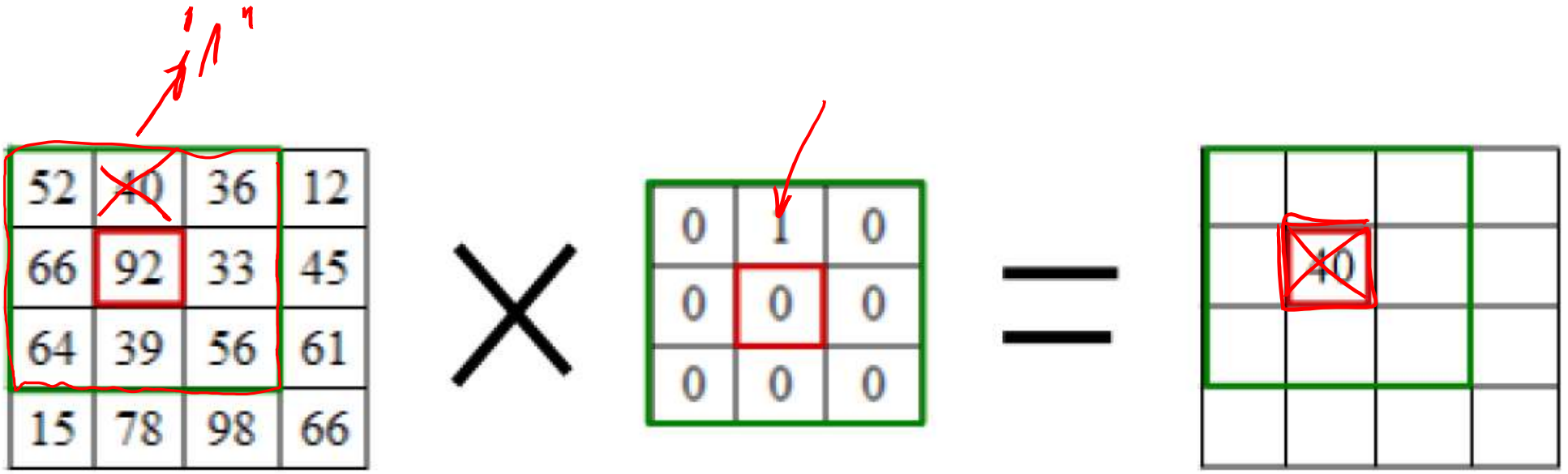
Convolutional Neural Networks (CNN)

Stand der Technik für Bilderkennung

- I.d.R.liegt die Eingabe als zwei- oder dreidimensionale Matrix (z.B. die Pixel eines Bildes) vor.
  - Dementsprechend sind die Neuronen im Convolutional Layer angeordnet.
- Die Aktivität jedes Neurons wird über eine diskrete Faltung (Konvolution von lateinisch „convolvere“ → „zusammenrollen“) berechnet.
  - Convolution-Operation (Faltung) für die Extraktion von räumlichen Merkmalen
  - Pooling-Operationen für die Dimensionsreduzierung
  - Fully-Connected-Schichten für die Klassifizierung auf Basis der extrahierten Merkmalen



Faltung
Erste Iteration



Originalmatrix
(z.B. Bild mit Grautönen)

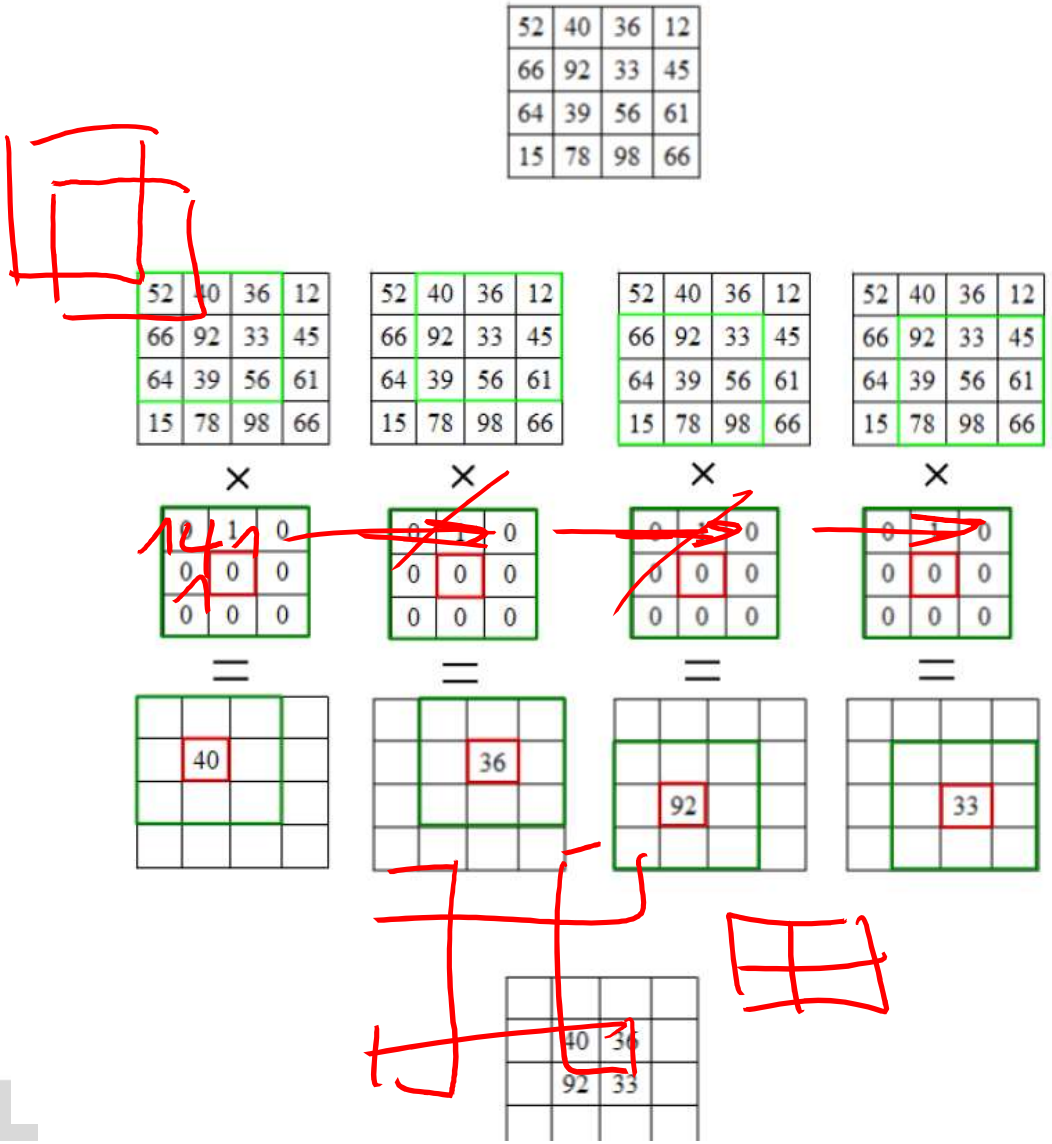
Faltungsmaske (Kernel)

Resultierende Matrix

Faltung

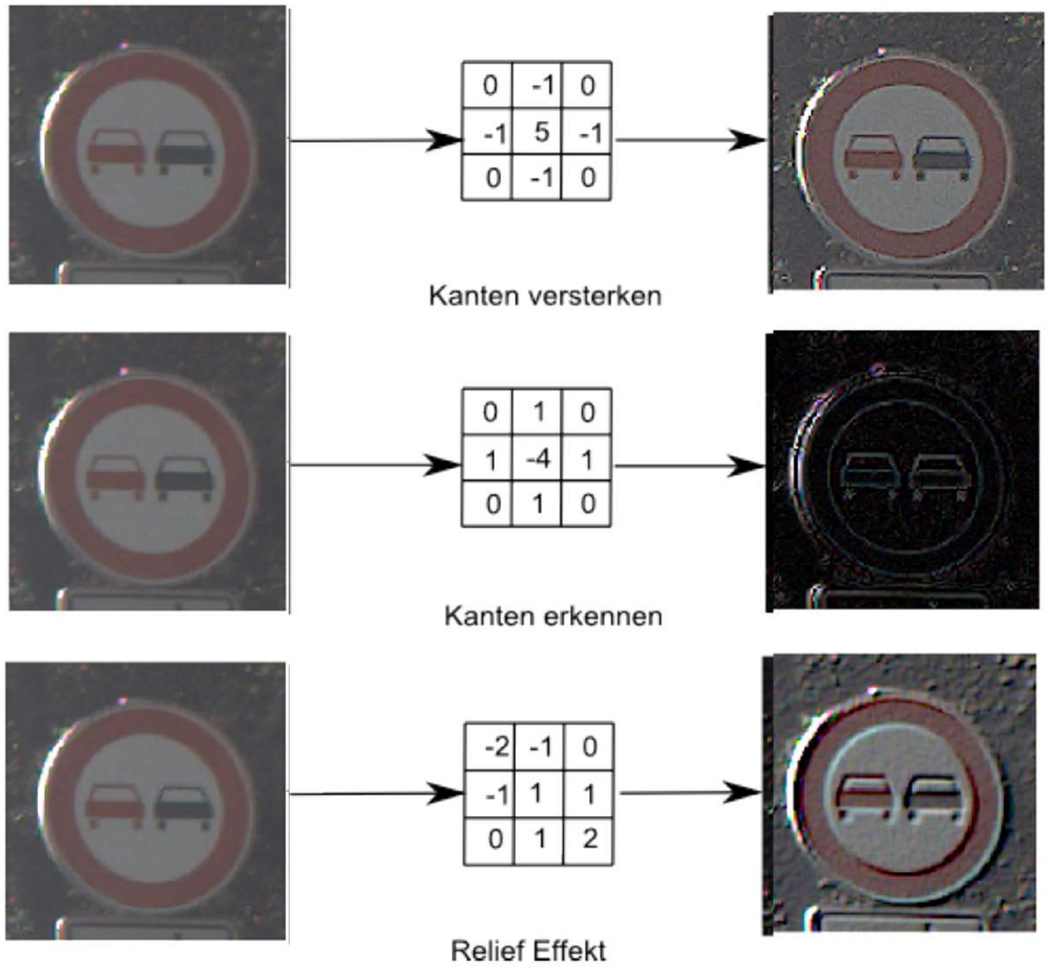
Filterung des kompletten Bildes

- Die Maske wird über alle Bildpositionen platziert.
  - Abhängig von der Kernelgröße besteht ein Bildrand von mindestens einer Pixelbreite
  - Diesen Pixeln am Rande kann kein Wert zugewiesen werden.
  - Dieser Rand kann mit verschiedenen Methoden behandelt werden, z.B kann der Rand mit Nullen oder mit den Werten benachbarter Pixel ausgefüllt werden (Padding).
- Es entsteht 1 Merkmalskarte gleicher Größe wie das Ausgangsbild.
- Mit verschiedenen Filtern entstehen verschiedene Merkmalskarten.



Beispiele von Konvolutions-Effekten

- Durch das Verwenden unterschiedlicher Masken entstehen mehrere Bildvarianten mit verschiedenen Effekten.

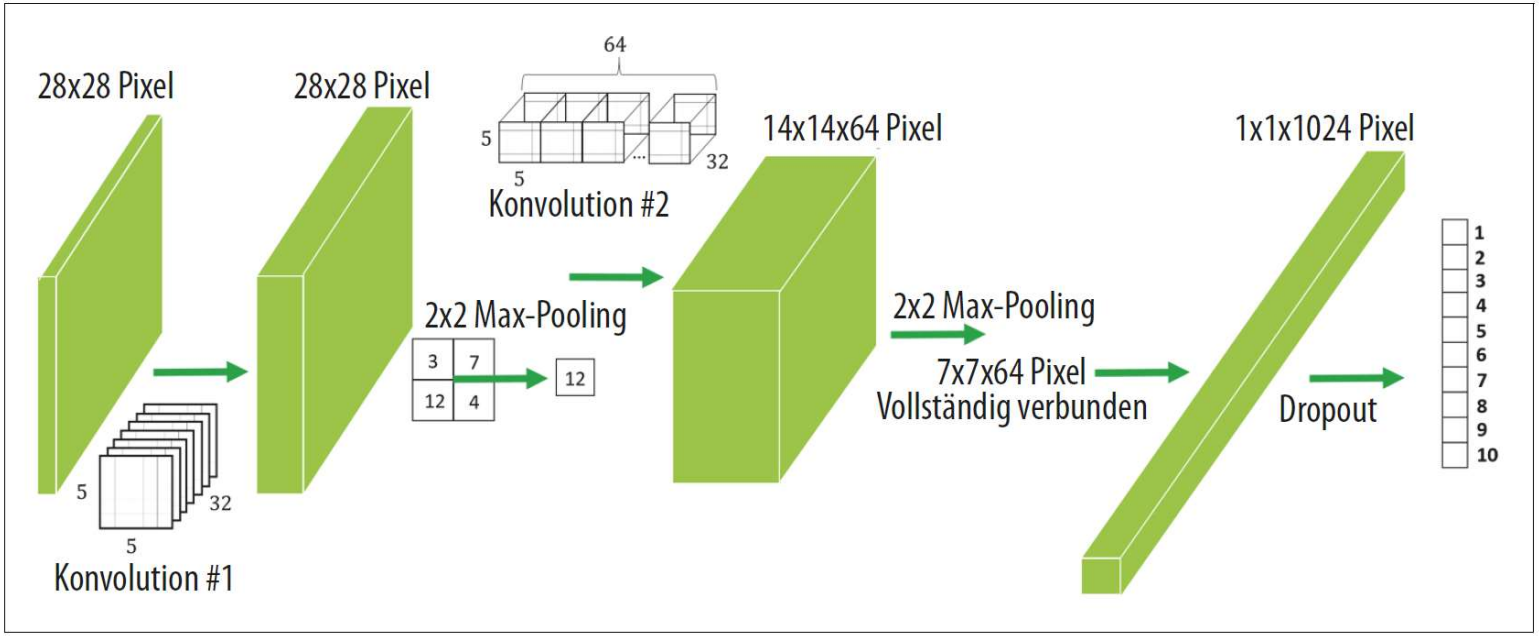


Beispiele von Konvolutions-Effekten

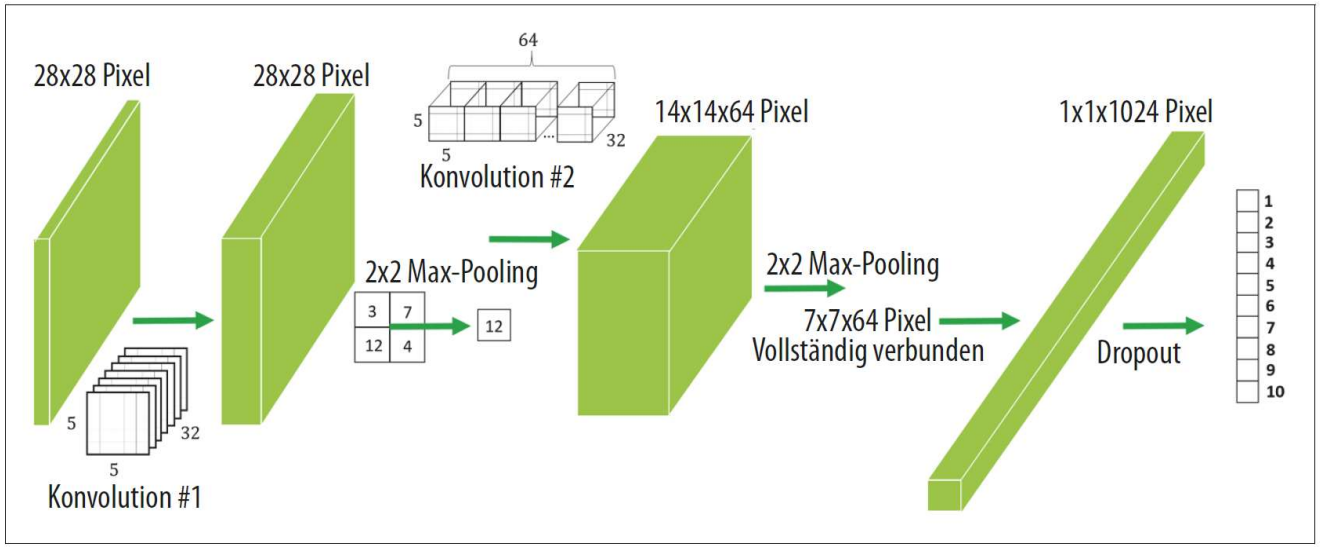
- Durch das Verwenden unterschiedlicher Masken entstehen mehrere Bildvarianten mit verschiedenen Effekten.
  - Kantendetektion
  - Scharfzeichnung
  - etc.

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

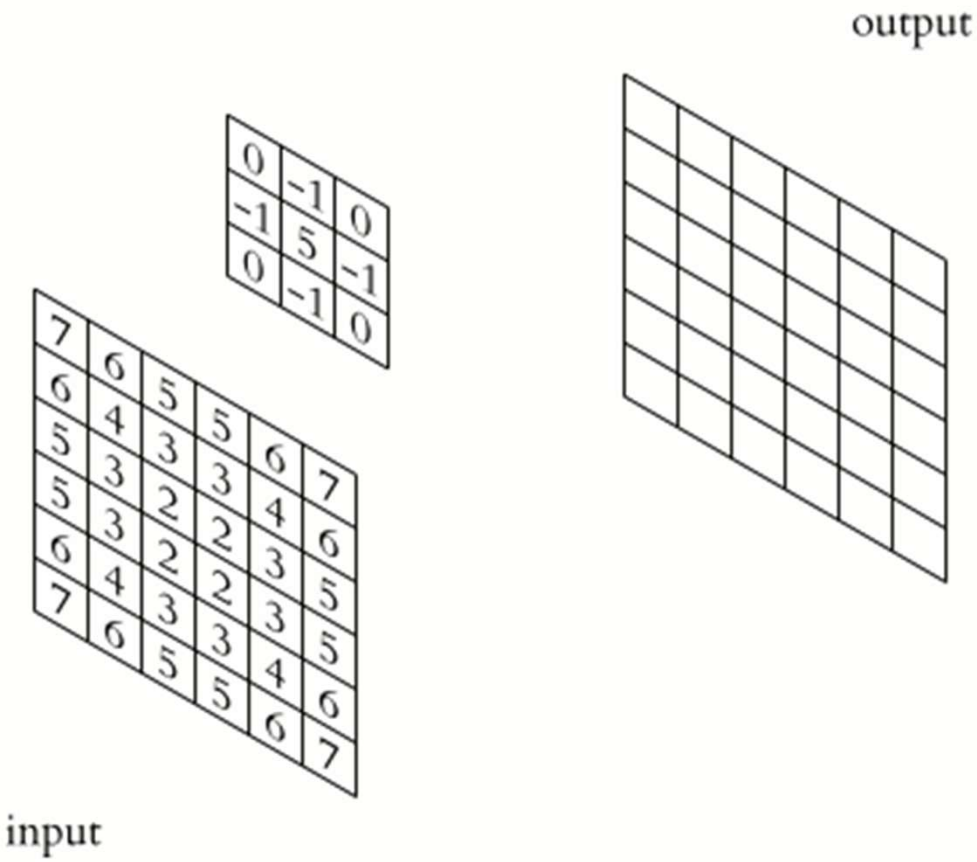
- Bilddaten in einem 2-D-Format mit der Größe 28 × 28 × 1 liegen vor.
- Als nächstes haben wir zwei aufeinanderfolgende Schichten für Konvolution und Pooling, jede davon mit 5 × 5 Konvolutionen und 64 Merkmalskarten, gefolgt von einer einzelnen vollständig verbundenen Schicht mit 1024 Einheiten.
- Vor dem Anwenden der vollständig verbundenen Schicht reihen wir alle Bildpunkte wieder zu einem einzelnen Vektor auf, da ihre räumliche Bedeutung von der vollständig verbundenen Schicht nicht mehr benötigt wird.



- Beachten Sie, dass die Größe des Bilds nach den zwei Konvolutions- und Pooling-Schichten 7 × 7 × 64 beträgt. Das ursprüngliche Bild mit 28 × 28 Pixeln wird zuerst auf die Größe 14 × 14 reduziert und anschließend von den Pooling-Operationen auf 7 × 7 verdichtet. Die 64 ist die Anzahl der in der zweiten Konvolutionsschicht erzeugten Merkmalskarten. Die Gesamtzahl der im Modell angelernten Parameter, von der sich ein Großteil in der vollständig verbundenen Schicht befindet (von 7 × 7 × 64 auf 1024), beträgt 3,2 Millionen. Ohne Max-Pooling wäre diese Zahl 16 Mal so groß (28 × 28 × 64 × 1024 oder etwa 51 Millionen).
- Die Ausgabe schließlich ist eine vollständig verbundene Schicht mit 10 Einheiten, was der Anzahl der Kategorien entspricht.



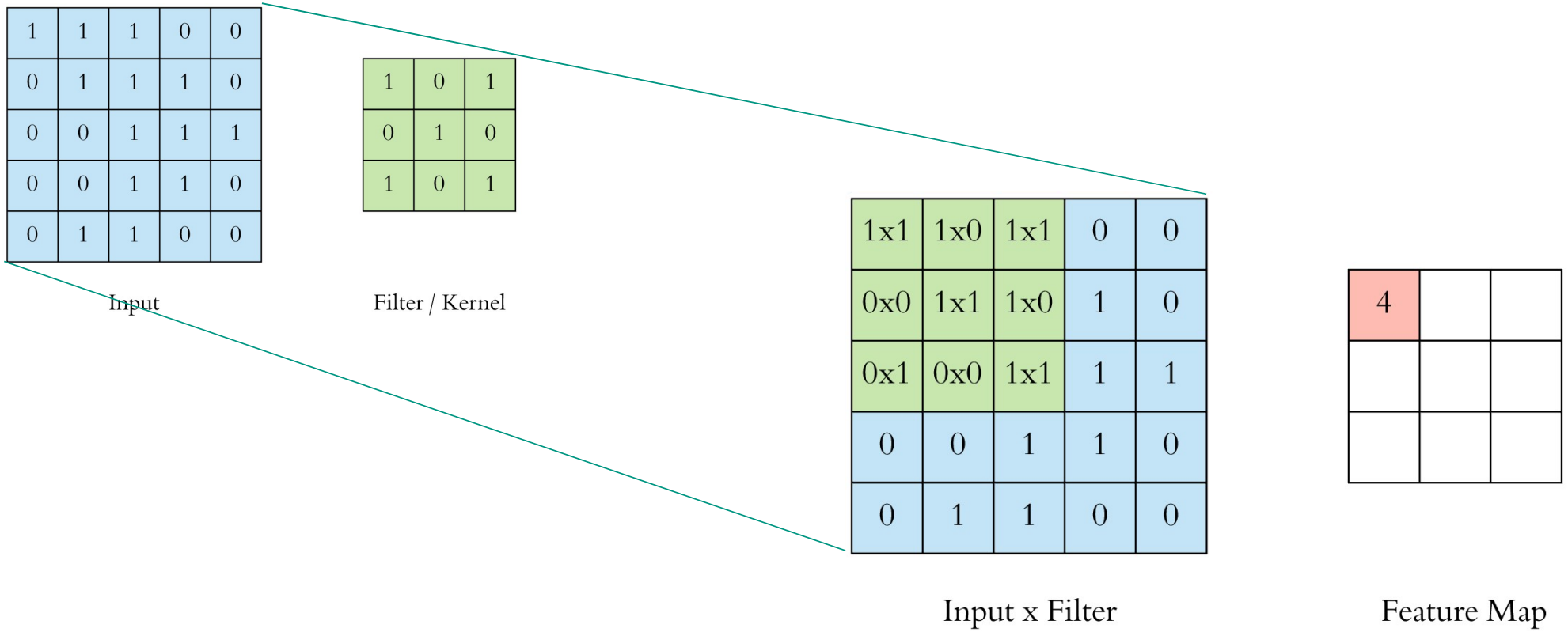
Padding



- Die Werte im Kernel werden selbstständig erlernt.
- Zur Behandlung der Randregionen des Inputs existieren verschiedene Padding-Methoden.

Filter („kleine Fenster von Gewichten“)

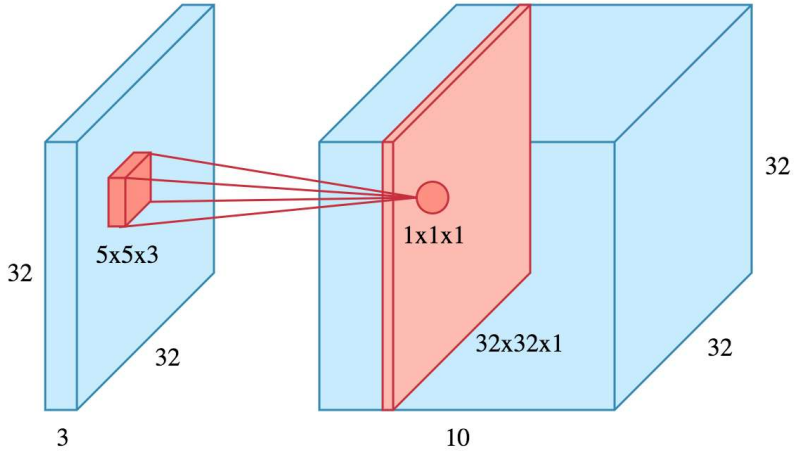
Konvolution einer 5 x 5 Bildmatrix mit einer 3x3 Filtermatrix



Pause

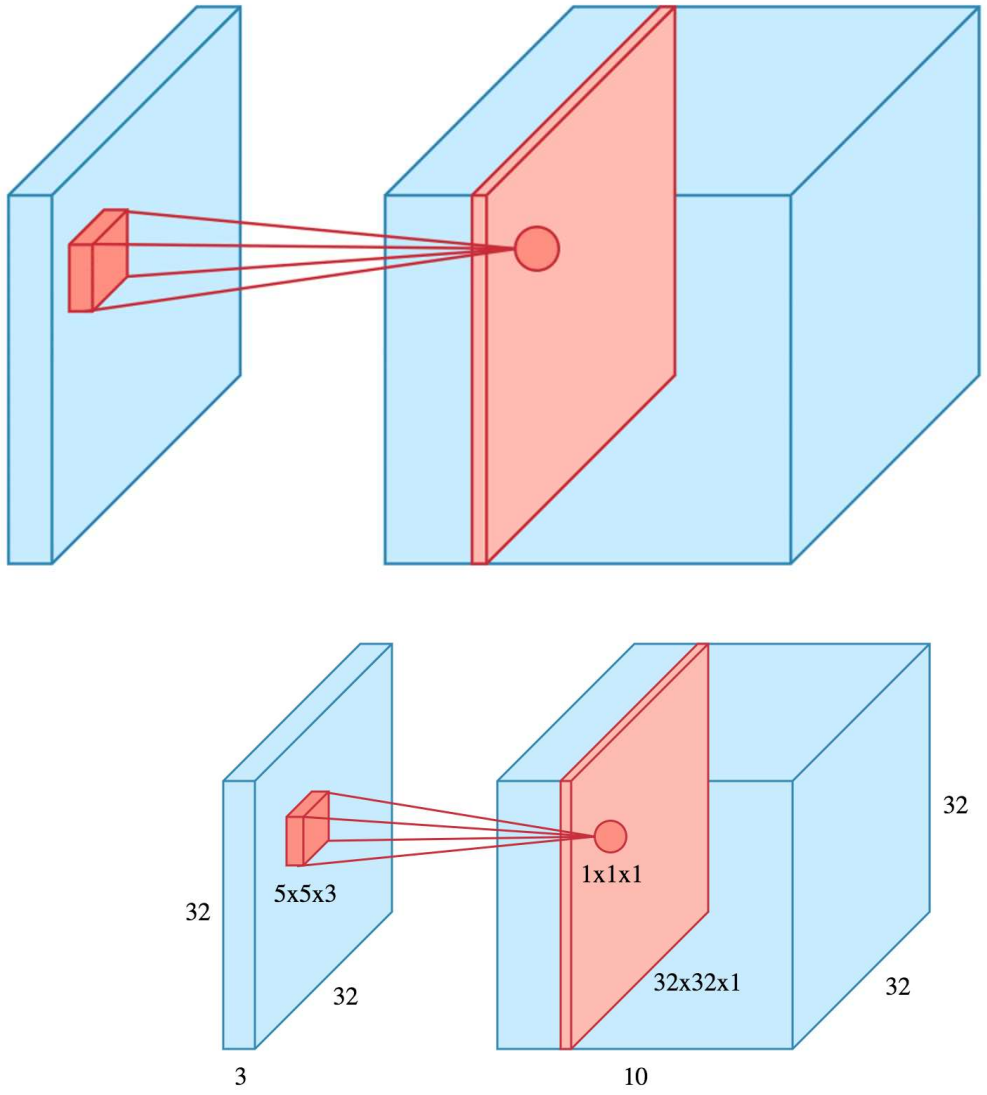
- But in reality these convolutions are performed in 3D.
- In reality an image is represented as a 3D matrix with dimensions of height, width and depth, where depth corresponds to color channels (RGB).
- A convolution filter has a specific height and width, like 3x3 or 5x5, and by design it covers the entire depth of its input so it needs to be 3D as well.
- One more important point before we visualize the actual convolution operation.
  - We perform multiple convolutions on an input, each using a different filter and resulting in a distinct feature map.
  - We then stack all these feature maps together and that becomes the final output of the convolution layer.
  - But first let's start simple and visualize a convolution using a single filter.

- Let's say we have a 32x32x3 image and we use a filter of size 5x5x3 (note that the depth of the convolution filter matches the depth of the image, both being 3).
- When the filter is at a particular location it covers a small volume of the input, and we perform the convolution operation described above.



3D vs. 2D

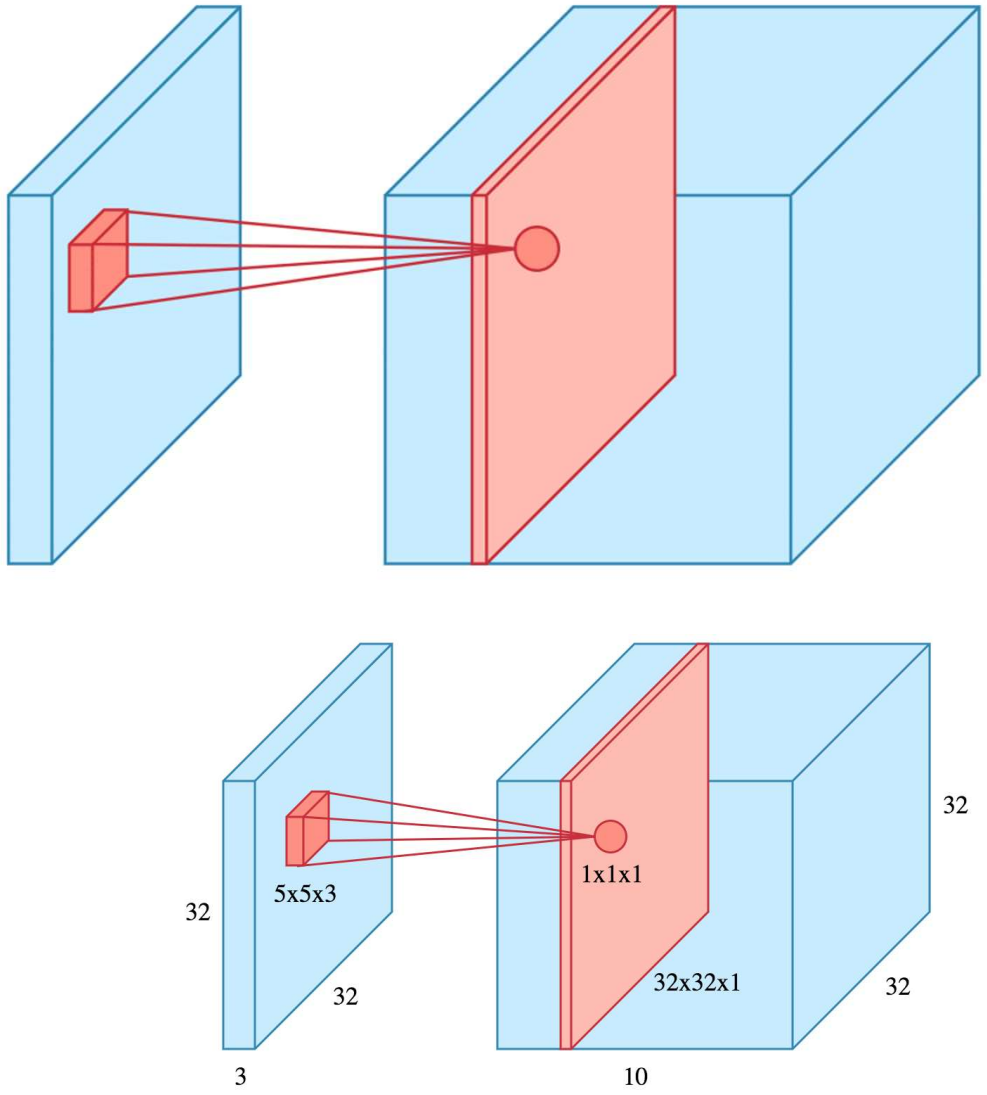
- The only difference is this time we do the sum of matrix multiply in 3D instead of 2D, but the result is still a scalar.
- We slide the filter over the input like above and perform the convolution at every location aggregating the result in a feature map.
- This feature map is of size 32x32x1, shown as the red slice on the right.
- If we used 10 different filters we would have 10 feature maps of size 32x32x1 and stacking them along the depth dimension would give us the final output of the convolution layer: a volume of size 32x32x10, shown as the large blue box on the right.
- Note that the height and width of the feature map are unchanged and still 32, it's due to padding and we will elaborate on that shortly.
- To help with visualization, we slide the filter over the input as follows. At each location we get a scalar and we collect them in the feature map. The animation shows the sliding operation at 4 locations, but in reality it's performed over the entire input.



3D vs. 2D

- In der Praxis werden Bilder mit 3D Werten für Höhe, Breite und Farbtiefe (RGB → x3) dargestellt.
- Der Unterschied ist, dass man jetzt in der Matrixkalkulation 3D an Stelle von 2D verwendet.
  - Das Ergebnis ist allerdings weiter Skalar.
- D.f. der Filter wird über die Matrixflächen verschoben.
  - Es entsteht jeweils 1 Merkmalskarte (hier 32x32x1) aus der Matrix 32x32x3
- Bei 10 Filtern entsteht ein Quader von 32x32x10, die große blaue Box rechts.
- Dass hier wieder Merkmalskarten 32x32 entstehen, hängt wieder mit dem Padding zusammen.

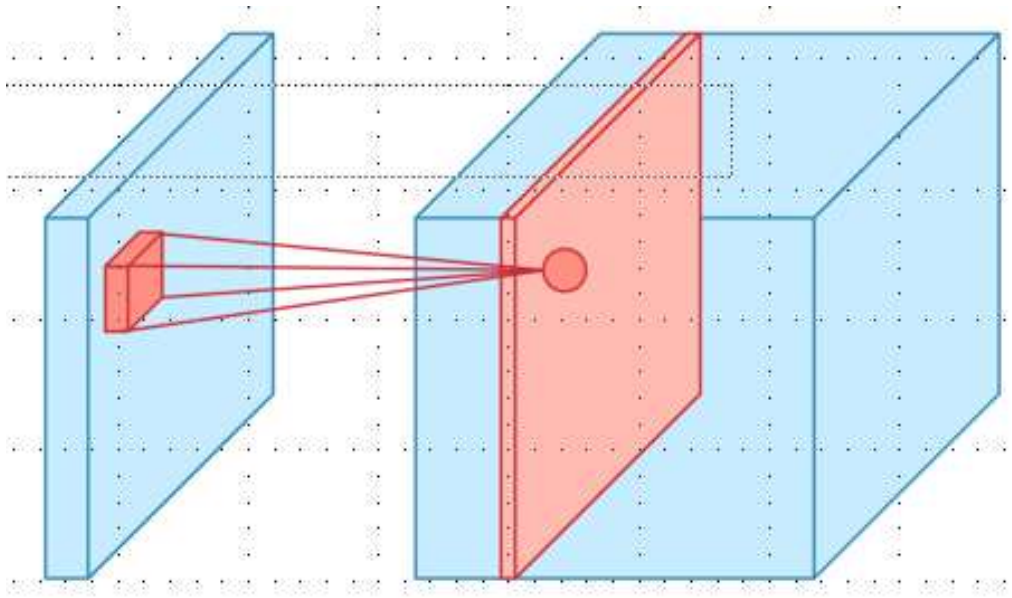
- Remark: *Die Animation findet repräsentativ für 4 Felder der Matrix statt. Real natürlich für alle Felder.*

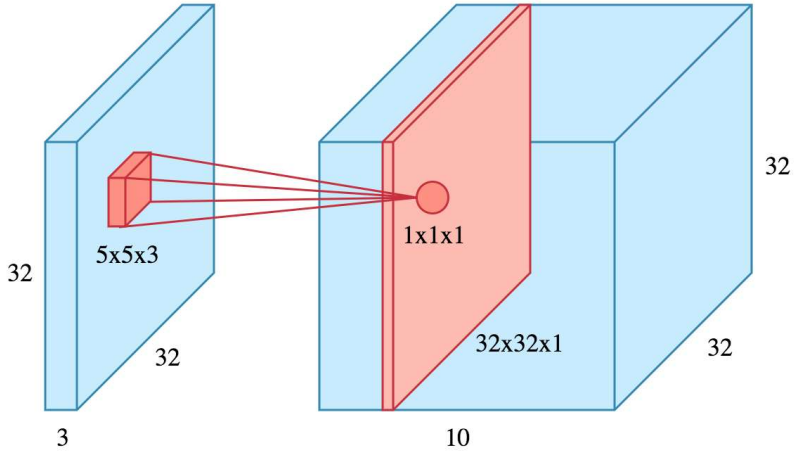


3D vs. 2D

- In der Praxis werden Bilder mit 3D Werten für Höhe, Breite und Farbtiefe (RGB → x3) dargestellt.
- Der Unterschied ist, dass man jetzt in der Matrixkalkulation 3D an Stelle von 2D verwendet.
  - Das Ergebnis ist allerdings weiter Skalar.
- D.f. der Filter wird über die Matrixflächen verschoben.
  - Es entsteht jeweils 1 Merkmalskarte (hier 32x32x1) aus der Matrix 32x32x3
- Bei 10 Filtern entsteht ein Quader von 32x32x10, die große blaue Box rechts.
- Dass hier wieder Merkmalskarten 32x32 entstehen, hängt wieder mit dem Padding zusammen.

- Remark: *Die Animation findet repräsentativ für 4 Felder der Matrix statt. Real natürlich für alle Felder.*

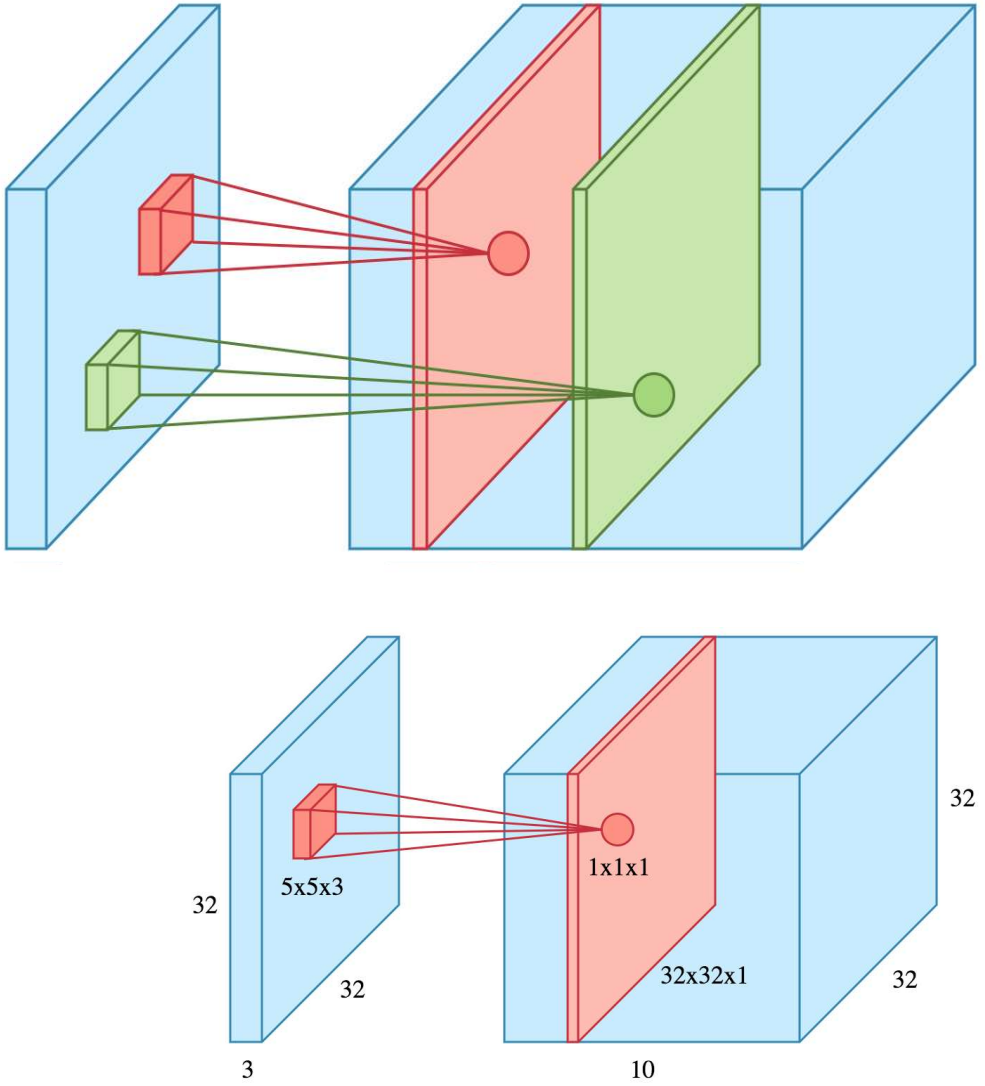




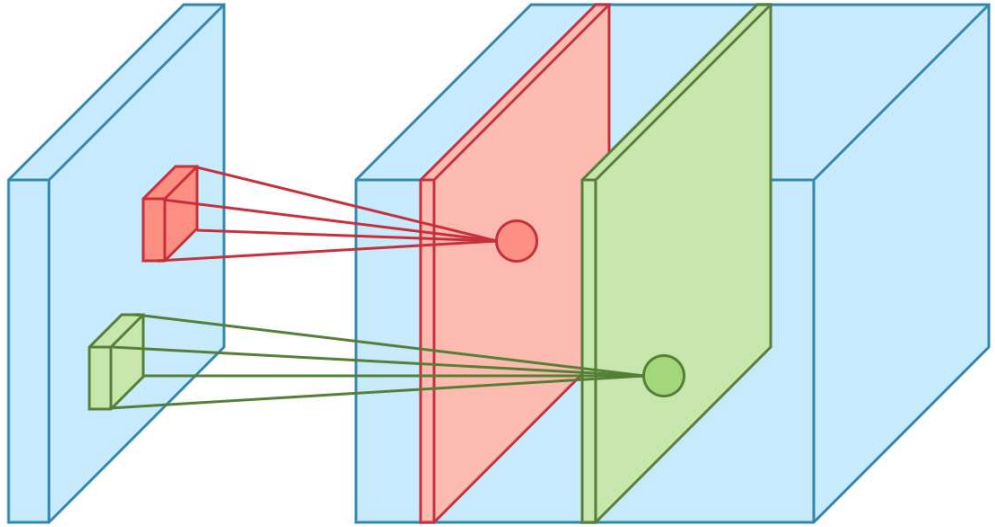
3D vs. 2D

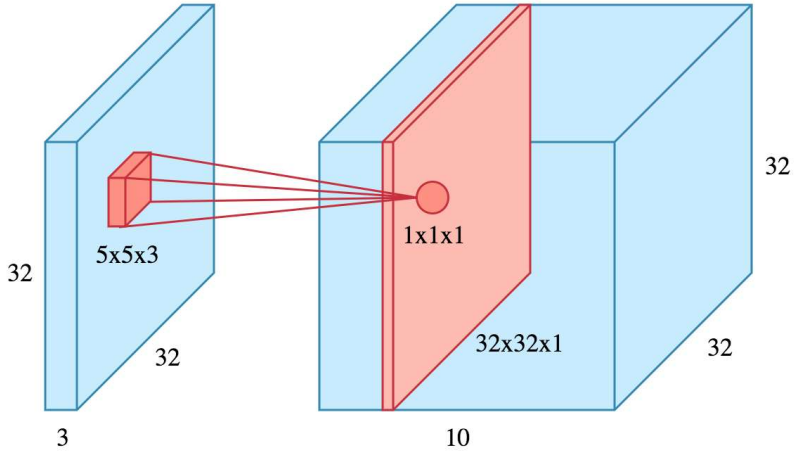
- In der Praxis werden Bilder mit 3D Werten für Höhe, Breite und Farbtiefe für RGB dargstellt.
- Der Unterschied ist, da man jetzt in der Matrixkalkulation 3D an Stelle von 2D verwendet.
  - Das Ergebnis ist allerdings weiter Skalar.
- Das Filter wird über die Matrixflächen verschoben.
- Es entsteht jeweils 1 Merkmalskarte (hier 32x32x1) aus der Matrix 32x32x3
- Bei 10 Filtern entsteht ein Quader von 32x32x10, die große blaue Box rechts.
- Dass hier wieder Merkmalskarten 32x32 entstehen, hängt wieder mit dem Padding zusammen.

- Remark: *Die Animation findet repräsentativ für 4 Felder der Matrix statt. Real natürlich für alle Felder.*



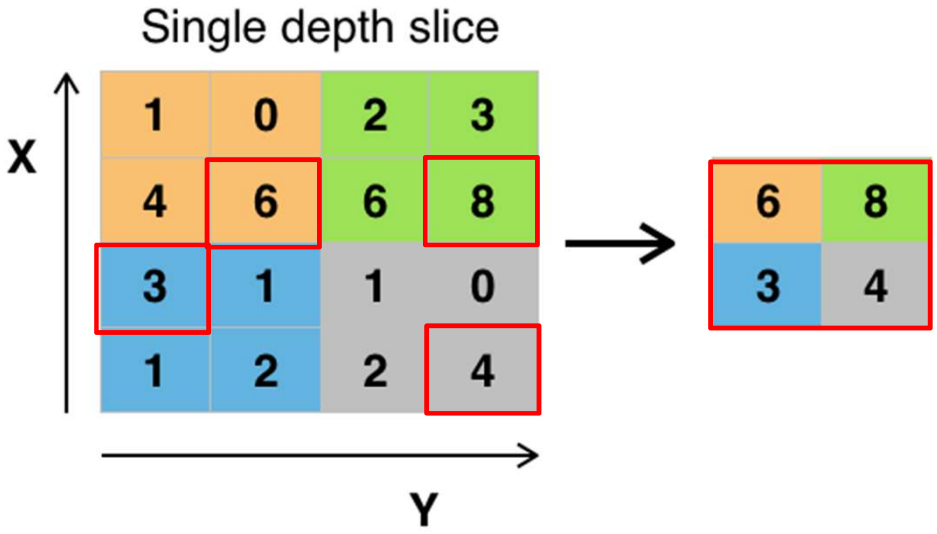
- Below we can see how two feature maps are stacked along the depth dimension.
- The convolution operation for each filter is performed independently and the resulting feature maps are disjoint.





Pooling

Verwerfen überflüssiger Informationen

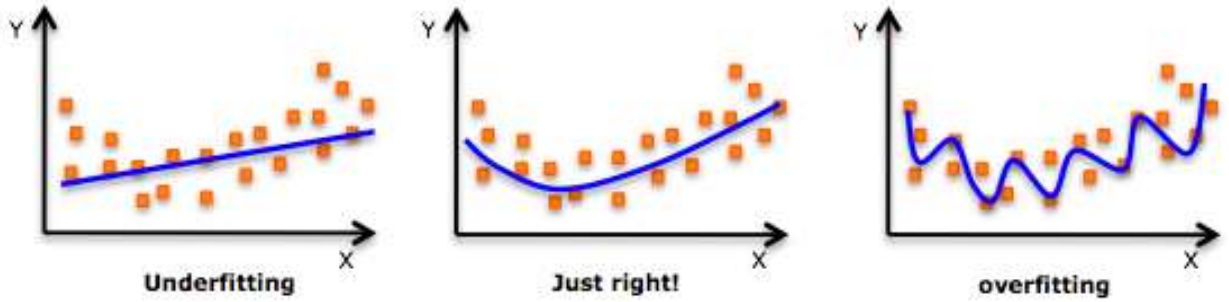


- Zur Objekterkennung in Bildern ist die *exakte* Position einer Kante im Bild von vernachlässigbarem Interesse → die ungefähre Lokalisierung ist hinreichend.

- Es gibt verschiedene Arten des Poolings.
  - Am stärksten verbreitet ist das Max-Pooling, wobei aus jedem 2 × 2 Quadrat aus Neuronen des Convolutional Layers nur die Aktivität des aktivsten (daher "Max") Neurons für die weiteren Berechnungsschritte beibehalten wird.
  - Die Aktivität der übrigen Neuronen wird verworfen.
  - Vorteile:
    - Verringerter Platzbedarf und erhöhte Berechnungsgeschwindigkeit
    - Daraus resultierende Möglichkeit zur Erzeugung tieferer Netzwerke, die komplexere Aufgaben lösen können.
    - Präventionsmaßnahme gegen Overfitting
- Alternativen wie das Mean-Pooling haben sich in der Praxis als weniger effizient erwiesen.

Overfitting/Underfitting eines Modells

- Overfitting
  - Trainiertes Modell ist "zu gut" trainiert und nun zu eng mit dem Trainingsdatensatz übereinstimmend.
  - Dies geschieht in der Regel, wenn das Modell zu komplex ist (d.h. zu viele Merkmale/Variablen im Vergleich zur Anzahl der Beobachtungen).
  - Das Modell wird sehr genau auf die Trainingsdaten sein, aber wahrscheinlich sehr ungenau auf untrainierte oder neue Daten.
- Underfitting
  - Im Gegensatz zum Overfitting bedeutet die Underfitting eines Modells, dass das Modell nicht zu den Trainingsdaten passt und somit die Muster in den Daten nicht erkennt.
  - Es bedeutet auch, dass das Modell nicht auf neue Daten verallgemeinert werden kann.



Grenzen

- Neuronale Netze sind in ihrem Einsatzgebiet und in ihrer Ausgestaltung sehr flexibel.
- Sie werden für Klassifizierungen, Prognosen und Clustering-Aufgaben angewendet.
- Insbesondere im Bereich der Bild-, Schrift- und Spracherkennung werden sie erfolgreich eingesetzt.
- Kritisch gesehen wird, dass neuronale Netze eine Black Box darstellen, deren Ergebnisse manchmal nur schwer zu erklären sind.
  - Die Gestaltung eines Netzwerkes wird oft als willkürlich angesehen und es besteht die Gefahr der Überanpassung (Overfitting) des Netzes an die Trainingsdaten.
  - Das Netz liefert dann perfekte Ergebnisse für die Trainingsdaten, stellt sich aber ungeeignet für „neue“ Daten heraus.
- Je komplizierter (tiefer) die Struktur der Netze ist, umso rechenaufwendiger werden die Verfahren.
- Dank der gestiegenen Rechenleistung aktueller Computer und immer neuer Deep-Learning-Bibliotheken erfahren neuronale Netze aber derzeit geradezu eine Renaissance in ihrer Anwendung.
  - *Bei aller Euphorie sollte man in der Erwartungshaltung aber auf dem Boden bleiben.*
- Man darf nicht vergessen:
  - Ein menschliches Gehirn besteht aus 80 bis 100 Milliarden Neuronen. Davon sind künstliche neuronale Netze noch „meilenweit“ entfernt.

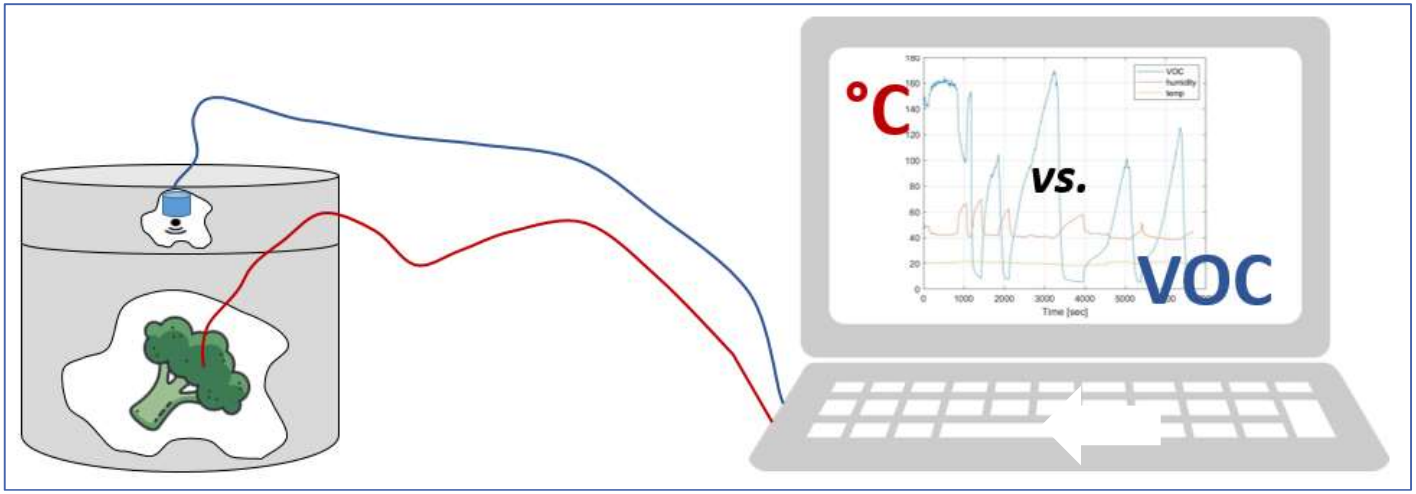


WMF Group – a Groupe SEB company

EXPERIMENTAL SETUP

Recognition of the state of food

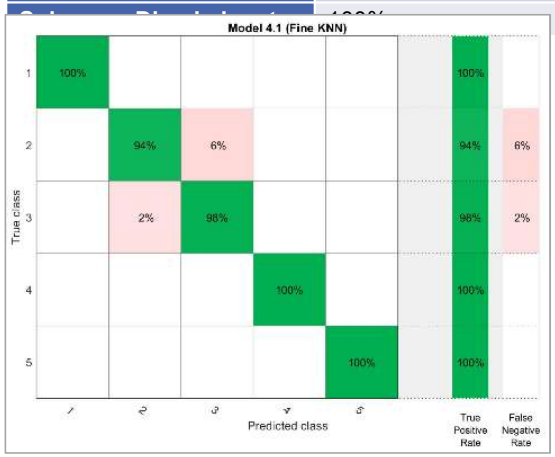
				
Tomato	**Pepper**	**Celery**	**Carrot**	**Apple**
2-3min	*3-5min*	*4-6min*	*5-8min*	*2-5min*
Washed & cut	*Washed & cut*	*Peeled & cut*	*Peeled & cut*	*Washed & cut*
				

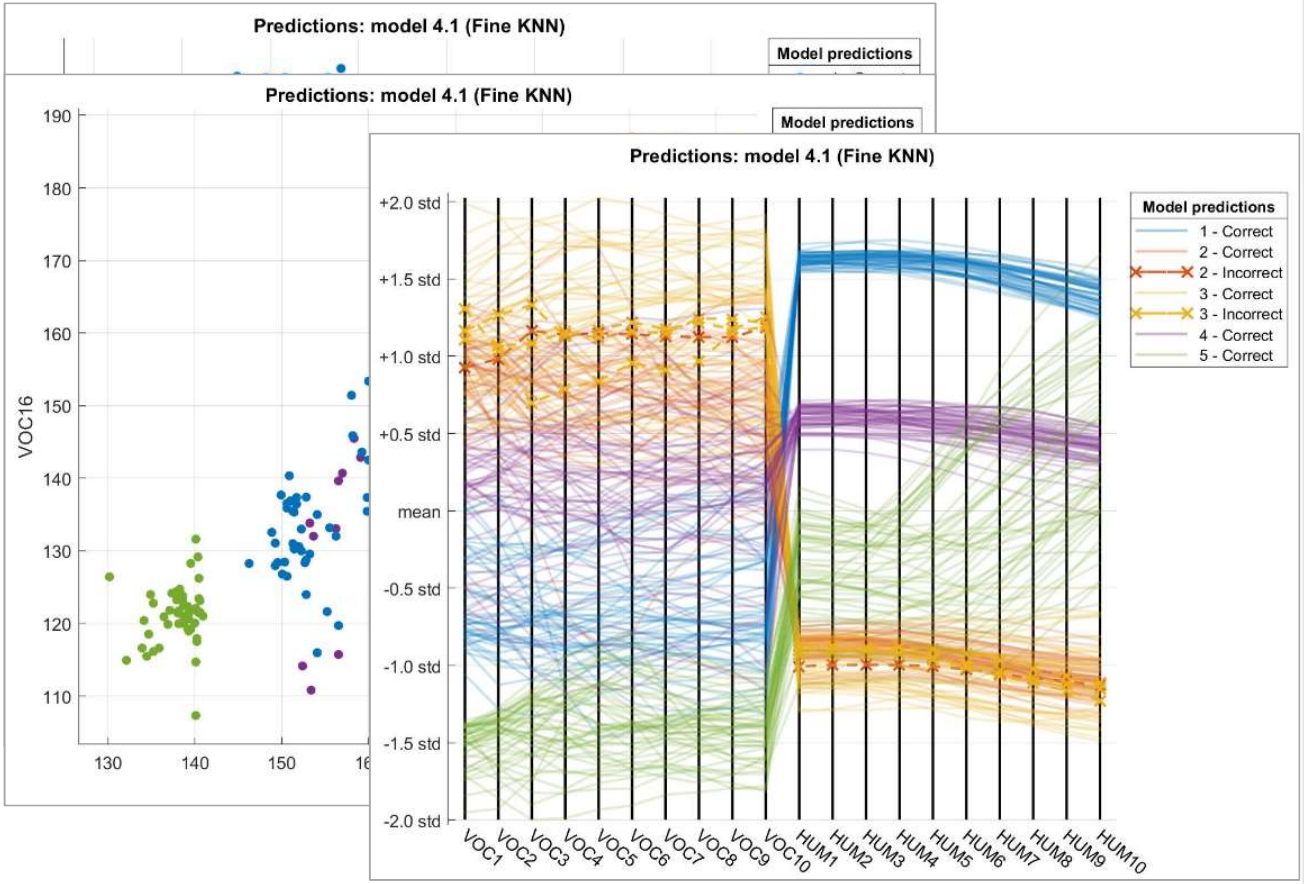


MODELLING OF ML-CONCEPT

Prediction results of classification learner

Model	Accuracy (PCA off)	Accuracy (PCA on)
Fine Tree	96.4%	88.4%
Medium Tree	96.4%	88.4%
Linear Discriminant	100%	90.4%
Quadratic Discriminant	99.6%	91.2%
Fine KNN	98.4%	92.4%
Medium KNN	96%	89.2%
Weighted KNN	98.4%	91.6%
Linear SVM	99.6%	90.8%
Quadratic SVM	100%	91.6%
Cubic SVM	99.6%	90.4%
Fine Gaussian SVM	96%	91.2%
Medium Gaussian SVM	99.2%	91.2%
Coarse Gaussian SVM	98.8%	71.6%
		87.6%

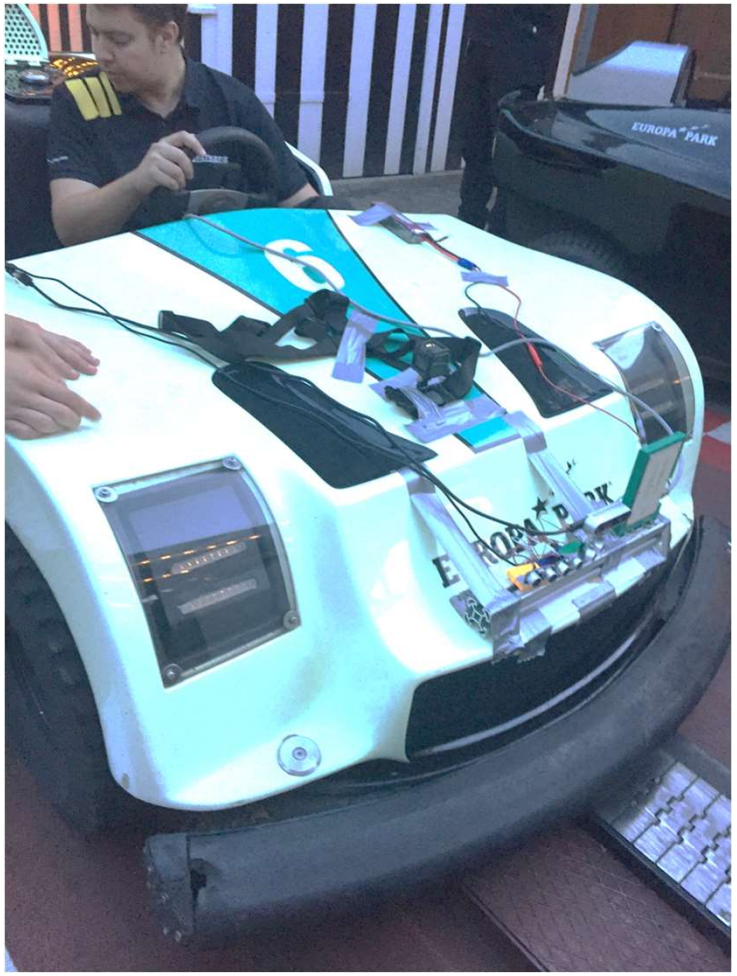





6




EUROPA PARK


EUROPA PARK



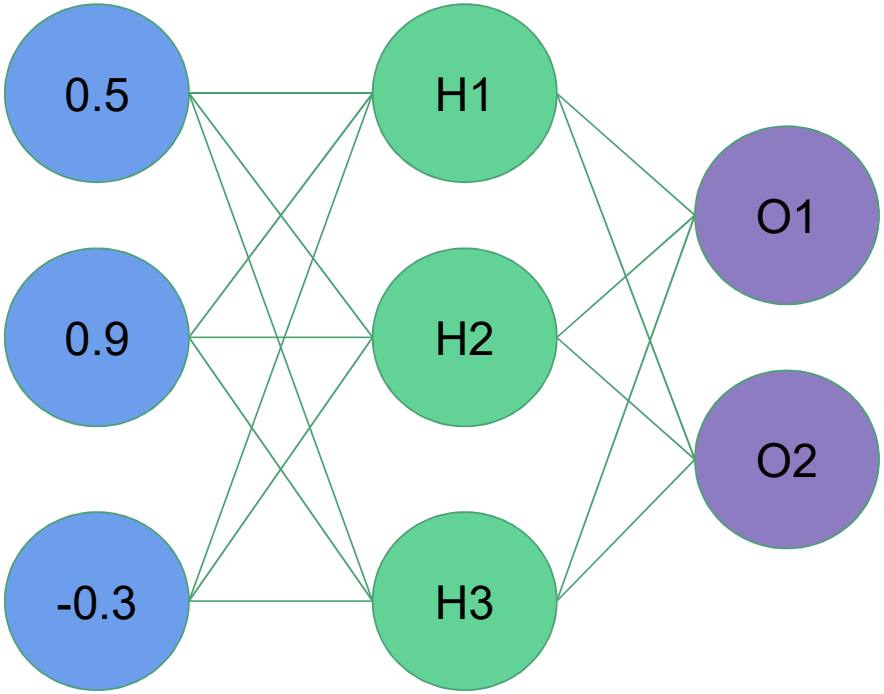


Backup

Beispiele

- Im Haushalt können Neuronale Netze bei Regelungsaufgaben wie der Benutzung einer Waschmaschine oder bei der Steuerung eines Staubsaugerroboters zum Einsatz kommen. Dieser kann mit Hilfe eines solchen Netzes seine Umgebung erlernen und kann bei jedem Saugvorgang seine abgefahrenen Pfade verbessern. Natürlich bleibt Ihre Wohnung nicht vollständig gleich. Hier wird einmal ein Tisch verrückt. Dort bekommt das Sofa einen neuen Standort. Da sich jedoch meist nicht alles gleichzeitig verändern wird, kann die Robotersteuerung durch ein Neuronales Netz die alten Kenntnisse, die sich noch verwenden lassen, weiter behalten und die neuen Kenntnisse zusätzlich lernen. Das heißt, mit Hilfe eines Neuronalen Netzes verblasst seine Erinnerung mit der Zeit und neue Informationen stellen sich in den Vordergrund, so dass er stets das momentan optimale Ergebnis liefert.
- Neuronale Netze können zur Optimierung einer Robotersteuerung eingesetzt werden. Industrieroboter z.B. bei der Produktion von Automobilen haben die Eigenschaft, dass sie aus einer Vielzahl von Gelenken bestehen, mit denen auch ein kompliziertes hineinreichen z.B. in eine Karosserie durchführbar ist. Die Gelenke können jedoch schnell unter Materialermüdung leiden, besonders wenn sie ruckartig bewegt werden und hohe Fliehkräfte wirken. Es geht darum, den Roboterarm möglichst schnell und möglich schonend zu bewegen, so dass alle Arbeiten geleistet werden können. Hierfür können neuronale Netze genutzt werden. Während des Betriebes verändert aufgrund von Abnutzungen der Roboterarm seine Eigenschaften. Eine Korrektur in dem Computerprogramm muss anhand verschiedener Parameter nachgehalten werden. Da sich die Parameter nur langsam ändern, kann auch hierfür ein Neuronales Netz genutzt werden.

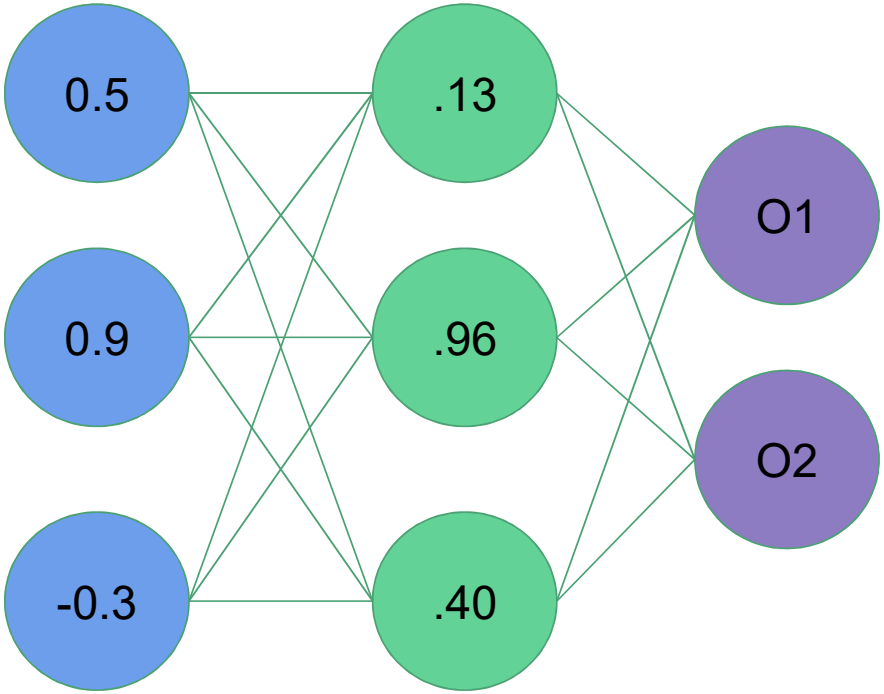
Inferenz
Neuronale Netze



H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

O1 Weights = (-3.0, 1.0, -3.0)
O2 Weights = (0.0, 1.0, 2.0)

Inferenz
Neuronale Netze

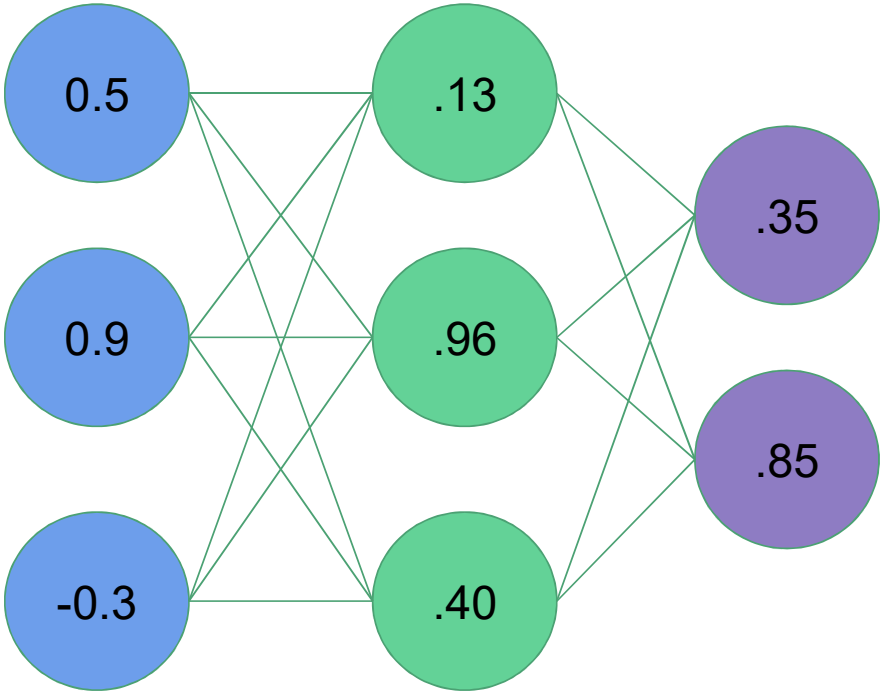


H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

O1 Weights = (-3.0, 1.0, -3.0)
O2 Weights = (0.0, 1.0, 2.0)

H1 = S(0.5 * 1.0 + 0.9 * -2.0 + -0.3 * 2.0) = S(-1.9) = .13
H2 = S(0.5 * 2.0 + 0.9 * 1.0 + -0.3 * -4.0) = S(3.1) = .96
H3 = S(0.5 * 1.0 + 0.9 * -1.0 + -0.3 * 0.0) = S(-0.4) = .40

Neuronale Netze
Inferenz



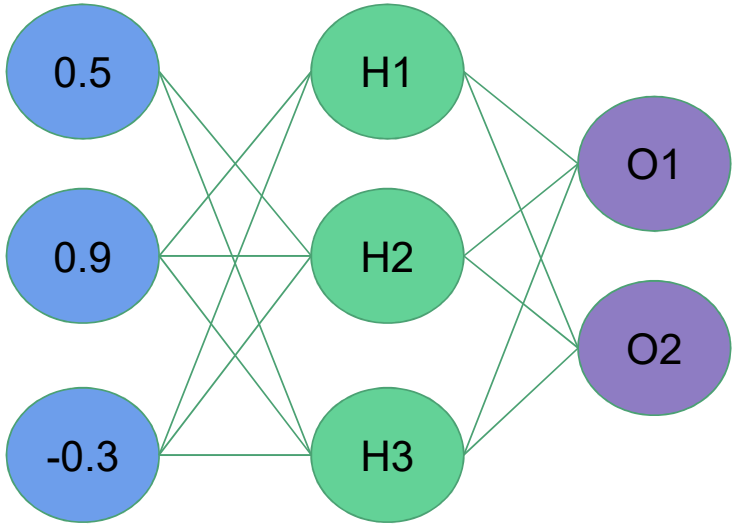
H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

O1 Weights = (-3.0, 1.0, -3.0)
O2 Weights = (0.0, 1.0, 2.0)

O1 = S(.13 * -3.0 + .96 * 1.0 + .40 * -3.0) = S(-.63) = .35
O1 = S(.13 * 0.0 + .96 * 1.0 + .40 * 2.0) = S(1.76) = .85

Matrix-Formulierung der Inferenz

Neuronale Netze



H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

Hidden Layer Weights | Inputs

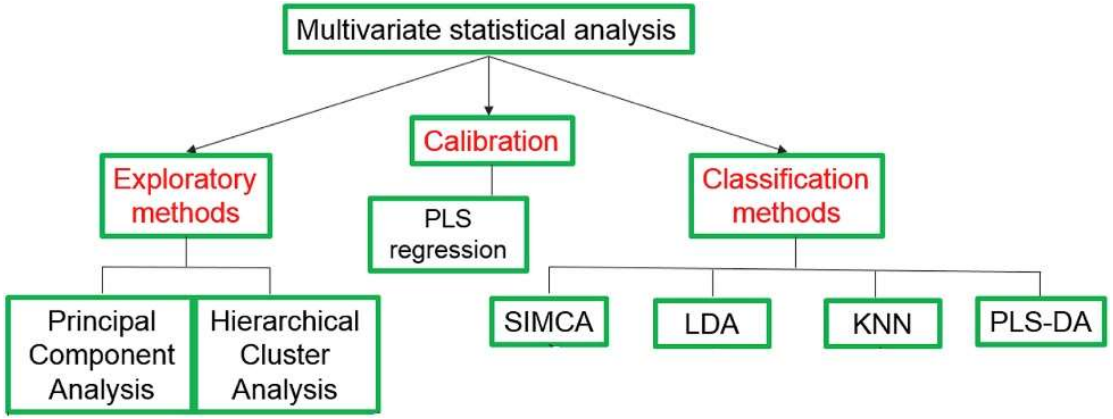
$$S\left(\begin{bmatrix} 1.0 & -2.0 & 2.0 \\ 2.0 & 1.0 & -4.0 \\ 1.0 & -1.0 & 0.0 \end{bmatrix} * \begin{bmatrix} 0.5 \\ 0.9 \\ -0.3 \end{bmatrix}\right) = S\left(\begin{bmatrix} -1.9 & 3.1 & -0.4 \end{bmatrix}\right) = \begin{bmatrix} .13 & .96 & 0.4 \end{bmatrix}$$

Hidden Layer Outputs

Unüberwachtes Lernen

„unsupervised learning“

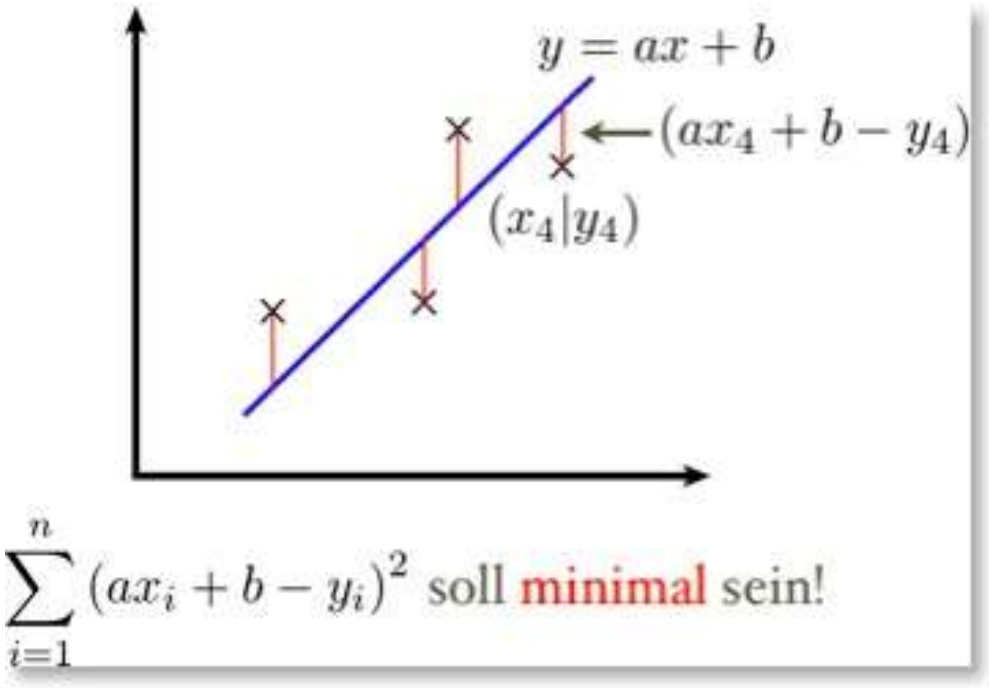
- Klassische Ballungen
  - k-means-Clustering
  - Agglomerative Hierarchical Clustering
- Begriffliche Ballungen („Conceptual Clustering“)
  - CLUSTER/2
  - Bildung von Begriffshierarchien („Concept Formation“)
- COBWEB
- CLASSIT
- Lernen durch Entdeckung
- BACON
- ABACUS
- ...
- Ohne Belohnung
- Ohne Vorwissen



Gauß'schen Methode der kleinsten Quadrate

Am Beispiel von 4 Messwerten

- Es soll eine passende Gerade (blau) gesucht werden.
- Nach einer Idee von **Carl-Friedrich Gauß** verwendet man als Maß für die Anpassung zunächst die vertikalen Abstände zwischen den tatsächlichen Messwerten und den entsprechenden Punkten auf der blauen Geraden.
- Für den vierten Punkt ist dieser Abstand angegeben.
- Diese Abstände werden quadriert und dann für alle Messwerte aufsummiert.
- Nach Gauß ist die beste Ausgleichslinie die, für die **die Summe der Quadrate der Abweichungen minimal ist**.
- Diese Idee heißt:
  - **Methode der kleinsten Fehlerquadrate**.
  - Das Stichwort „minimal“ deutet darauf hin, dass es sich um eine **Extremwertaufgabe** handelt.
  - Die Parameter a und b der Geraden müssen so gewählt werden, dass die bewusste Summe minimal wird.



Gütemaße für die Evaluation von Modellen

- Es existiert eine Reihe von Verfahren zum Testen der Güte statistischer Modelle.
- Die meisten solcher Verfahren liefern Kennzahlen, deren Quantität eine Aussage über die Modellgüte treffen. Ein Beispiel hierfür ist das Bestimmtheitsmaß R^2.
  - Dieses Maß erklärt den Anteil einer Variablen Y an der durch ein statistisches Modell erklärten Varianz.
  - Andere Gütetests beziehen sich auf die Beurteilung binärer Klassifikatoren.
  - Als Beispiel sei die Verifikation numerischer Wettervorhersagemodelle genannt.
  - An ihnen wird u.a. untersucht, wie genau ein vorhergesagter Parameter mit tatsächlich eingetretenen Ereignissen übereinstimmt (z.B. Niederschlag ja/nein).
  - Es werden Häufigkeitszahlen bestimmt, die angeben, wie oft eine Vorhersage bei eingetretenen Ereignissen korrekt war und wie oft inkorrekt.
  - Äquivalent dazu das Ganze für Nicht-Ereignisse.
  - Aus den erhaltenen vier Häufigkeiten lassen sich kategorische Gütemaße definieren.
  - Als Beispiele seien die Kennzahlen
    - Probability Of Detection
    - False Alarm Rate
    - True Skill Statisticsgenannt
  - Kontinuierliche Gütemaße beziehen sich dagegen auf kontinuierliche Parameter wie z.B. die Temperatur.
  - Hier findenübliche statistische Maße wiemittlerer Fehler,mittlerer absoluter Fehler,RMSE[1]unddieStandardabweichung Anwendung
  - Die Methodik, von der einige Verfahren erwähnt wurden, bezieht sich u.a. immer auf die Analyse eines kompletten Datensatzes, auf dem ein statistisches Modell beruht.
- Einen anderen Ansatzverfolgt dagegen die Kreuzvalidierung.
  - Ein Datensatz wird in mehrere Teile aufgeteilt.
  - Auf Basis eines Teils des Datensatzes wird ein statistisches Modellabgeleitet.
  - Das Modell wird auf den Rest des Datensatzes angewendet und die Abweichung zu den tatsächlichen Werten untersucht.
  - Das Verfahren dient also in erster Linieder Überprüfung derPrognosegüteeines Modells.

Gütemaße für die Evaluation von Modellen

Regression

- **R^2- Metric (R-Squared)**
  - Der Wert der R^2 kann Werte zwischen 0 und 1 annehmen
  - Nahe 1: Vorhersagen des Models sind gut
  - Nahe 0: Vorhersagen sind nicht besser als raten

- $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$

$y_i = vorhergesagte\ Werte$

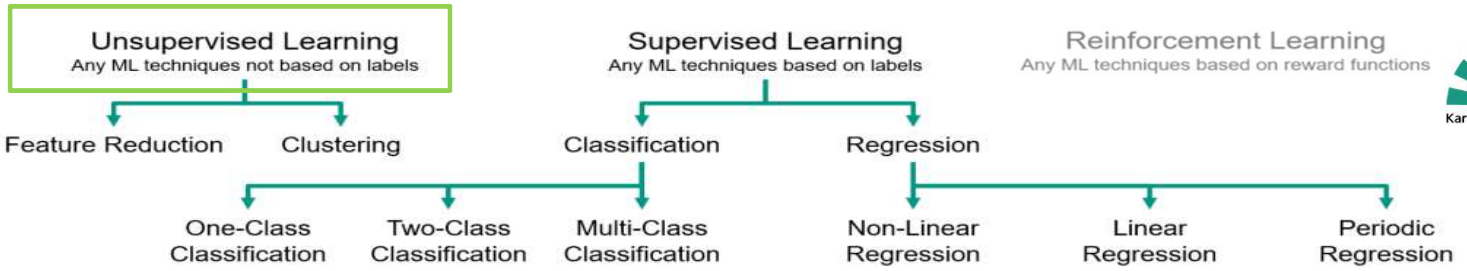
$\tilde{y}_i = korrekte\ Werte$

$\bar{y} = Median\ der\ Werte$

Übungen

Veranstaltungsplan Informationstechnik 2 SS2019						
Vorlesung/Übung:		Mi, 8:00 - 9:30 Uhr		Gebäude 30.22, Gaede-Hörsaal		
Vorlesung/Übung:		Fr, 9:45 - 11:15 Uhr		Gebäude 30.22, Gaede-Hörsaal		
Woche	**Datum Mi.**	**Vorlesung**	**Datum Fr.**	**Übung**	**Inhalt Mi.**	**Inhalt Fr.**
1.	24. Apr		26. Apr	VL		VL0: Einleitung und Motivation
2.	01. Mai	frei	03. Mai	VL	Tag der Arbeit	VL1: Algorithmische Grundlagen
3.	08. Mai	ÜB	10. Mai	VL	ÜB1: Organisatorisches; Besprechung Aufg. 1;	VL2: Sortieralgorithmen
4.	15. Mai		17. Mai	VL		VL3: Algorithmen auf Graphen
5.	22. Mai	VL	24. Mai	frei	VL4: Optimierungsalgorithmen	frei
6.	29. Mai	ÜB	31. Mai	VL	ÜB2: Besprechung Aufg. 2;	VL5: Big Data Einführung
7.	05. Jun	ÜB	07. Jun	VL	ÜB3: Besprechung Aufg. 3;	VL6: Big Data - Prozesse
8.	12. Jun	frei	14. Jun	frei	Pfingsten	Pfingsten
9.	19. Jun	VL	21. Jun	ÜB	VL7: Big Data - Modelling 1	ÜB4: Besprechung Aufg. 4;
10.	26. Jun	ÜB	28. Jun	VL	ÜB5: Besprechung Aufg. 5;	VL8: Big Data - Modellin 2
11.	03. Jul		05. Jul	VL		VL9: Big Data - Evaluation
12.	10. Jul	ÜB	12. Jul	VL	ÜB6: Besprechung Aufg. 6;	VL10: Big Data - Infrastruktur
13.	17. Jul		19. Jul	VL		VL11: Anwendungen Institut
14.	24. Jul	ÜB	26. Jul	VL	ÜB8: Besprechung Aufg. 7;	VL12: Cyber Security

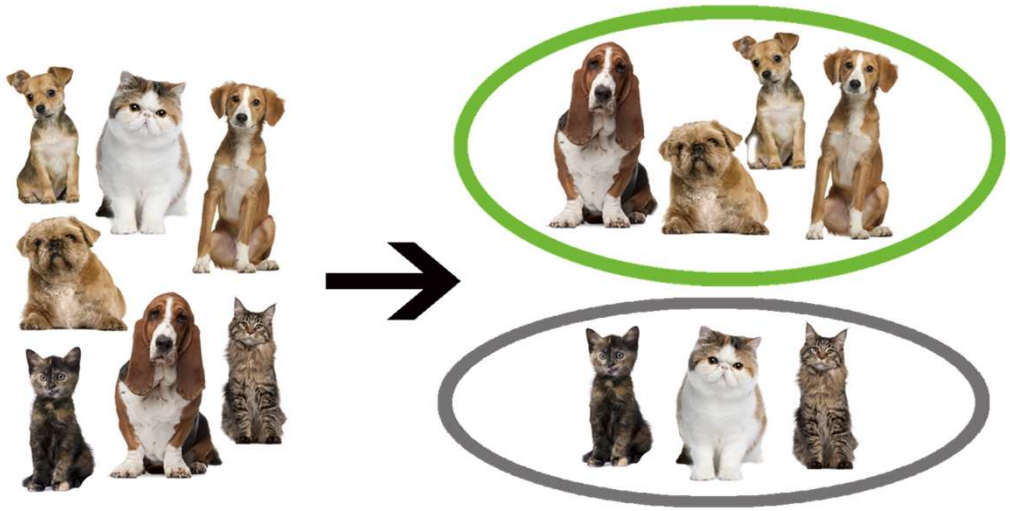
Machinelles Lernen



Algorithm ↓	Abk.	Feature Reduction	Clustering	One-Class Classification	Two-Class Classification	Multi-Class Classification	Non-Linear Regression	Linear Regression	Periodic Regression
Faktorenanalyse		X							
Principal Component Analysis	PCA	X		X					
	K-means		X						
hierarchical cluster analysis	HCA		X						
DBSCAN			X						
One Class Support Vector Machine	OCSVM			X					
Isolation Forest				X					
	LODA			X					
(künstliche) Neuronale Netze	NN			X (Autoencoder)	X	X	X	X	X
Support Vector Machine	SVM				X				
Decision Tree					X	X			
Bayes-Klassifikation					X	X			
Random Forest						X			
Diskriminanzanalyse				x	x	x			
Logistic Regression							X	X	
Linear Regression								X	
Harmonic Regression									X
Nächste-Nachbar-Klassifikation	K-NN	x	x						

2 Ansätze des Unüberwachten Lernen

- Clustering
  - Der Machine Learning Algorithmus wird die Bilder in die Kategorien Katzen und Hunde aufteilen, obwohl er darauf nicht trainiert wurde und ohne die Kategorien zu kennen.

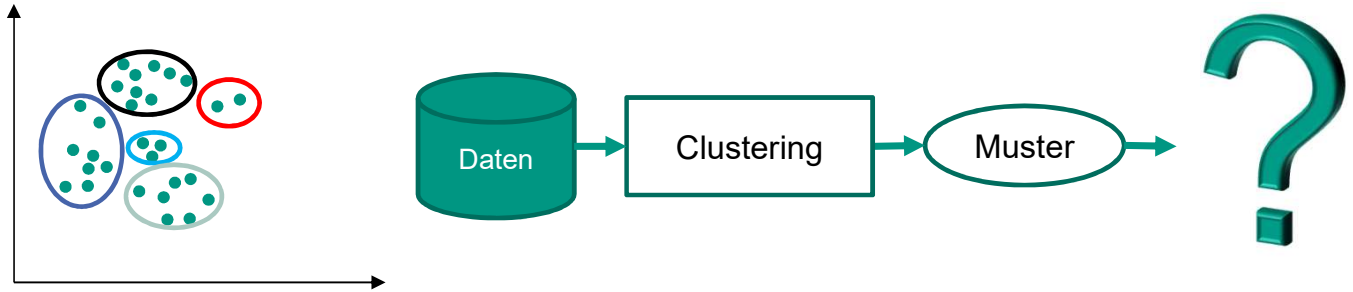


- Reduzierung der Dimensionalität (dimensionality oder feature reduction)
  - Das System erkennt durch ein Muster, welche (im Vorfeld definierten) Dimensionen zusammen gehören und reduziert diese.
  - Das steigert die Performance bei der Verarbeitung und Analyse.

Modellierung der Daten – Data Mining

Clustering (unüberwacht)

- Gruppierung von gleichartigen Situationen

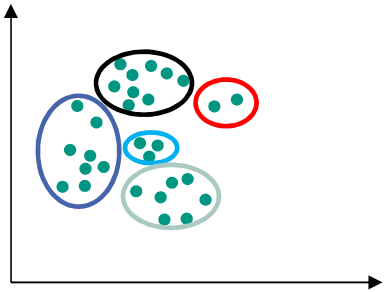


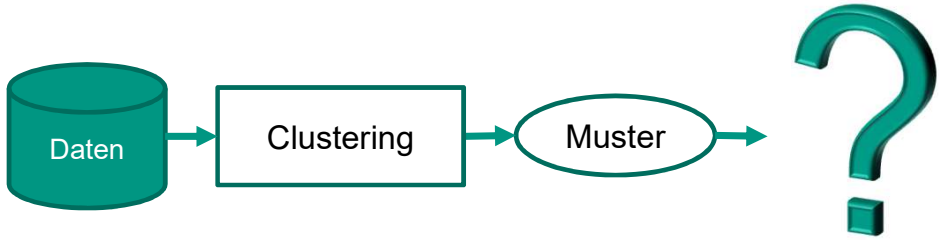
- Vorteil:
  - Benötigt keine vorherigen Annotationen
  - Entdeckung von bisher unbekannten Differenzierbarkeiten
- Nachteil
  - Nicht zielgerichtet
  - Stark von Datenselektion, Transformation und Clustering-Verfahren abhängig
  - Bewertung der Ergebnisse schwierig

Modellierung der Daten – Data Mining

Clustering (unüberwacht)

- Gruppierung von gleichartigen Situationen

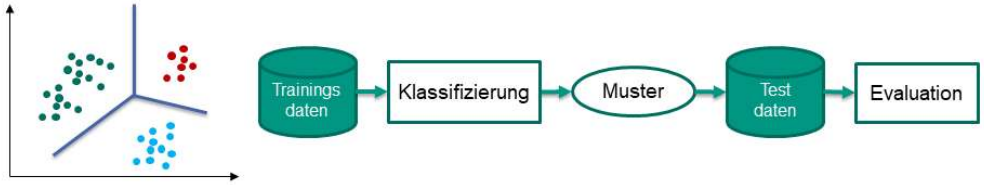




- Vorteil:
  - Benötigt keine vorherigen Anr
  - Entdeckung von bisher unbek
- Nachteil
  - Nicht zielgerichtet
  - Stark von Datenselektion, Tra
  - Bewertung der Ergebnisse sc

Support Vector Machine (überwacht)

- Gruppierung/ Separierung von vorsortierten Daten



- Vorteil:
  - Zielgerichtet durch Vorsortierung
  - Evaluation durch Testdaten möglich
    - Vergleich der entdeckten Muster (Optimierung)
    - Vergleich erleichtert Merkmalselektion
- Nachteil:
  - Nur anwendbar auf vorsortierte Daten
  - Neues Wissen beschränkt auf die vorgegeben Klassen

Clusteranalyse

Unüberwachtes Lernen, ohne Belohnung, ohne Vorwissen

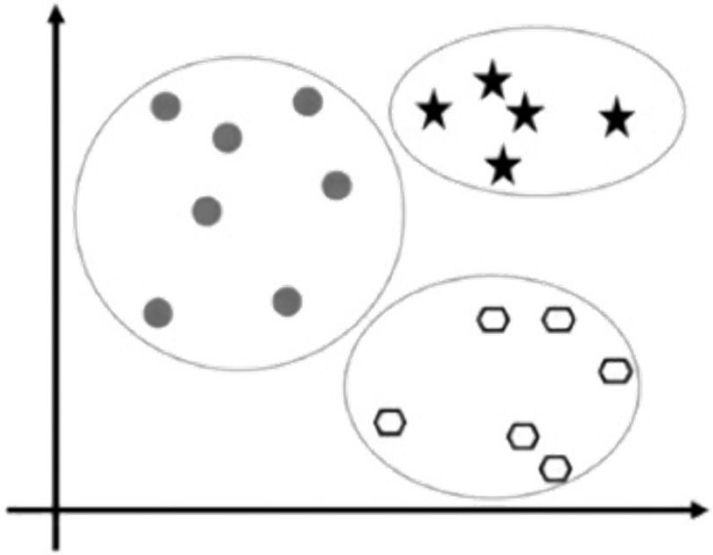
- Clustering nimmt eine Gruppierung anhand von Ähnlichkeiten vor
  - *Dabei ist jedoch nicht gewährleistet, dass diese Ähnlichkeiten aussagekräftig oder für irgendeinen Zweck nützlich sind.*
- Ziel ist die Objekte so zu Gruppen (*Clustern*) zusammenzufassen, dass die Objekte in einer Gruppe ähnlich und die Gruppen untereinander unähnlich sind.
- Da im Gegensatz zu klassifizierenden Verfahren die Gruppen vor der Anwendung der Analyse nicht bekannt sind, handelt es sich um ein Verfahren des unüberwachten Lernens.
- Clusteranalysen werden häufig eingesetzt als:
  - Grundlage einer Markt- bzw. Kundensegmentierung,
  - Grundlage zur anschließenden, automatischen Klassifizierung von Daten,
  - Im Bereich der Bilderkennung.

Clusteranalyse

Unüberwachtes Lernen, ohne Belohnung, ohne Vorwissen

- Clustering nimmt eine Gruppierung anhand von Ähnlichkeiten vor
  - *Dabei ist jedoch nicht gewährleistet, dass diese Ähnlichkeiten aussagekräftig oder für irgendeinen Zweck nützlich sind.*
- Ziel ist die Objekte so zu Gruppen (*Clustern*) zusammenzufassen, dass die Objekte in einer Gruppe ähnlich und die Gruppen untereinander unähnlich sind.
- Da im Gegensatz zu klassifizierenden Verfahren die Gruppen vor der Anwendung der Analyse nicht bekannt sind, handelt es sich um ein Verfahren des unüberwachten Lernens.
- Clusteranalysen werden häufig eingesetzt als:
  - Grundlage einer Markt- bzw. Kundensegmentierung,
  - Grundlage zur anschließenden, automatischen Klassifizierung von Daten,
  - Im Bereich der Bilderkennung.

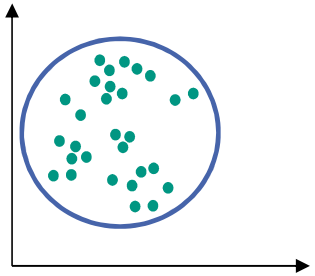
- Punktewolken
  - Die Punkte stellen die einzelnen Datensätzen in einem n-dimensionalen Raum dar, wobei die n-Dimensionen den Variablen entsprechen
  - Ein Cluster stellt dann eine Punktewolke dar, die ähnliche Punkte zusammenfasst.



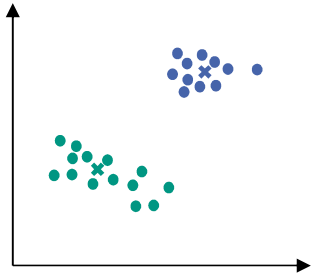
Clusteranalyse
Kategorisierung

- Unter dem Begriff der Clusteranalyse wird ein Vielzahl von verschiedenen heuristischen Verfahren zusammengefasst.
- Die einzelnen Clusterverfahren unterscheiden sich anhand der verwendeten Ähnlichkeitsmaße sowie der Vorgehensweise, mit der eine möglichst gute, eindeutige Trennung der Cluster erzielt werden soll.
  - ► Es gibt keinen Algorithmus, der eine "optimale" Gruppierung garantiert
- Die Clusterverfahren werden wie folgt unterteilt:

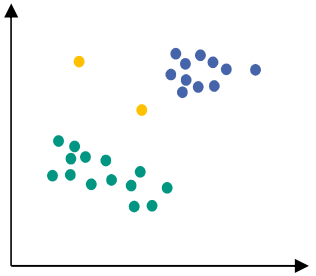
Hierarchisch



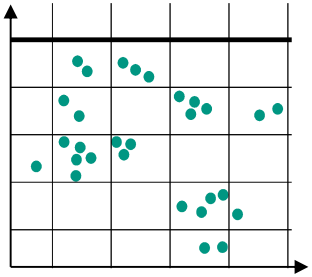
Partitionierend



Dichte-basiert



Gitter-basiert



Hierarchische Clusterverfahren

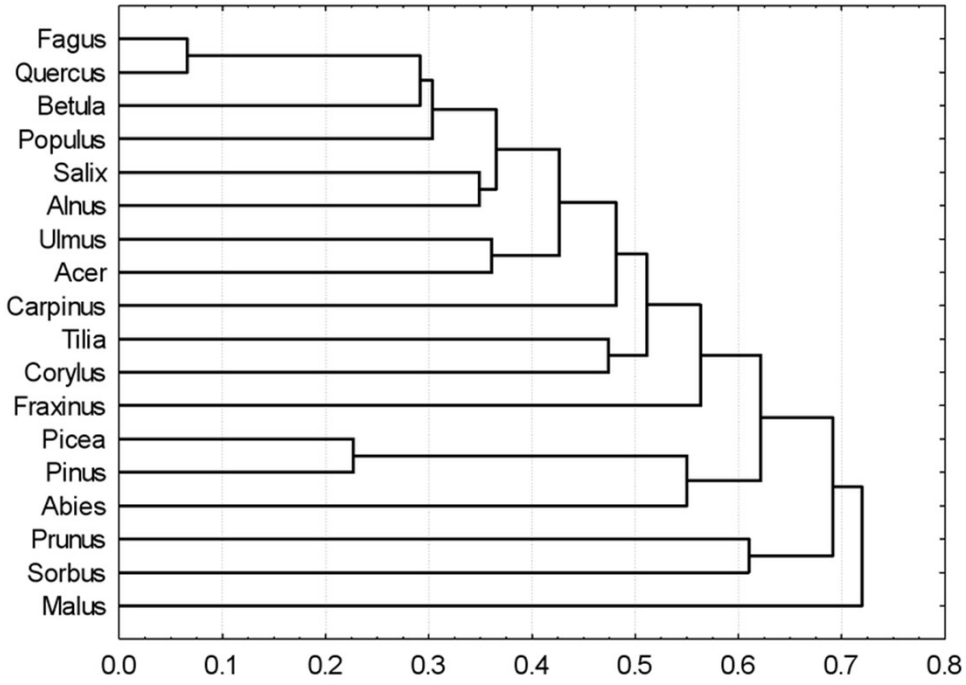
- Hierarchische Verfahren erzeugen eine Clusterstruktur möglicher Einteilungen, so dass die Objekte auf oberster Ebene in nur wenige Cluster unterteilt werden, die sich wiederum in eigene Cluster unterteilen etc.
  - Bei den agglomerierenden (anhäufenden) Verfahren wird initial jedes Objekt der Datensammlung als eigenes Cluster interpretiert.
    - Es werden sukzessive einzelne, zueinander ähnliche Objekte zu Clustern und bereits erkannte Cluster zu größeren Clustern verschmolzen bis der Abstand zwischen je zwei Clustern einen bestimmten Schwellwert übersteigt oder wenn nur noch ein großes Cluster übrig ist.
  - Divisive (unterteilende) Verfahren beginnen anfänglich mit einem einzigen Cluster.
    - Jedes Objekt der Datensammlung ist diesem Cluster zugeordnet. Anschließend werden immer neue (Teil-) Cluster erstellt, d.h. das ursprüngliche Cluster immer feiner unterteilt.
    - Dieser Prozess dauert so lange an, bis nur noch Cluster bestehend aus Einzelobjekten übrig bleiben.
- Hierarchische Clusterverfahren sind relativ rechenaufwändig.
  - Sie besitzen eine Laufzeit von etwa $O(n^2)$.
  - Durch Verwendung zusätzlicher Heuristiken kann jedoch eine annähernd lineare Laufzeit erreicht werden.

Quelle: GRIN - Clustering und Evaluierung von Benutzerprofilen bei Web-Portalen https://www.grin.com/document/56106

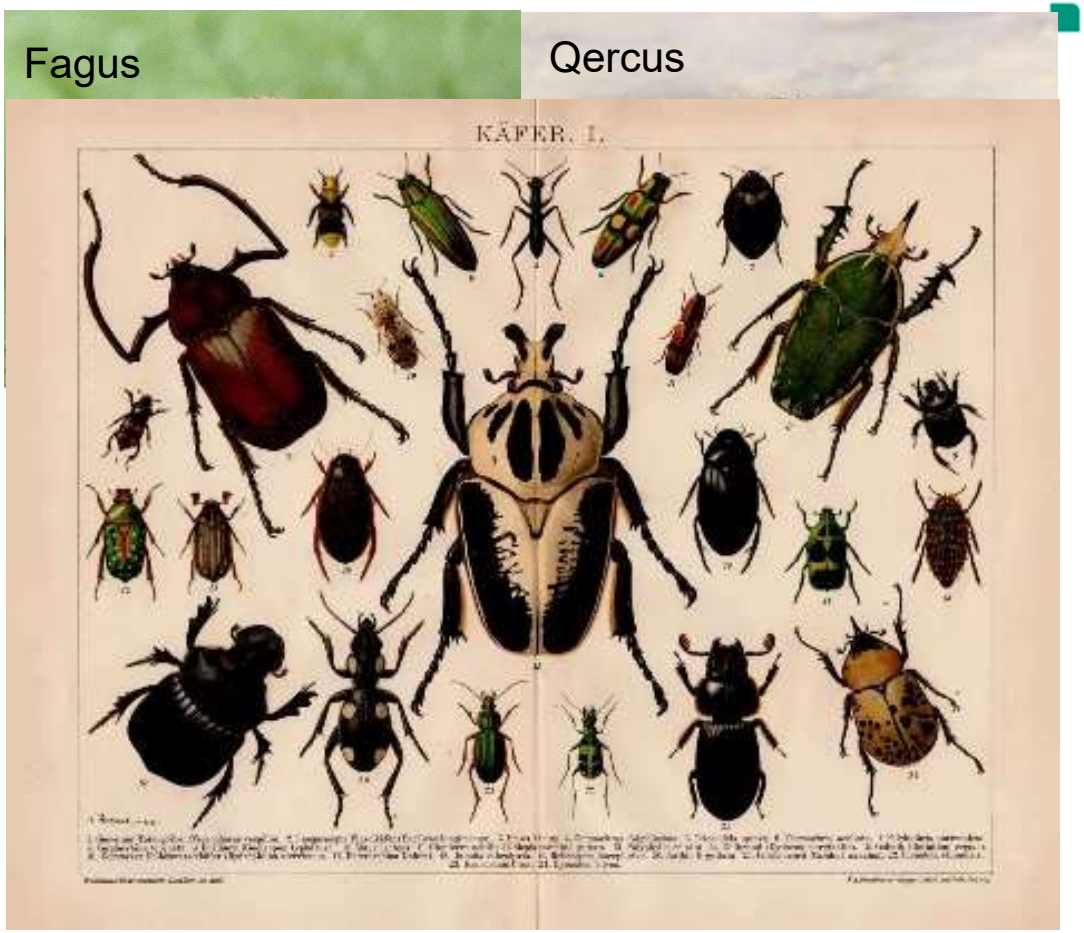
Dendrogramm

- Beiden hierarchischen Verfahrenstypen, agglomerierend wie divisiv, ist gemeinsam, dass eine Baumstruktur entsteht, die meist als Dendrogramm bezeichnet wird.
- Die Länge eines Querbalkens zeigt an, auf welcher Hierarchieebene Cluster verschmolzen werden.
- Wenn ein einzelner Punkt erst weit oben im Dendrogramm mit den anderen Gruppierungen vereint wird, weist dies darauf hin, dass er sich deutlich von den anderen Punkten unterscheidet.
  - Wir bezeichnen das als einen »Ausreißer«

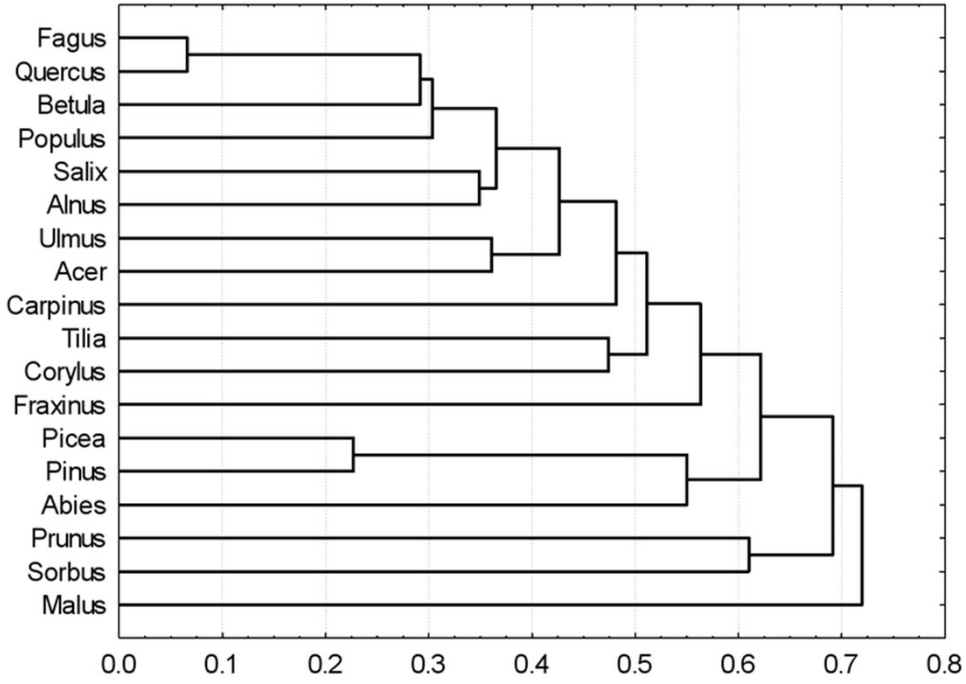
- Beispiel: *Dendrogramm der in einer Clusteranalyse errechneten Ähnlichkeitswerte (sörensen-koeffizient) von 300 xylobionten Käferarten mit Vorkommen an Rotbuche aus dem Steigerwald und dem Spessart*



Dendrogramm

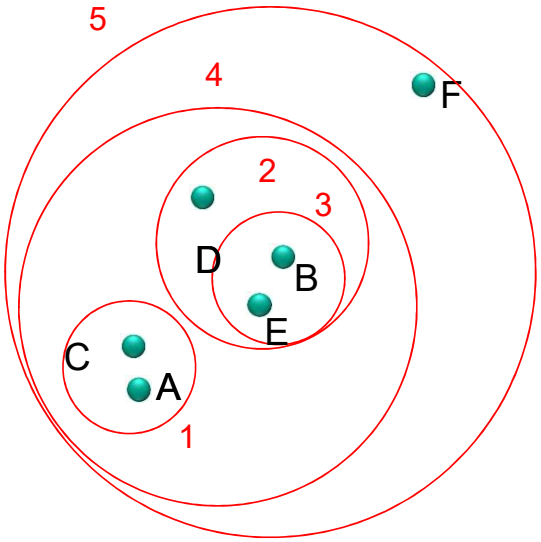


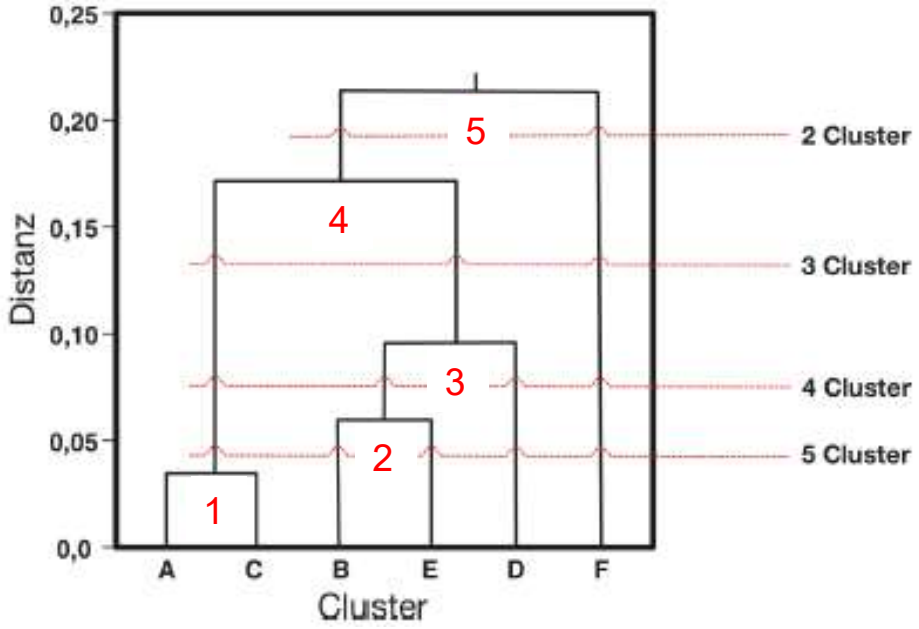
- Beispiel: *Dendrogramm der in einer Clusteranalyse errechneten Ähnlichkeitswerte (sörensen-koeffizient) von 300 xylobionten Käferarten mit Vorkommen an Rotbuche aus dem Steigerwald und dem Spessart*



Dendrogramm

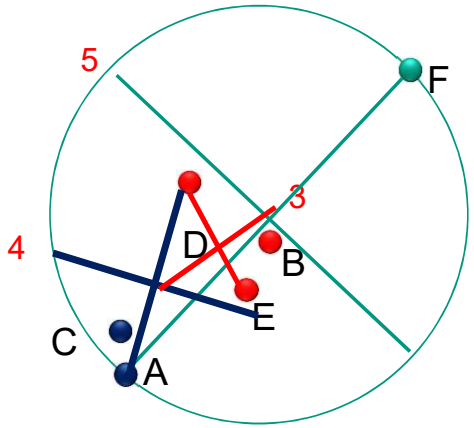
Beispiel (agglomerierend)

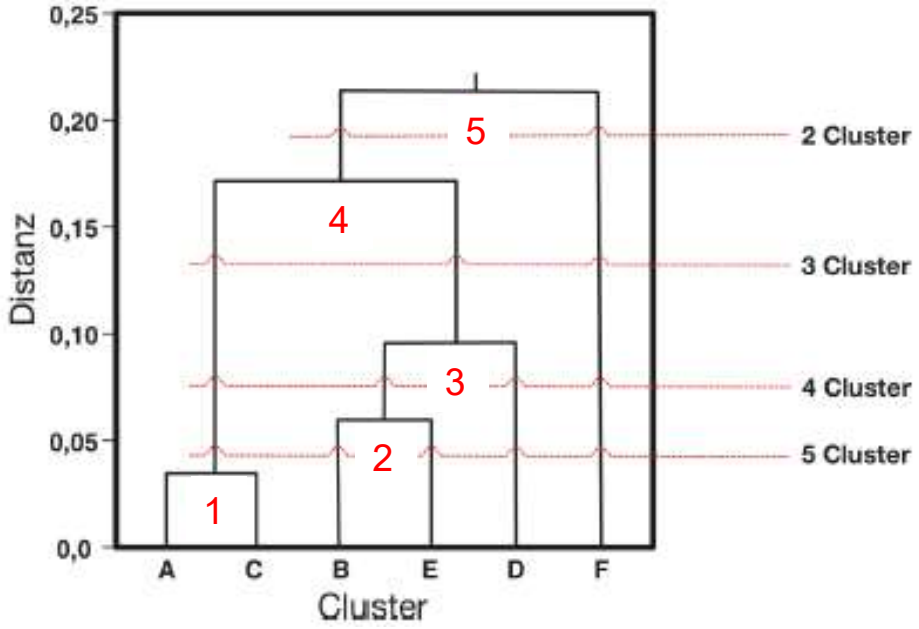




- Sechs Punkte und ihre möglichen Cluster.
- Links sind die sechs Punkte A bis F und die Kreise 1 bis 5 dargestellt, die verschiedene auf der Distanz beruhende Gruppierungen kennzeichnen.
- Sie bilden eine implizite Hierarchie → Das Dendrogramm der Gruppierung im unteren Teil der Abbildung zeigt diese Hierarchie explizit.

Dendrogramm
Beispiel (divisiv)





- Kaufman und Rousseeuw (1990) beschreiben eine *Divisive Clustering Procedure* wie folgt:
  - Starte mit einem kreisförmigen Cluster, das alle Punkte enthält.
    - Durchmesser ist der größte Abstand 2er Punkte (Maximaldistanz)
  - Dieser Cluster wird in zwei Cluster geteilt, wobei die Senkrechte auf die Mitte des Abstandes den Teiler bildet.
  - Die Schritte werden je Teil-Cluster solange wiederholt, bis alle Cluster nur noch ein Objekt enthalten.

Partitionierende Clusterverfahren

- Die Aufgabe partitionierender Clusterverfahren besteht darin, eine Datensammlung, ausgehend von einer initialen Partitionierung, in k disjunkte Mengen derart zu partitionieren, dass sich die Objekte innerhalb einer Gruppe so ähnlich wie möglich sind.
  - Jedes Objekt wird einem eindeutigen Cluster zugewiesen.
  - Es entsteht keine hierarchische Clusterstruktur.
- Der Vorteil partitionierender Clusterverfahren liegt in der Untersuchung sehr großer Datensammlungen, wo die Erstellung eines Dendrogramms nur schwer durchzuführen ist.
- *Bei partitionierenden Clusterverfahren ist es notwendig, aber auch problematisch, vor dem Start des Algorithmus anzugeben, auf wie viele (unbekannte) Partitionen k der Algorithmus die Datensammlung untersuchen soll.*
- Damit bleibt die Anzahl der Cluster konstant.
  - Sicherlich lässt sich der Algorithmus mehrere Male mit verschiedenen Werten für k starten, jedoch muss man in der Lage sein, sich zwischen verschiedenen k-Werten zu entscheiden.
  - Welches k zur optimalen Clustereinteilung führt, kann nur anhand einer Ähnlichkeitsfunktion (*score function*) bestimmt werden.

Partitionierende Clusterverfahren

- Anwendung
  - Partitionierende Clusterverfahren sind geeignet, um für eine vorgegebene Anzahl von Clustern die beste Aufteilung der Objekte zu finden.
  - Deshalb ist es wichtig, eine gute Anfangsgruppierung vorzugeben.
  - Hierfür wird häufig die Lösung eines hierarchischen Clusterverfahrens verwendet, die dann mittels eines partitionierenden Verfahren verbessert wird.
  - Sinnvoll ist auch, die anfängliche Gruppierung zu variieren und die erzielten Ergebnisse zu vergleichen.
  - Auch kann die Anzahl der Cluster verändert werden
- Das wohl wichtigste partitionierende Verfahren ist der k-means Algorithmus von MacQueen.
  - Dieser wird häufig zur Verbesserung der Lösung eines hierarchischen Clusterverfahrens eingesetzt.

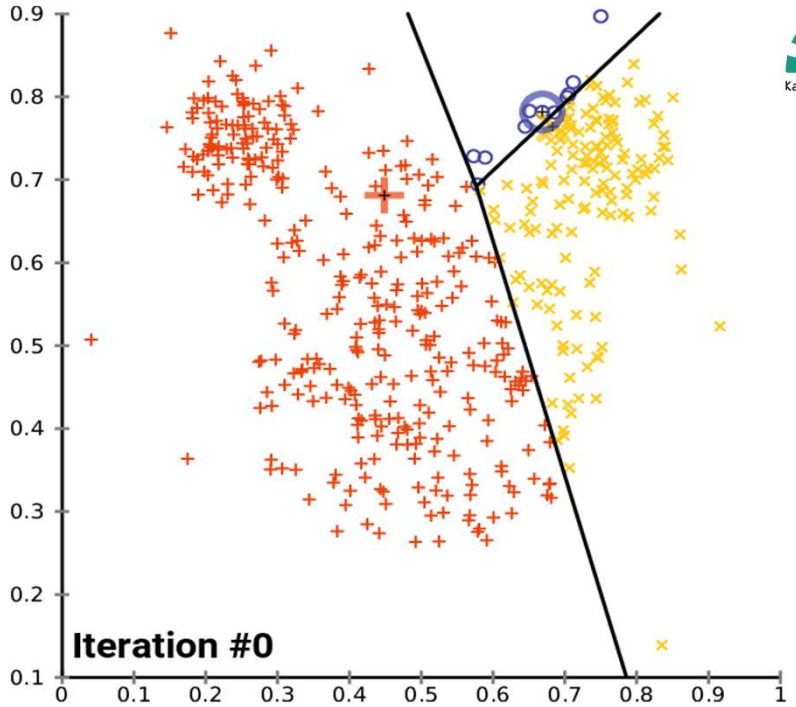
k-Means

Partitionierend

- Bildung von Mengen ähnlicher Objekte in k Gruppen
  - mit möglichst geringer Varianz
  - In Gruppen ähnlicher Größen
- Die Clusterzentren werden zufällig festgelegt und die Summe der quadrierten Abstände der Objekte zu ihrem nächsten Clusterzentrum wird minimiert.
- Das Update der Clusterzentren geschieht durch Mittelwertbildung aller Objekte in einem Cluster.

$$J = \sum_{i=1}^{k} \sum_{x_j \epsilon S_i} \left\| x_j - \mu_i \right\|^2$$

x_j: Datenpunkt
S_i: Cluster
μ_i: Schwerpunkt



Lloyd Algorithmus:

1. k unterschiedliche Zentren $c_1, c_2, \ldots, c_k$
2. Solange sich die Zielfunktion verbessert:
   Partitioniere P in Cluster $S_1, S_2, \ldots, S_k$ dass S_i die Punkte aus P enthält, deren nächstgelegenes Zentrum c_i ist
   Für jedes $1 \leq i \leq k$ sei $c_i \leftarrow \mu(C_i)$

k-Means

Varianten (für Interessierte)

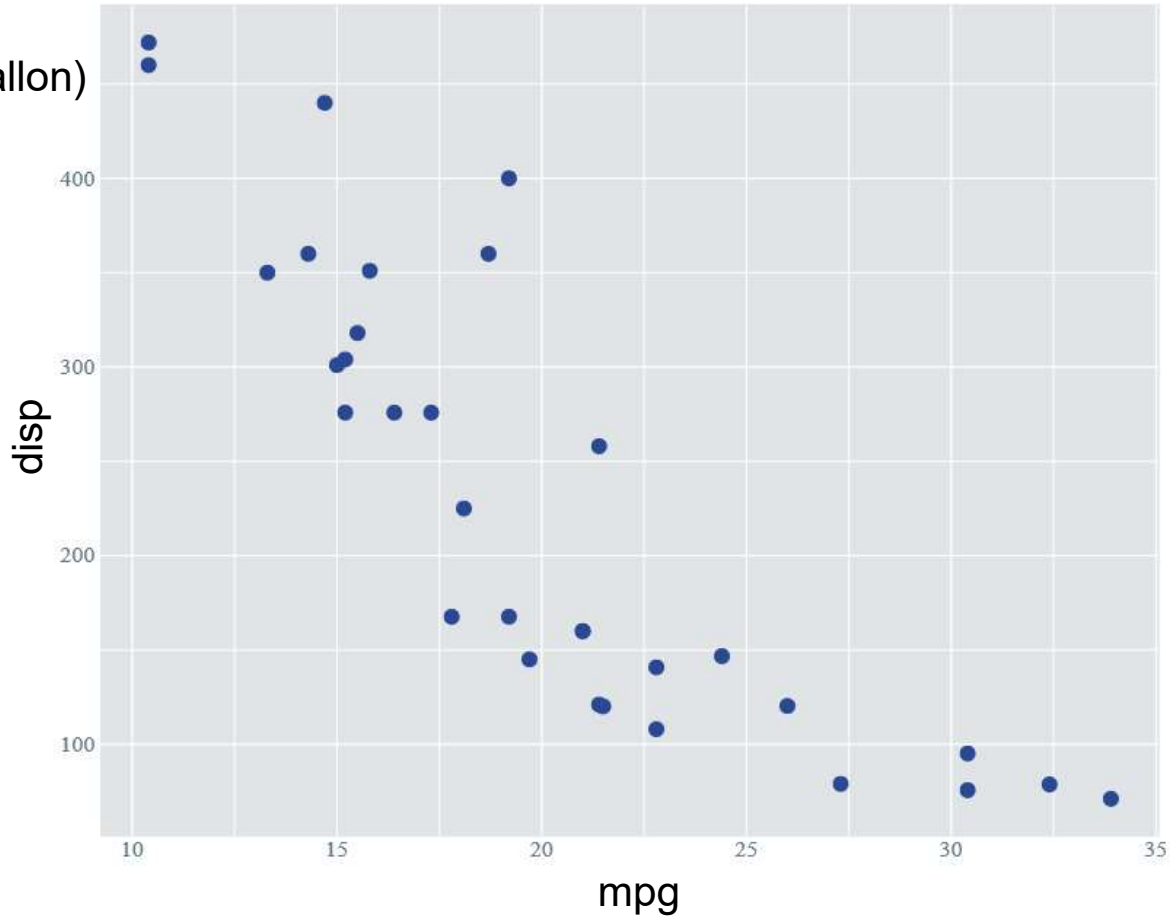
- k-Means++
  - Als Clusterzentren werden auch zufällig Objekte so ausgewählt, so dass sie etwa uniform im Raum der Objekte verteilt sind. Dies führt zu einem schnelleren Algorithmus.
- k-Median-Algorithmus
  - Hier wird die Summe der Manhattan-Distanzen der Objekte zu ihrem nächsten Clusterzentrum minimiert. Das Update der Clusterzentren geschieht durch die Berechnung des Medians aller Objekte in einem Cluster. Ausreißer in den Daten haben dadurch weniger Einfluss.
- k-Medoids oder Partitioning Around Medoids (PAM)
  - Die Clusterzentren sind hier immer Objekte. Durch Verschiebung von Clusterzentren auf ein benachbartes Objekt wird die Summe der Distanzen zum nächstgelegenen Clusterzentrum minimiert. Im Gegensatz zum k-Means Verfahren werden nur die Distanzen zwischen den Objekten benötigt und nicht die Koordinaten der Objekte.
- EM-Clustering
  - Die Cluster werden als multivariate Normalverteilungen modelliert.
  - Mit Hilfe des EM-Algorithmus werden die unbekannten Parameter der Normalverteilungen iterativ geschätzt.
  - Im Gegensatz zu k-means wird damit eine weiche Clusterzuordnung erreicht:
  - Mit einer gewissen Wahrscheinlichkeit gehört jedes Objekt zu jedem Cluster und jedes Objekt beeinflusst so die Parameter jeden Clusters.
- Fuzzy C-Means
  - Für jedes Objekt wird ein Zugehörigkeitsgrad zu einem Cluster berechnet, oft aus dem reellwertigen Intervall [0,1] (Zugehörigkeitsgrad=1: Objekt gehört vollständig zu einem Cluster, Zugehörigkeitsgrad=0: Objekt gehört nicht zu dem Cluster).
  - Dabei gilt: je weiter ein Objekt vom Clusterzentrum entfernt ist, desto kleiner ist auch sein Zugehörigkeitsgrad zu diesem Cluster.
  - Wie im k-Median-Verfahren werden die Clusterzentren dann verschoben, jedoch haben weit entfernte Objekte (kleiner Zugehörigkeitsgrad) einen geringen Einfluss auf die Verschiebung als nahe Objekte.
  - Damit wird auch eine weiche Clusterzuordnung erreicht: Jedes Objekt gehört zu jedem Cluster mit einem entsprechenden Zugehörigkeitsgrad.

Beispiel: 32 Automobile aus dem Datensatz *mtcars*

Beispiel: s. https://www.inwt-statistics.de/blog-artikel-lesen/Clusteranalyse_in_R.html

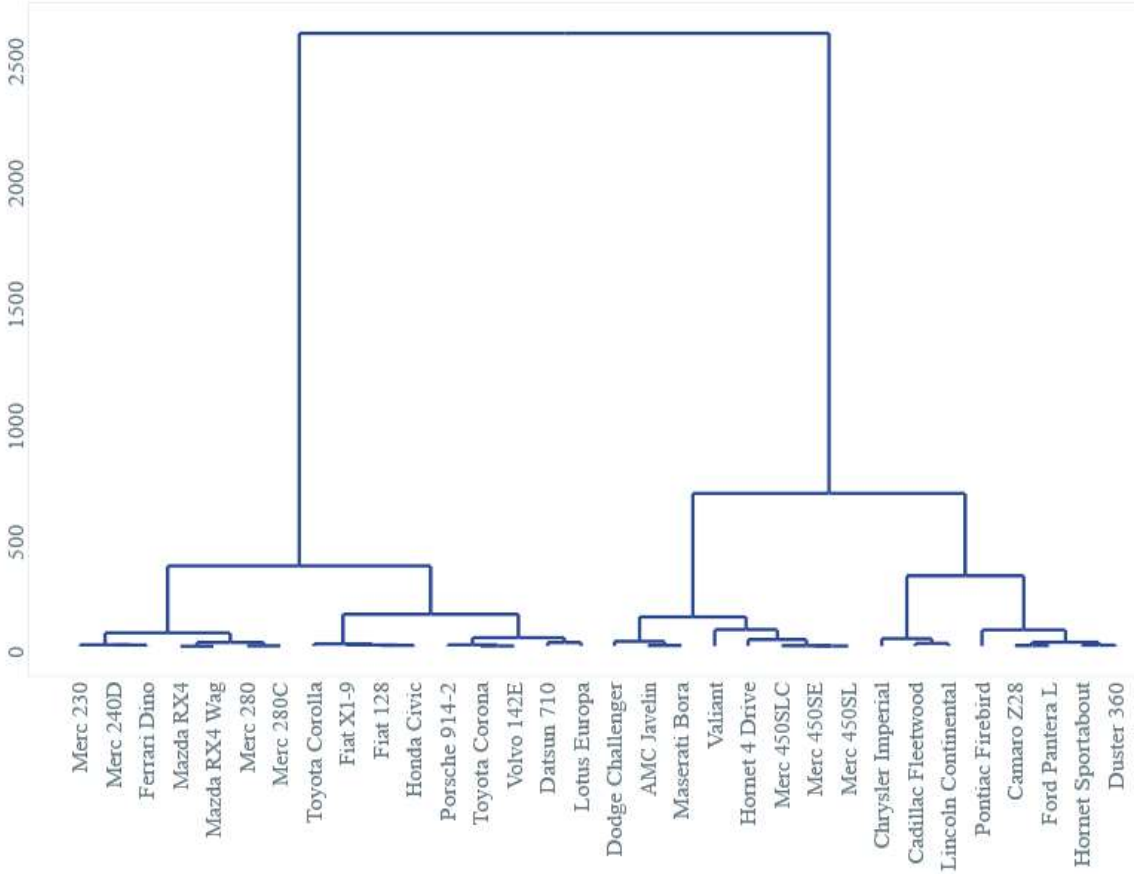
- Für die Analyse verwendete Variablen
  - mpg (Kraftstoffverbrauch in miles per gallon)
  - disp (Hubraum in Kubikinch)





Dendrogramm zu *mtcars*

- Für die Analyse verwendete Variablen
  - mpg (Kraftstoffverbrauch in *miles per gallon*)
  - disp (Hubraum in *Kubikinch*)
- Nun führen wir ein hierarchisches Verfahren durch
  - Auf der y-Achse des Dendrogramms ist die Distanz (euklidischer Abstand) abgetragen.
- Augenscheinlich sind 2 Cluster in diesem Falle sinnvoll.

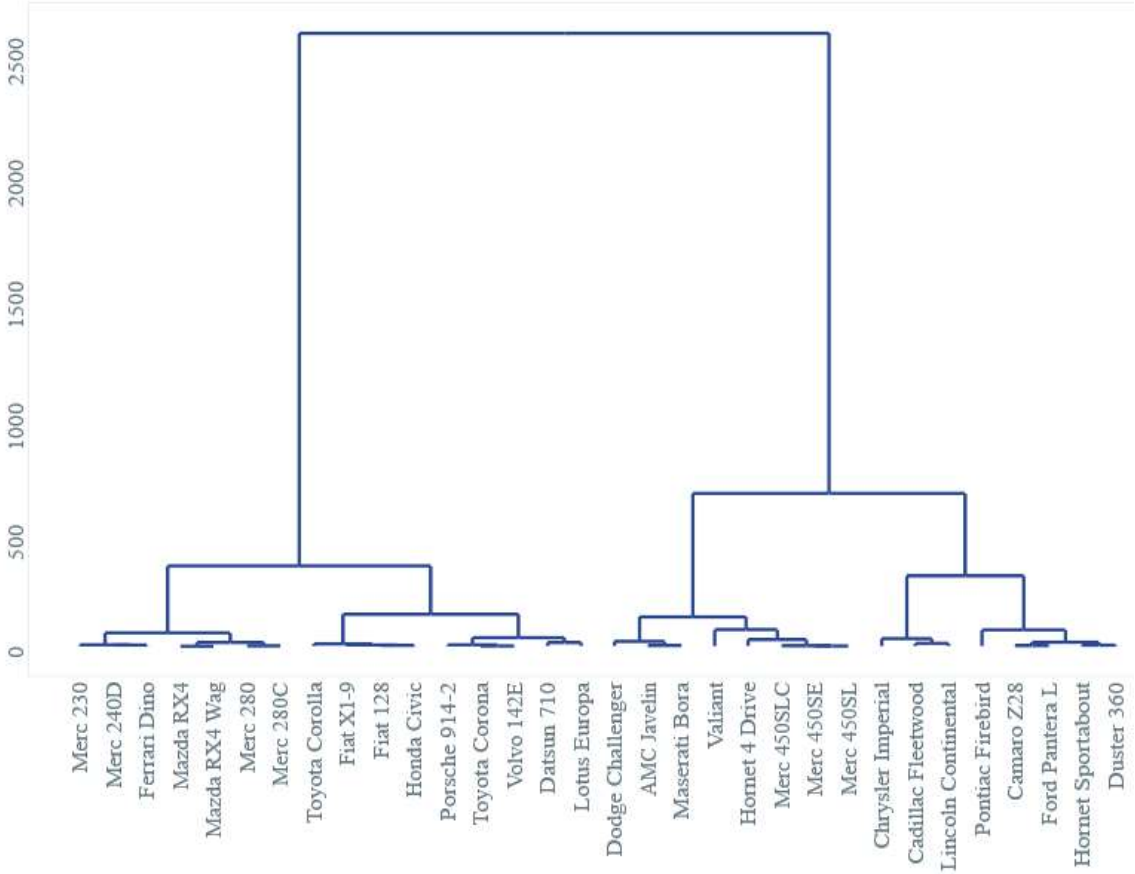


Dendrogramm zu *mtcars*

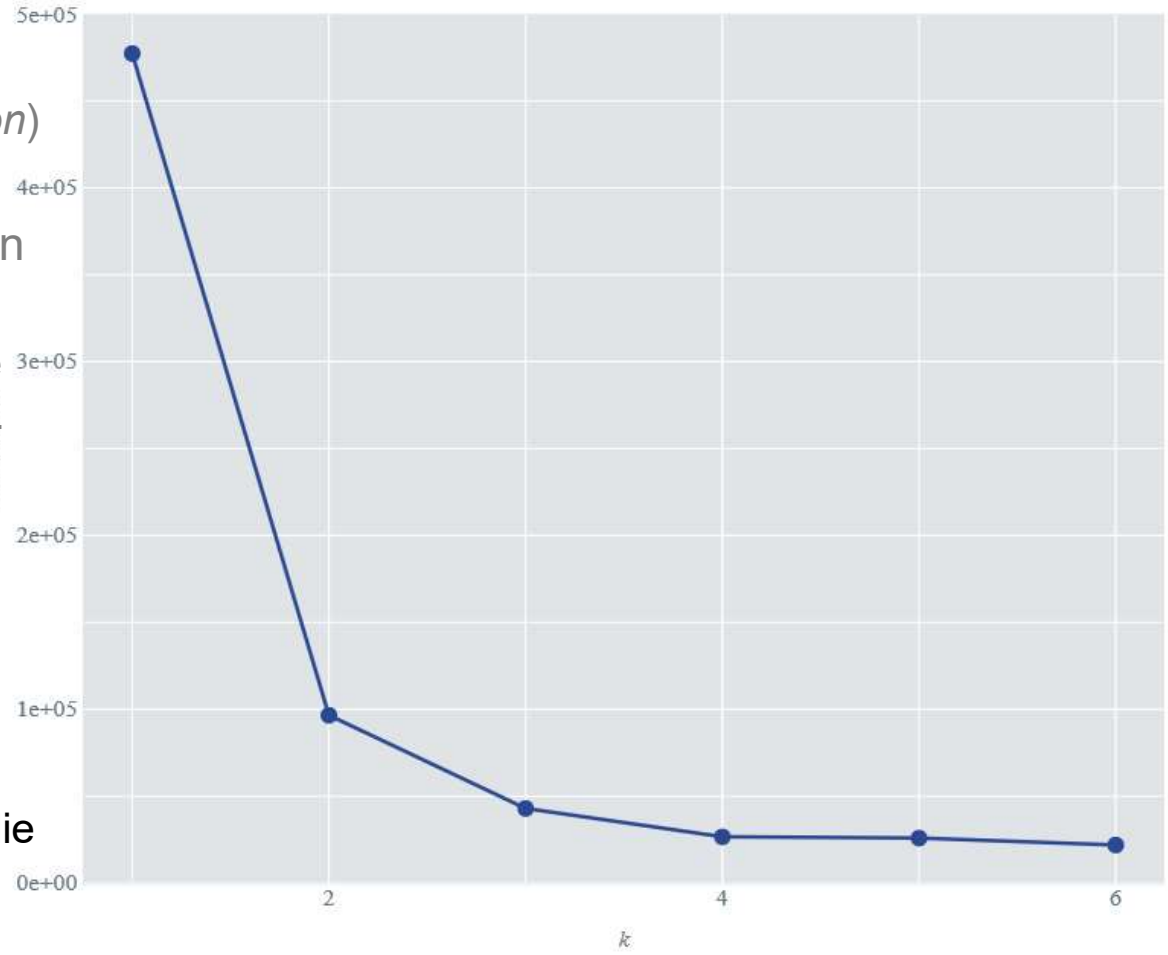
Ferrari

Ferrari Dino 246 GT

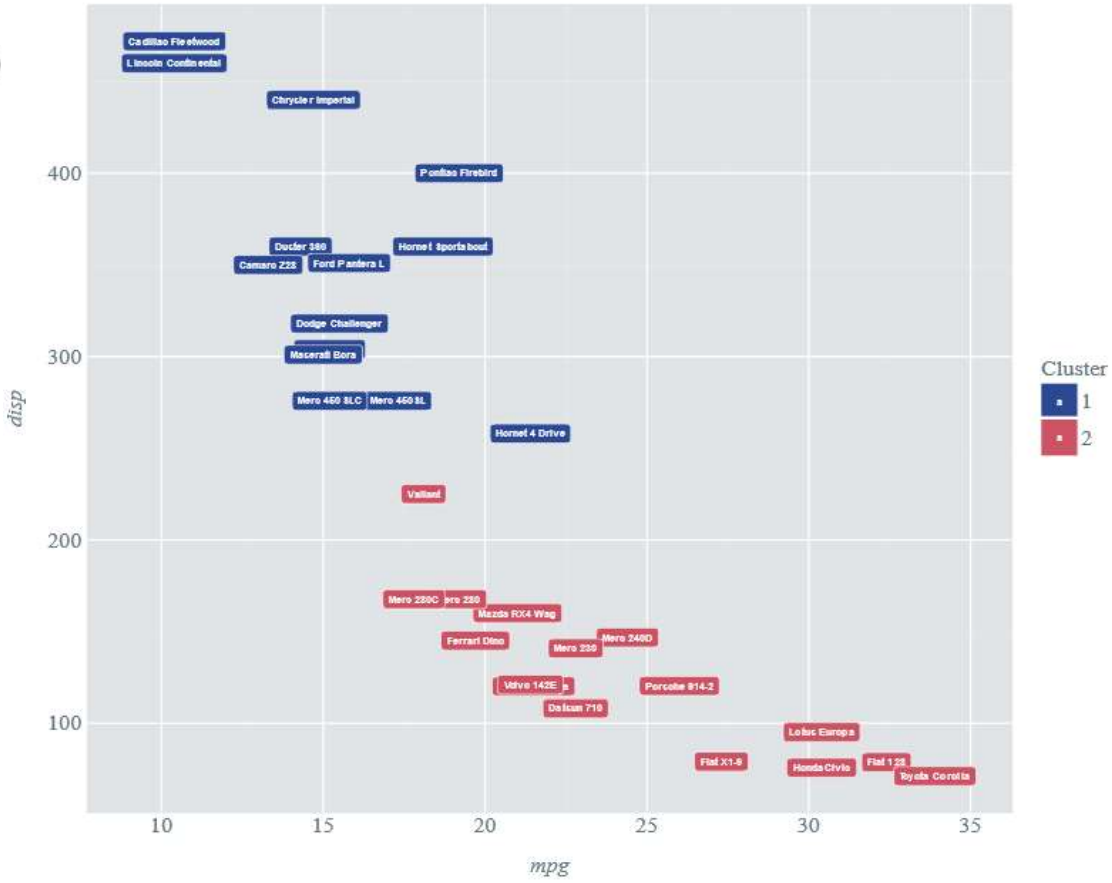
Dino 206 GT / 246 GT	
Produktionszeitraum:	1969–1974
Klasse:	Sportwagen
Karosserieversionen:	Coupé, Cabriolet
Motoren:	Ottomotoren: 2,0–2,4 Liter (132–143 kW)
Länge:	4200–4210 mm
Breite:	1700 mm
Höhe:	1115 mm
Radstand:	2280–2340 mm
Leergewicht:	1000–1100 kg



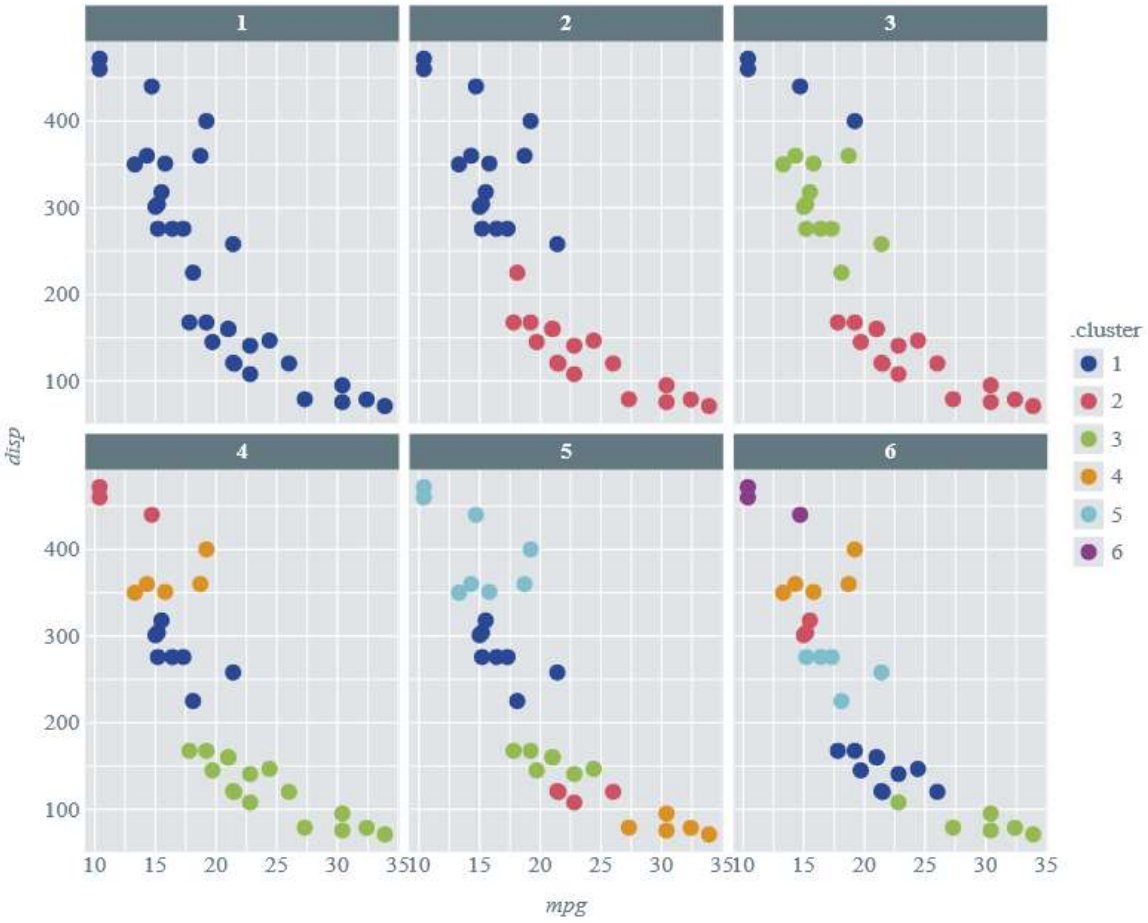
- Für die Analyse verwendete Variablen
  - mpg (Kraftstoffverbrauch in *miles per gallon*)
  - disp (Hubraum in *Kubikinch*)
- Nun führen wir ein hierarchisches Verfahren durch
  - Auf der y-Achse des Dendrogramms ist die Distanz (euklidischer Abstand) abgetragen.
- Augenscheinlich sind 2 Cluster in diesem Falle sinnvoll.
- Als nächstes erstellen wir einen Screeplot, wobei für jedes k ein k-means Verfahren durchgeführt wird.
  - Der Screeplot zeigt auf der x-Achse die Anzahl der Cluster k und auf der y-Achse die Varianz innerhalb der Cluster.



- Für die Analyse verwendete Variablen
  - mpg (Kraftstoffverbrauch in *miles per gallon*)
  - disp (Hubraum in *Kubikinch*)
- Nun führen wir ein hierarchisches Verfahren durch
  - Auf der y-Achse des Dendrogramms ist die Distanz (euklidischer Abstand) abgetragen.
- Augenscheinlich sind 2 Cluster in diesem Falle sinnvoll.
- Nun sehen wir uns die durch den k-means-Algorithmus erhaltene Gruppenstruktur für k=2 an



- Für die Analyse verwendete Variablen
  - mpg (Kraftstoffverbrauch in *miles per gallor*
  - disp (Hubraum in *Kubikinch*)
- Nun führen wir ein hierarchisches Verfahren durch
  - Auf der y-Achse des Dendrogramms ist die Distanz (euklidischer Abstand) abgetragen.
- Augenscheinlich sind 2 Cluster in diesem Falle sinnvoll.
- Nun sehen wir uns die durch den k-means-Algorithmus erhaltene Gruppenstruktur für k=2 an
- An diesem Punkt wäre es interessant, vergleichsweise die Gruppenstruktur auch für k=1, k=3 etc. zu sehen.



Interpretation der Clusterlösung

- Nachdem wir die Clusterlösung validiert haben, kommen wir nun zum eigentlich schwierigsten Teil der Analyse: der Interpretation.
  - Für die Analyse haben wir den Kraftstoffverbrauch mpg und den Hubraum disp herangezogen.
- Zur Hilfestellung nehmen wir die restlichen Informationen aus dem Datensatz *mtcars*
  - Dazu übergeben wir die Spalte mit der Clusterzuordnung für jedes Auto an den ursprünglichen Datensatz zurück.
  - Nun können wir die restlichen Variablen getrennt nach Cluster untersuchen.

cluster	avg_cyl	avg.horsepower	avg_drat	avg_weight	avg_qsec	share_vmotor
(dbl)	(dbl)	(dbl)	(dbl)	(dbl)	(dbl)	(dbl)
1	7.9	202.6	3.2	3.9	16.9	0.0
2	4.7	97.4	3.9	2.6	18.6	0.4

cluster	share_manual_transmission	avg_gears	avg_carb
(dbl)	(dbl)	(dbl)	(dbl)
1	0.1	3.3	3.3
2	0.3	4.1	2.4

Interpretation der Clusterlösung

- Erkenntnisse:
  - Cluster 1 hat generell eine höhere Leistung und einen höheren Verbrauch als die aus Cluster 2.
  - Cluster 1-Autos haben durchschnittlich knapp 8 Zylinder, während im Cluster 2 sich typischerweise Autos mit 4 bis 6 Zylindern befinden.
- Mit den gesammelten Informationen interpretieren wir die zwei gefundenen Cluster als zwei Preisklassen.
  - Das Cluster 1 enthält die luxuriöseren Autos, mit mehr Leistung, mehr Zylindern und höherem Verbrauch, man könnte die Gruppe als 'Fahrzeuge der Oberklasse' betiteln.
  - Das Cluster 2 enthält demnach leistungsschwächere Autos, die günstiger im Preis sind und einen geringeren Verbrauch aufweisen.

[, 1] mpg Miles/(US) gallon
[, 2] cyl Number of cylinders
[, 3] disp Displacement (cu.in.)
[, 4] hp Gross horsepower
[, 5] drat Rear axle ratio
[, 6] wt Weight (1000 lbs)
[, 7] qsec 1/4 mile time
[, 8] vs Engine (0 = V-shaped, 1 = straight)
[, 9] am Transmission (0 = automatic, 1 = manual)
[,10] gear Number of forward gears
[,11] carb Number of carburetors

cluster	avg_cyl	avg.horsepower	avg_drat	avg_weight	avg_qsec	share_vmotor
(dbl)	(dbl)	(dbl)	(dbl)	(dbl)	(dbl)	(dbl)
1	7.9	202.6	3.2	3.9	16.9	0.0
2	4.7	97.4	3.9	2.6	18.6	0.4

cluster	share_manual_transmission	avg_gears	avg_carb
(dbl)	(dbl)	(dbl)	(dbl)
1	0.1	3.3	3.3
2	0.3	4.1	2.4

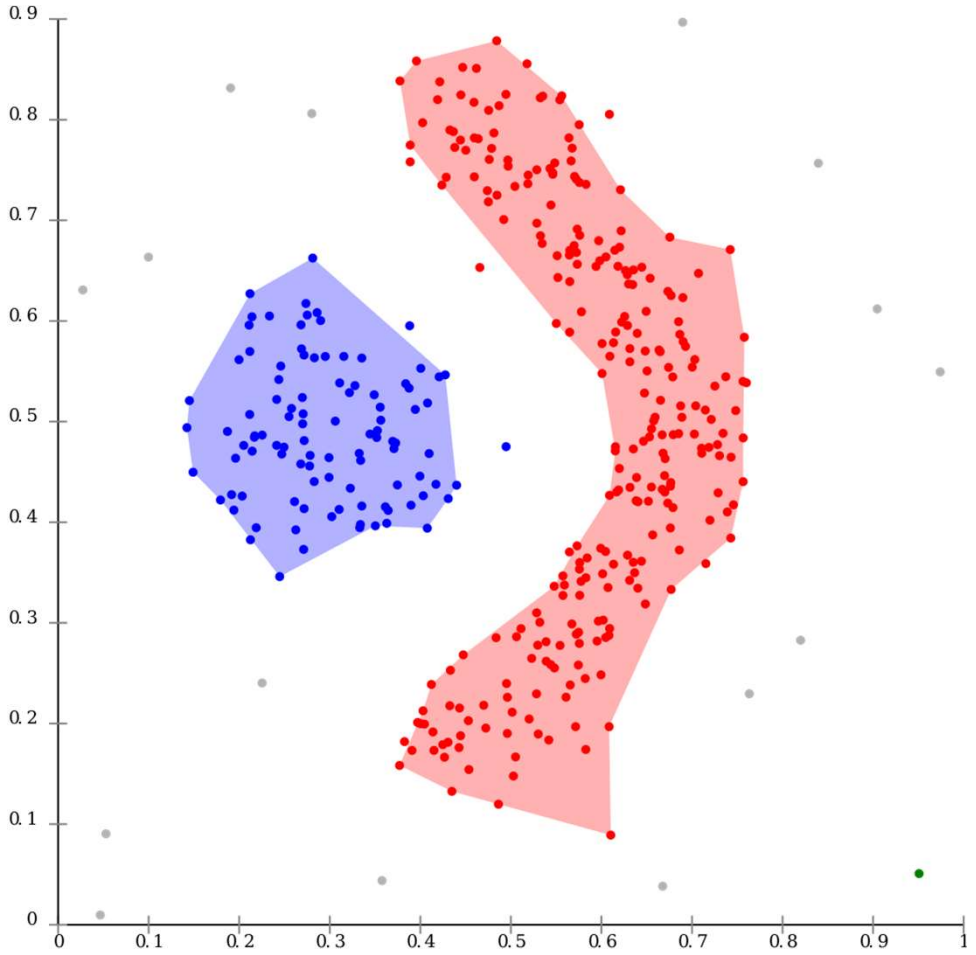
Interpretation der Clusterlösung

- Um diese Interpretation abzusichern, haben wir die unverbindliche Preisempfehlung (Achtung: die Modelle stammen aus den 1970ern) für die Autos herausgesucht. In der unteren Tabelle sind die Durchschnittswerte für jedes der beiden Cluster angegeben, in Klammern steht zusätzlich der Median.

Durchschnittl. Preisempfehlung Cluster 1	Durchschnittl. Preisempfehlung Cluster 2
9974 Dollar (7474)	6283 Dollar (4295)

Dichtebasierte Verfahren

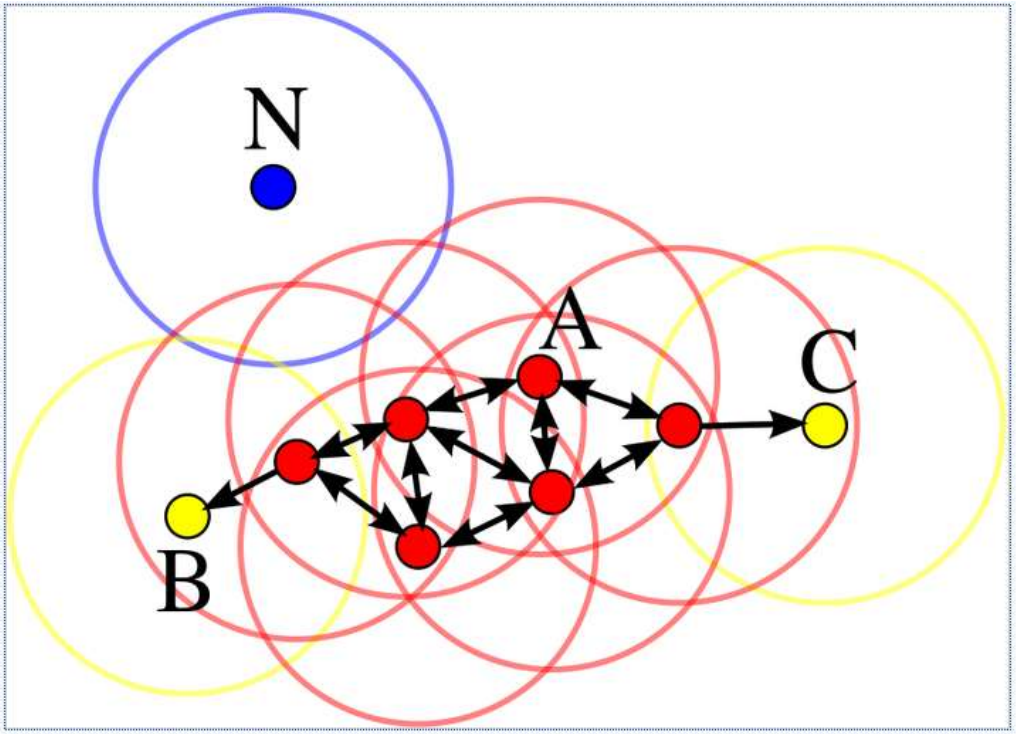
- Bei dichtebasiertem Clustering werden Cluster als Objekte in einem mehr-dimensionalen Raum betrachtet, welche dicht beieinander liegen, getrennt durch Gebiete mit geringerer Dichte.
- Dichtebasierte Verfahren
  - Dichtebasierte Clusterverfahren wurden entwickelt, um Cluster mit unregelmäßigen Formen darstellen zu können.
  - Diese Methoden erkennen Cluster als Regionen mit einer hohen Dichte an Objekten.
  - Die Cluster werden durch Regionen mit geringer Objektdichte voneinander getrennt.
  - Gruppen werden nach der Dichte der Punktewolke gruppiert.



DBSCAN

Dichtebasiertes Verfahren

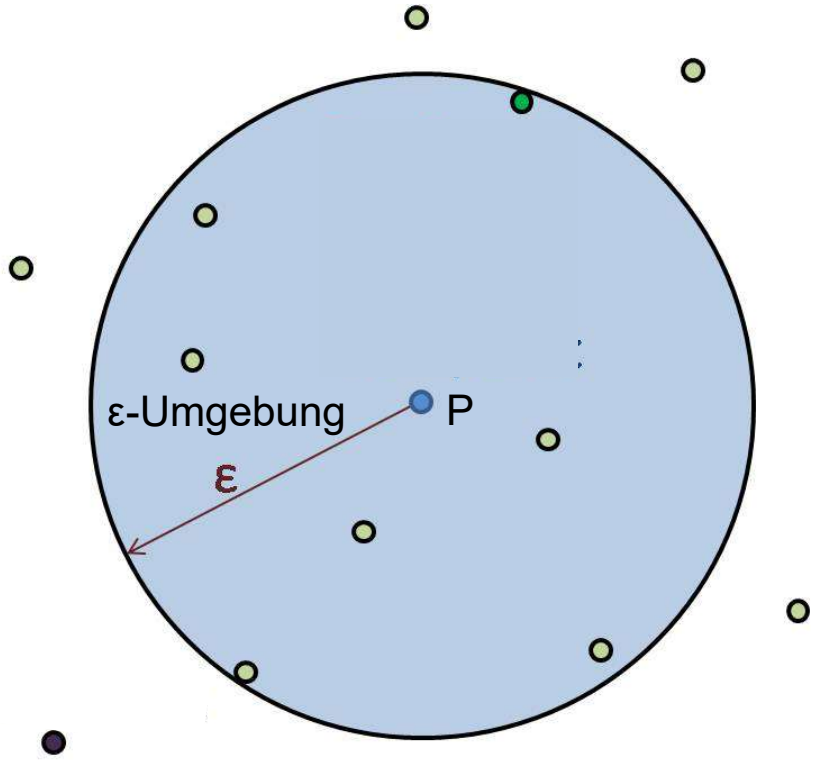
- Die bekannteste dichtebasierte Methode ist DBSCAN
  - *Density Based Spatial Clustering of Applications with Noise*, 1996 von Ester et al. veröffentlicht.
- DBSCAN clustert folgendermaßen:
  - Für jedes Objekt wird eine Nachbarschaft (konzentrischer kreis) festgelegt.
  - Wenn sich innerhalb dieser Nachbarschaft mindestens eine definierte Anzahl von Objekten *minPts* befindet, dann wird dieses Objekt ein Kern genannt.
  - Die Objekte in der Nachbarschaft eines Kernes können jedoch selbst wieder Kerne sein.
  - Kerne, die sich in der Nachbarschaft eines anderen Kerns befinden werden mit diesem verknüpft.
    - ► Solche Verbindungen werden als Region hoher Objektdichte oder als Cluster bezeichnet.
  - Bedingt durch diese Form des Clusterwachstums können die Cluster unregelmäßige Formen annehmen.
  - Alle Objekte, die schließlich nicht in einem Cluster enthalten sind werden als Störgeräusche aufgefasst.



Punkte bei A sind Kernpunkte. Punkte B und C sind *dichte-erreichbar* von A und dadurch *dichte-verbunden* und gehören zu dem gleichen Cluster. Punkt N ist weder ein Kernpunkt, noch dichte-erreichbar, also Rauschen. ($minPts$=3 oder $minPts$=4)

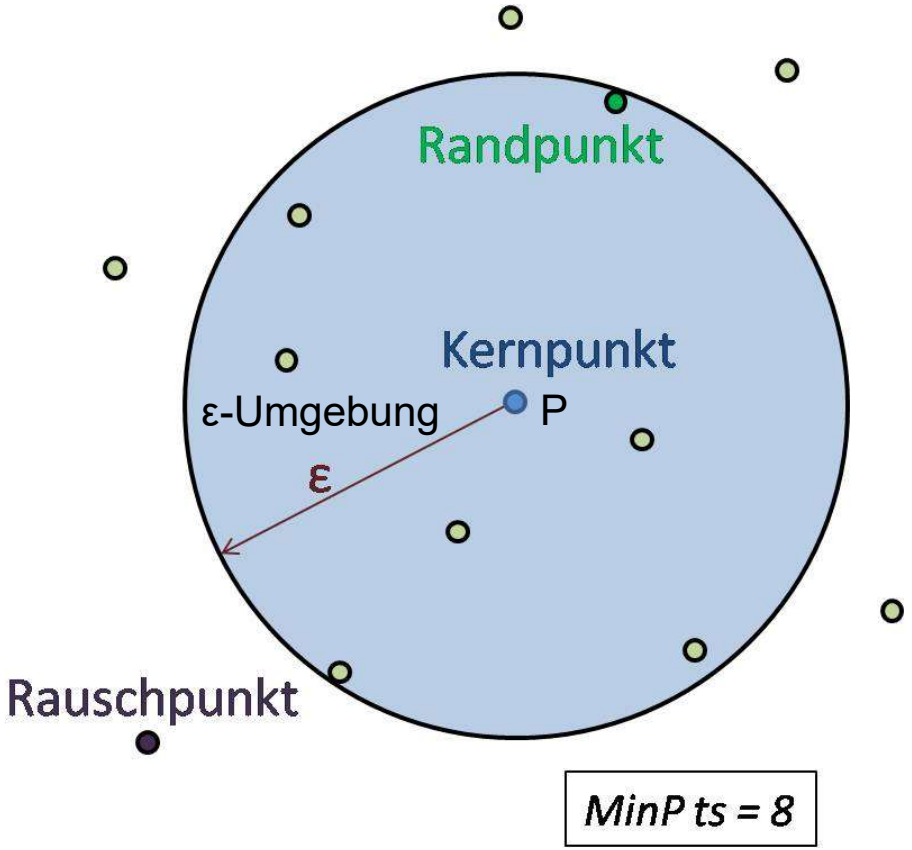
DBSCAN
Vorgehensweise

- Zunächst muss man sich überlegen, wie die Dichte eines speziellen Datenpunkts bemessen werden kann.
  - Beim zentrumsbasierten Zugang wird die Dichte für den Punkt p durch die Anzahl der Punkte in einer ε-Umgebung für ein zu bestimmendes ε geschätzt.
  - Die ε-Umgebung von p besteht aus allen Punkten, die höchstens den Abstand ε zu p besitzen.
- Ist der Radius ε sehr groß, so besitzen alle Datenpunkte die Dichte m (= Anzahl Datenpunkte), da dann für jeden Punkt p alle Punkte in der ε-Umgebung von p liegen.
  - Wählt man ε zu klein, so hat jeder Punkt die Dichte 1.
    - ► Daher ist die Bestimmung von ε im Vorfeld der Analyse von zentraler Bedeutung.
  - Ein weiterer Parameter, den wir für den DBSCAN-Algorithmus benötigen, ist MinPts.
    - ► Dieser legt fest, wie viele Punkte in der ε-Umgebung von einem Punkt p liegen müssen, damit p zu einem Cluster gehört.



Daten in drei verschiedene Kategorien

- Ein Datenpunkt liegt entweder im Inneren einer dichten Region, auf dem Rand einer solchen oder in einem spärlich besetzten Gebiet.
- Kernpunkt:
  - Die Anzahl der Datenpunkte in der ε-Umgebung des Kernpunkts beträgt mindestens MinPts.
- Randpunkt:
  - Ein Randpunkt ist kein Kernpunkt, liegt aber in der ε-Abstand eines Kernpunktes.
- Rauschpunkt:
  - Ein Rauschpunkt ist weder Kern- noch Randpunkt.



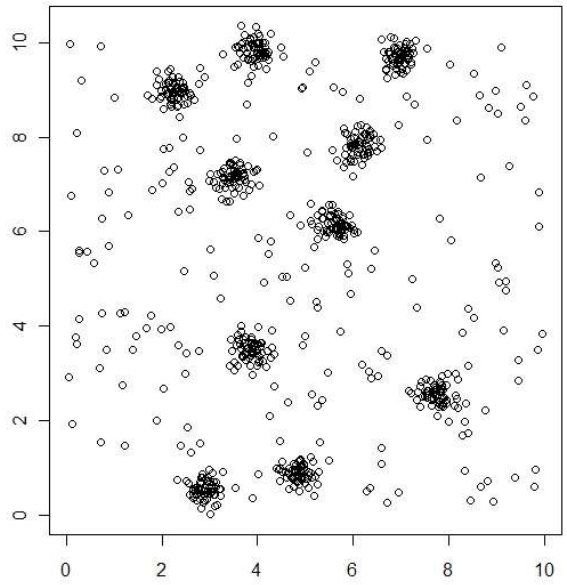
DBSCAN
Algorithmus

- Informell:
  - Es landen zwei Kernpunkte, deren Abstand höchstens ε beträgt, im gleichen Cluster, Randpunkte werden dem Cluster eines entsprechenden Kernpunktes zugeordnet und Rauschpunkte werden vernachlässigt.
- Zeit- und Speicherplatz-Aufwand
  - Der Zeitaufwand beträgt $O(m \times t)$, wobei t die Zeit zum Zählen der Punkte in einer ε-Umgebung ist, im Worst-Case daher $O(m^2)$.
    - Das bedeutet, im schlimmsten Fall benötigt man m^2 viele elementare Rechnungen, wobei m die die Anzahl der Datenpunkte ist.
- Auswahl der DBSCAN-Parameter
  - Damit DBSCAN ein „gutes“ Clustering liefert, muss man die Werte der Parameter ε und MinPts „geschickt“ wählen.
  - Der übliche Ansatz zur Bestimmung dieser Werte, ist die Betrachtung der Distanz eines Punktes p zu seinem k.-nächstem Nachbarn, also dem Punkt, der den k-größten Abstand zu p hat.
  - Diese Distanz werden wir mit k - dist bezeichnen.
  - Für Punkte innerhalb eines Clusters ist k - dist in der Regel klein, für Rauschpunkte dagegen vergleichsweise groß.
  - Wir berechnen daher k - dist für alle Datenpunkte und zeichnen das Ergebnis aufsteigend in ein Diagramm.
  - Man sollte dann einen scharfen Übergang erkennen können, welcher eine gute Wahl für ε nahelegt. k sollte in etwa die Größe der zu erwartenden Cluster widerspiegeln, für zweidimensionale bietet sich k = 4 an.
  - MinPts geben wir schließlich den Wert von k.

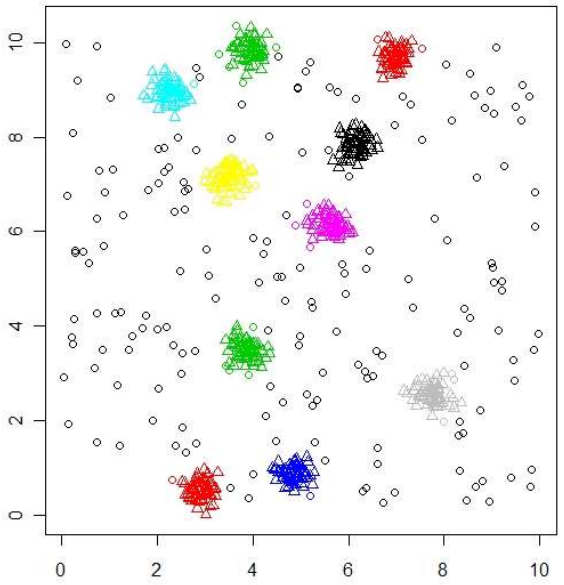
DBSCAN

Vor- und Nachteile von DBSCAN / Beispiele

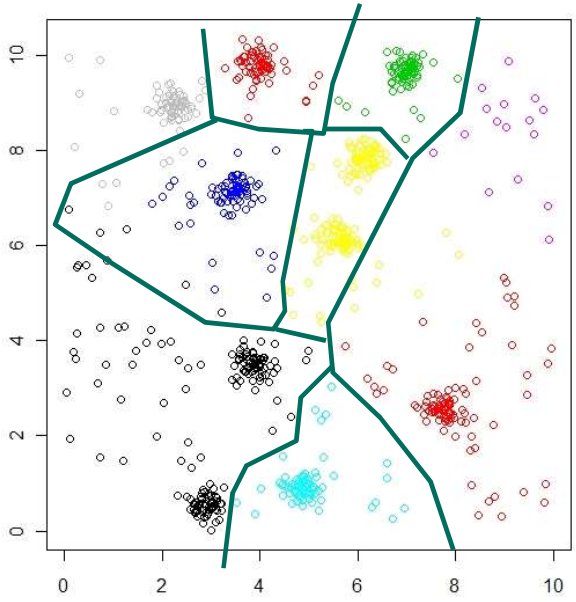
- DBSCAN filtert Rauschen meist sehr gut aus den Daten und kann Cluster beliebiger Form und Größe erfassen.
- Beispiel: Zunächst ein verrauschter Datensatz mit zehn natürlichen Clustern.
  - DBSCAN erkennt diese problemlos, wohingegen der K-Means-Algorithmus kein erwünschtes Clustering liefert; das Rauschen wird nicht unterdrückt.



Datensatz mit 10 Clustern und starkem Rauschen



Cluster in DBSCAN



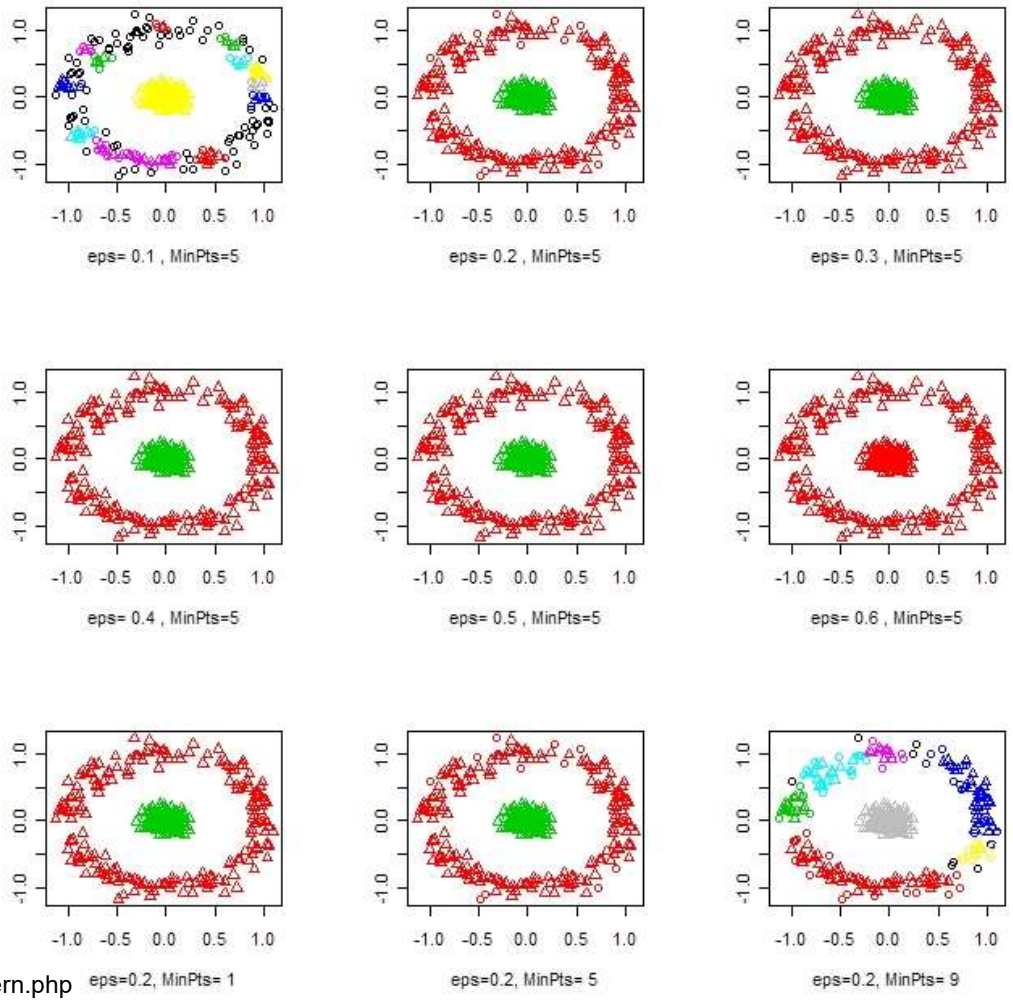
Cluster in K-Means

DBSCAN

Beispiel: Datensatz, welcher aus einem Ring und Häufungspunkt in der Ringmitte besteht.

- Bei einem Standartwert *MinPts = 5* kann man ε zwischen 0,2 und 0,6 frei wählen, um das natürliche Clustering zu erhalten.
- Auch bei der Wahl von *MinPts* muss man deutlich vom Standartwert abweichen, bis DBSCAN ein unerwünschtes Clustering wählt.

DBSCAN



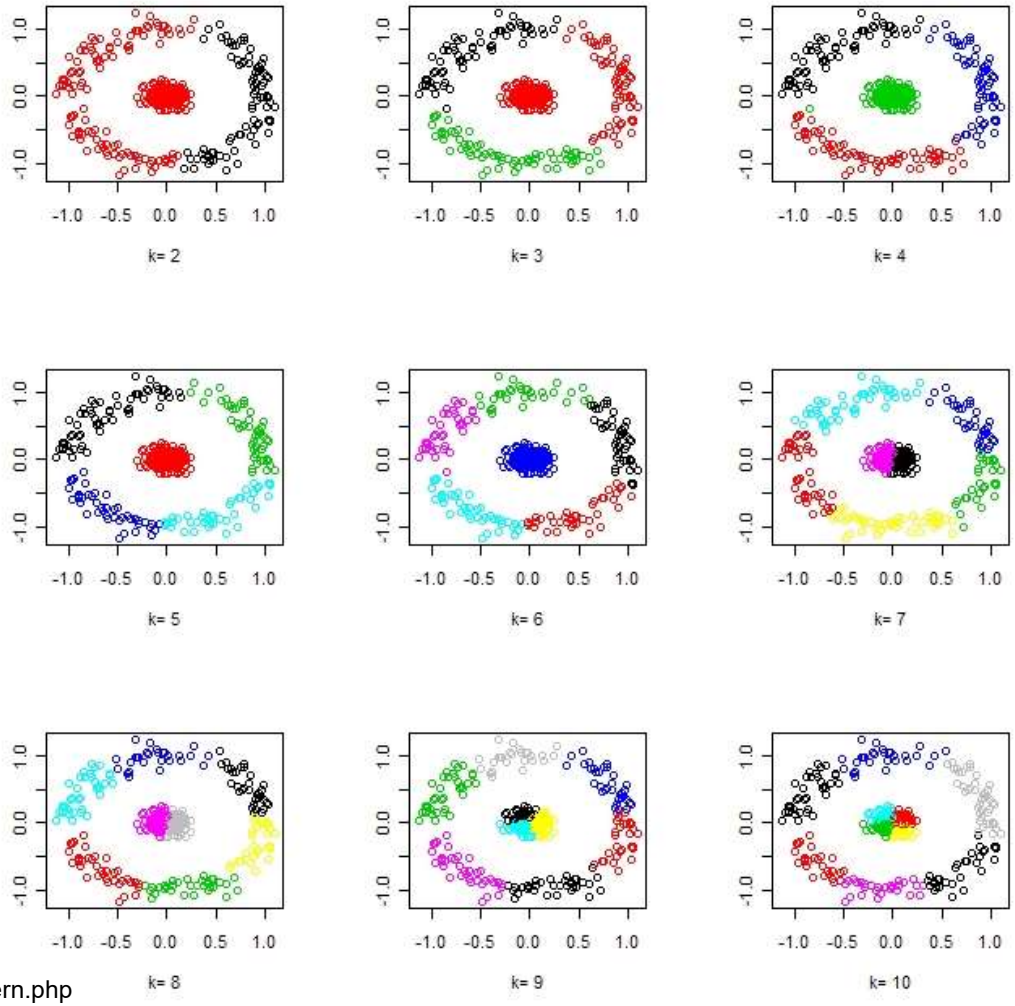
Quelle: http://www-m9.ma.tum.de/material/felix-klein/clustering/Methoden/Dichteverbundenes_Clustern.php

DBSCAN

Beispiel: Datensatz, welcher aus einem Ring und Häufungspunkt in der Ringmitte besteht.

- Bei einem Standartwert *MinPts = 5* kann man ε zwischen 0,2 und 0,6 frei wählen, um das natürliche Clustering zu erhalten.
- Auch bei der Wahl von *MinPts* muss man deutlich vom Standartwert abweichen, bis DBSCAN ein unerwünschtes Clustering wählt.
- DBSCAN behandelt Daten mit Clustern besonderer Form recht gut, wohingegen K-Means beim selben Datensatz erhebliche Probleme bekommt (s. rechts)

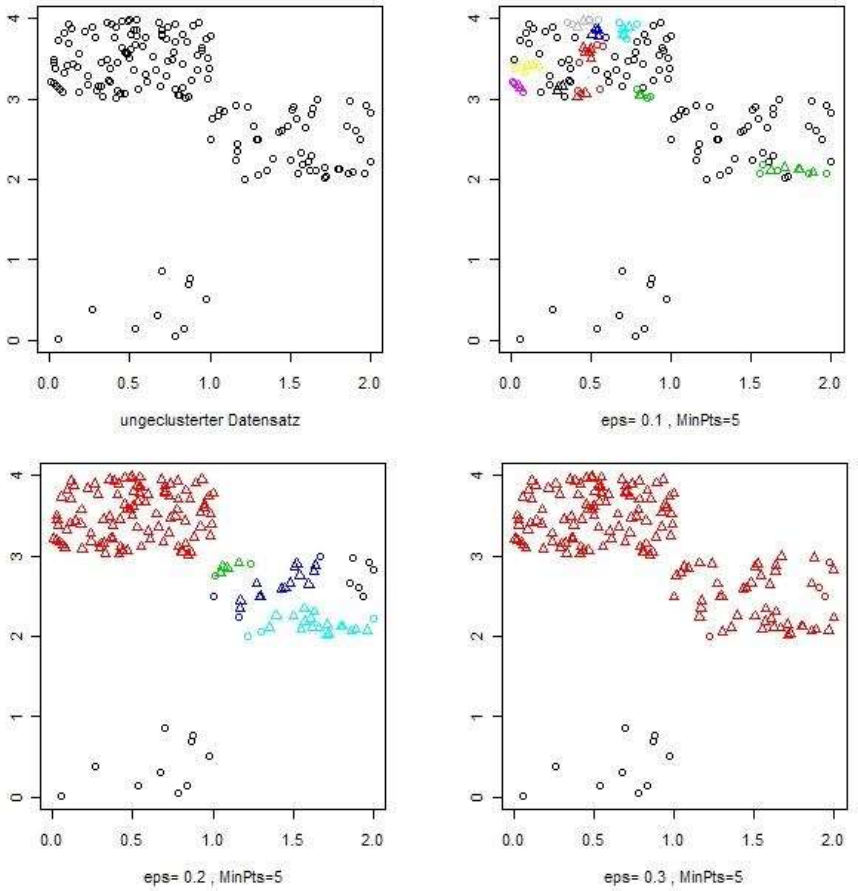
k-Means



Quelle: http://www-m9.ma.tum.de/material/felix-klein/clustering/Methoden/Dichteverbundenes_Clustern.php

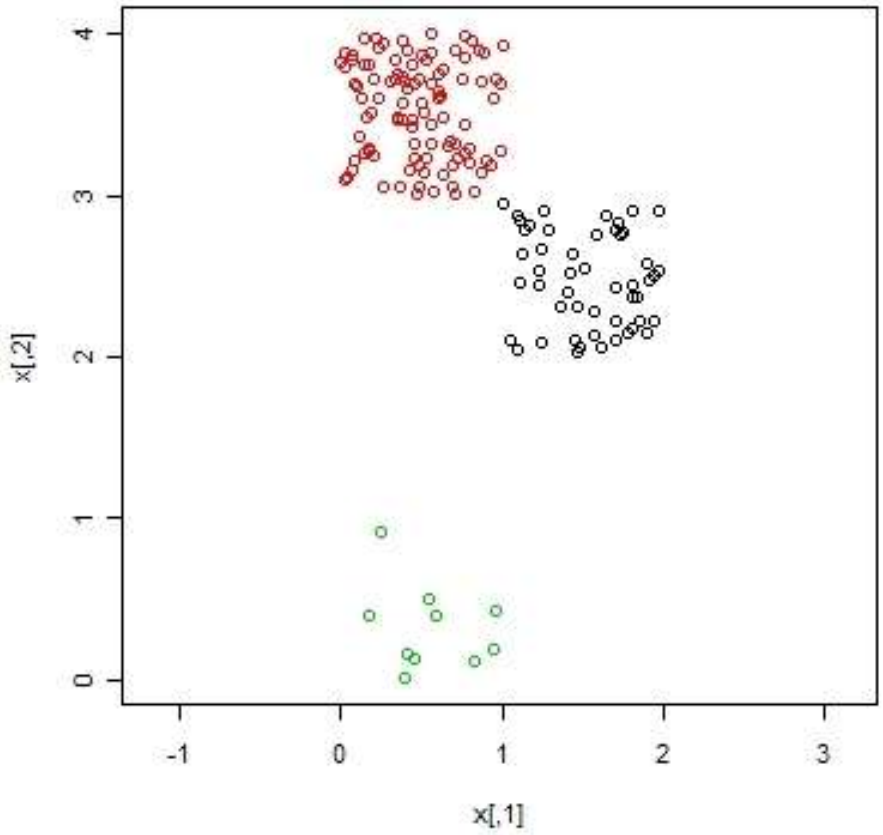
Schwierigkeiten des DBSCAN

- Andererseits hat der DBSCAN-Algorithmus Schwierigkeiten mit der Erkennung von Clustern unterschiedlicher Dichten.
  - Wählt man ε sehr groß, um „schwache“ Cluster zu finden, werden möglicherweise Rauschgebiete als Cluster erkannt.
  - Im rechten Beispiel findet man keine Parameterwerte für ε und MinPts, sodass DBSCAN die drei natürlichen Cluster (Rechtecke) erkennt, da die Punktdichten in den Objekten sehr unterschiedlich sind.



Schwierigkeiten des DBSCAN

- Andererseits hat der DBSCAN-Algorithmus Schwierigkeiten mit der Erkennung von Clustern unterschiedlicher Dichten.
  - Wählt man ε sehr groß, um „schwache“ Cluster zu finden, werden möglicherweise Rauschgebiete als Cluster erkannt.
  - Im rechten Beispiel findet man keine Parameterwerte für ε und MinPts, sodass DBSCAN die drei natürlichen Cluster (Rechtecke) erkennt, da die Punktdichten in den Objekten sehr unterschiedlich sind.
- K-Means erkennt die Cluster hingegen problemlos.
- Außerdem treten beim DBSCAN-Algorithmus Probleme bei hochdimensionalen Daten (Datensatz mit vielen Attributen) auf, da hier die Dichte schwierig zu definieren ist.
- Schließlich kann DBSCAN teuer werden, wenn die Berechnung der nächsten Nachbarn aufwändig ist.



Dichtebasierte Verfahren

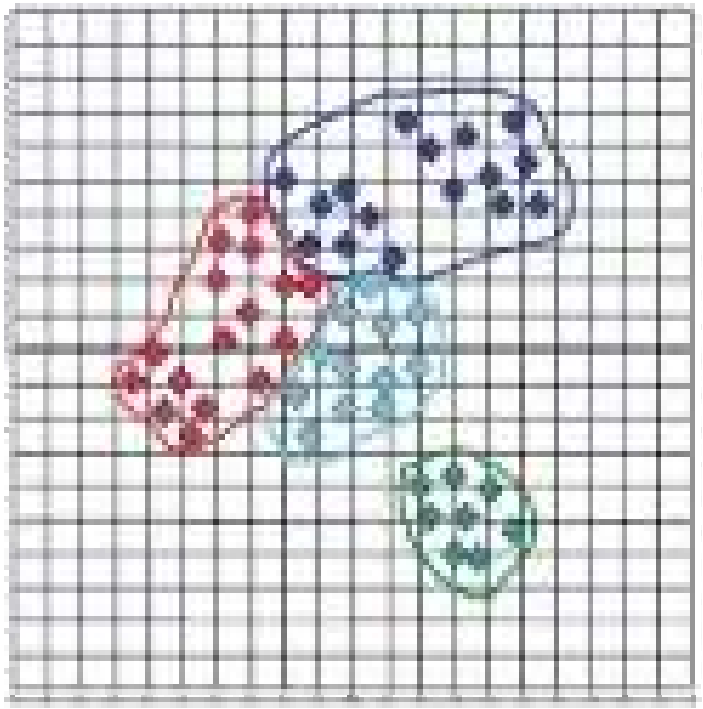
Weitere Beispiele

- OPTICS (Ordering Points To Identify the Clustering Structure)
  - Der Algorithmus erweitert DBSCAN, so dass auch verschieden dichte Cluster erkannt werden.
  - Die Wahl des Parameters ε ist nicht mehr so ausschlaggebend um die Clusterstruktur der Objekte zu finden.
- Maximum-Margin Clustering
  - Es werden (leere) Bereiche im Raum der Objekte gesucht, die zwischen zwei Clustern liegen. Daraus werden Clustergrenzen bestimmt und damit auch die Cluster.
  - Die Technik ist eng angebunden an Support-Vektor-Maschinen.

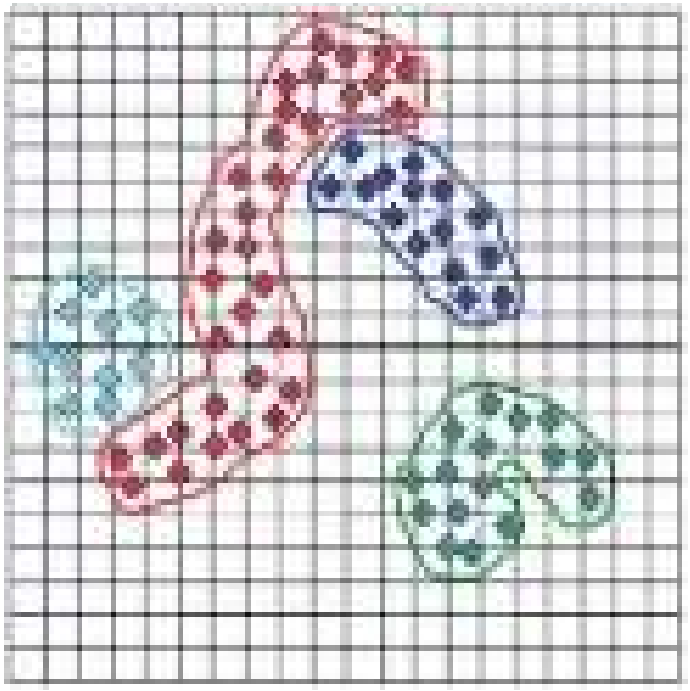
Vergleich: dichtebasiertes und hierarchisches Clusterverfahren

Beispiel: Dichtebasiert

- Beispiel 1



- Beispiel 2

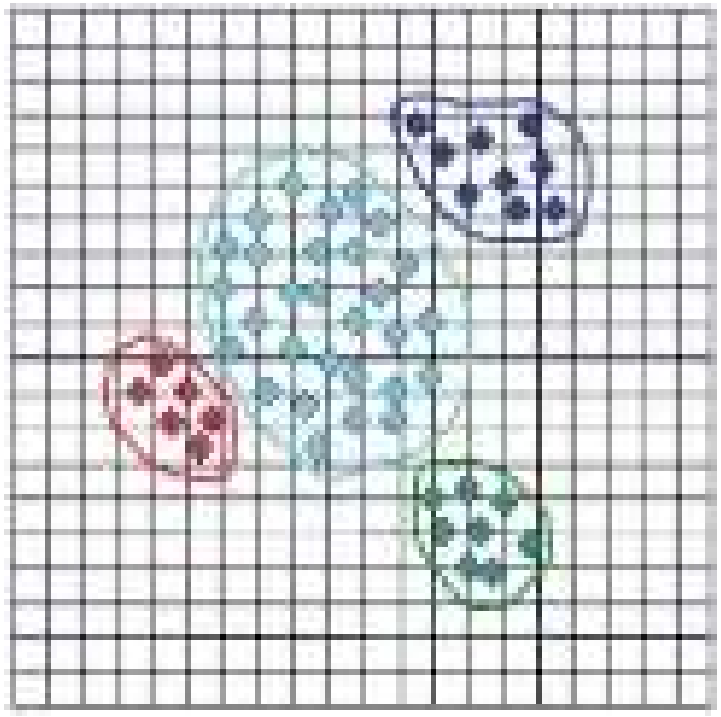


- Dichtebasierte Verfahren erkennen langgezogene Cluster oder Cluster mit unregelmäßigen Formen gut, wohingegen hierarchische Verfahren zu einer eher kugelförmige Clusterbildung tendieren.

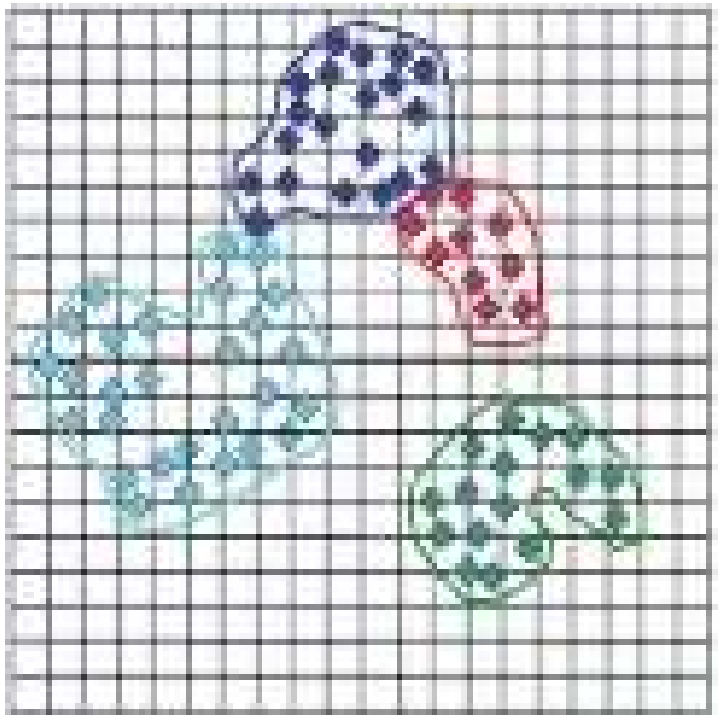
Vergleich: dichtebasiertes und hierarchisches Clusterverfahren

Beispiel: hierarchisch

- Beispiel 1



- Beispiel 2



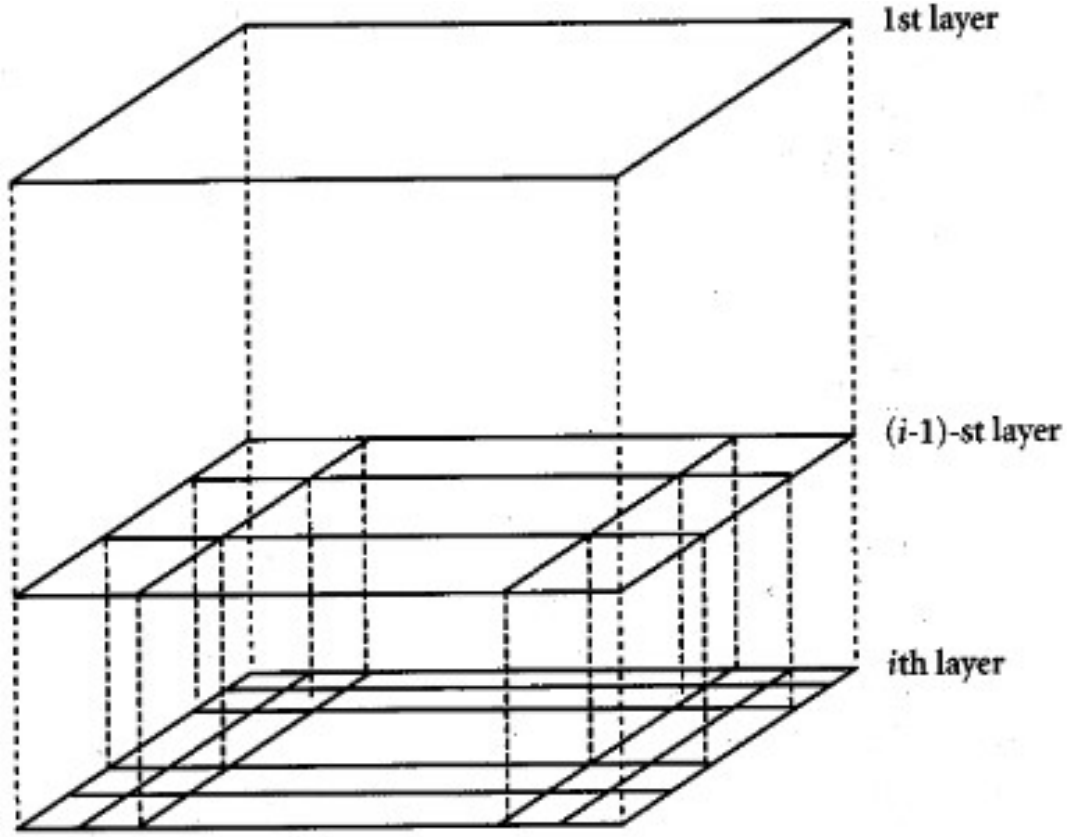
- Dichtebasierte Verfahren erkennen langgezogene Cluster oder Cluster mit unregelmäßigen Formen gut, wohingegen hierarchische Verfahren zu einer eher kugelförmige Clusterbildung tendieren.

Gitterbasierte Verfahren

- Gitterbasierte Verfahren unterteilen den gesamten Raum in eine endliche Anzahl von Zellen, welche eine Gitterstruktur formen, auf der alle Clusteroperationen durchgeführt werden.
  - Da die Objekte im Raum meist nicht gleichmäßig verteilt sind existieren in der Regel Zellen mit einer höheren Dichte an Objekten.
- Das Ziel gitterbasierter Methoden ist, den Raum so zu unterteilen, dass effizient nach Clustern gesucht werden kann.
  - Sie sind somit insbesondere für große und hochdimensionale Suchräume geeignet.
- Eine weitere Strategie bei der Suche nach dicht besetzten Gebieten ist, die Dimension des Suchraums zu verkleinern.
  - Wird in diesem niedriger dimensionalen Raum keine Häufung von Objekten gefunden, so ist dies auch in höheren Dimensionen nicht zu erwarten.
- Aufgrund der rechteckigen Form der Zellen können die Cluster jedoch bei vielen Methoden nur horizontal oder vertikal abgegrenzt werden.
  - Um differenzierte, unregelmäßige Cluster darstellen zu können, muss die Gitterstruktur entsprechend fein gewählt werden. Dies vermindert jedoch die Effizienz der Algorithmen.
- Gitterbasierte Methoden sind beispielsweise STING (Statistical Information Grid), WaveCluster oder CLIQUE (Clustering in Quest).

Gitterbasierte Verfahren

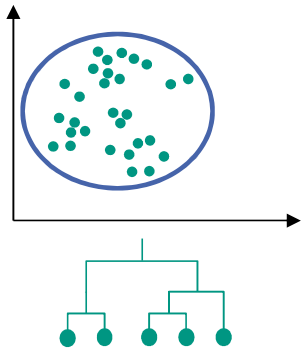
- Ausgehend von der detailliertesten Ebene werden für alle Zellen statistische Maße (z.B. Anzahl, Mittelwert, Standard-abweichung, Min, Max, Verteilungstyp) berechnet, die nach „oben“ aggegriert werden.
  - Cluster können durch Zusammenfassung einzelner Zellen wachsen.
- Vorteil:
  - Effizienz (nur abhängig von #Zellen, nicht von #Objekten)
- Nachteil:
  - Clustergrenzen können nur horizontal oder vertikal sein.



Modellierung der Daten – Data Mining

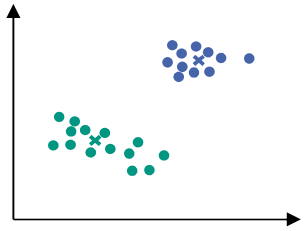
Clustering – Verschiedene Verfahren

Hierarchisch



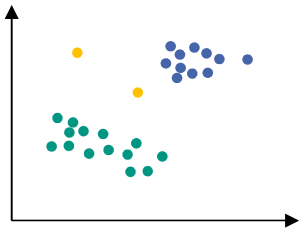
- Verschmelzen des geringsten Abstands
- Bis alle im gleichen Cluster

Partitionierend



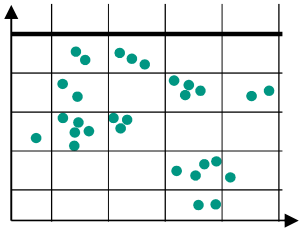
- Anzahl Cluster vorgegeben
- Random Seed
- Zugehörigkeit bestimmen
- Clustermittelpunkte neu berechnen
- Iterationsanzahl oder bis sich nichts mehr ändert

Dichte-basiert



- Cluster, Nachbarn und Dichte gegeben
- Berechnung ob Kern, Nachbar oder Rauschen
- Nachbar von Kernen ausweiten
- Iterationsanzahl oder bis alle Punkte durchlaufen sind

Gitter-basiert

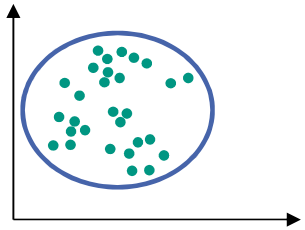


- Dichte der Gitter, Punkte
- Berechnung Punkte/Gitter
- Punkte > e

Modellierung der Daten – Data Mining

Clustering – Verschiedene Verfahren

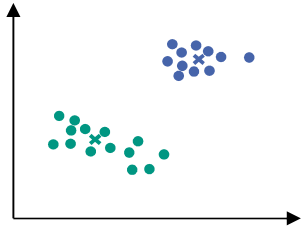
Hierarchisch



\+
- Cluster Hierarchie
- Keine Parameterwahl

−
- Keine Iteration
- Keine Optimierung

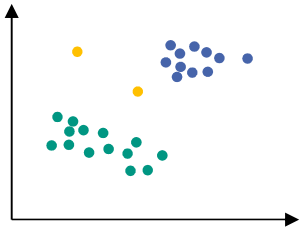
Partitionierend



\+
- Schnell
- Daten unabhängig

−
- Anzahl Cluster muss gewählt werden

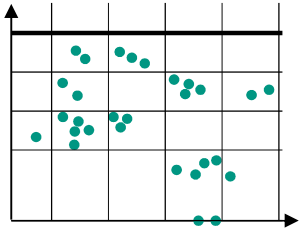
Dichte-basiert



\+
- Unregelmäßige Formen
- Rauschunterdrückung

−
- Sensibel auf Parameterwahl

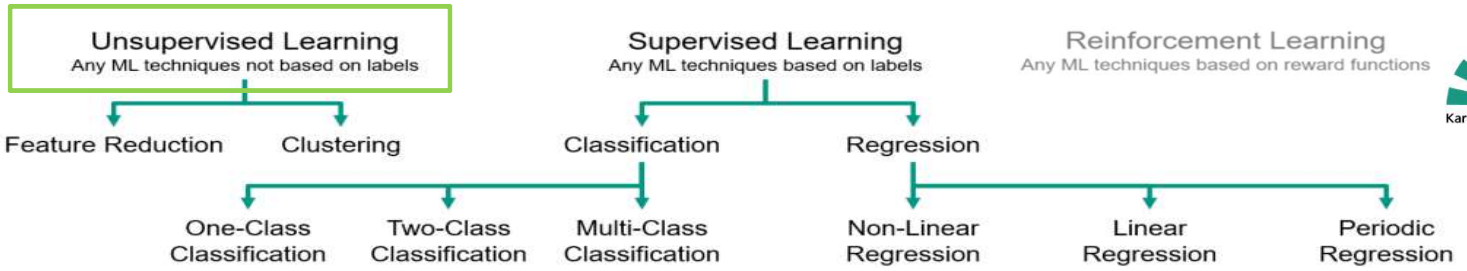
Gitter-basiert



\+
- Für Hochdimensionale Räume geeignet

−
- Eher grob
- Weniger Optimierungspotenzial

Machinelles Lernen



Algorithm	Abk.	Feature Reduction	Clustering	One-Class Classification	Two-Class Classification	Multi-Class Classification	Non-Linear Regression	Linear Regression	Periodic Regression
Faktorenanalyse		X							
Principal Component Analysis	PCA	X		X					
	K-means		X						
hierarchical cluster analysis	HCA		X						
DBSCAN			X						
One Class Support Vector Machine	OCSVM			X					
Isolation Forest				X					
	LODA			X					
(künstliche) Neuronale Netze	NN			X (Autoencoder)	X	X	X	X	X
Support Vector Machine	SVM				X				
Decision Tree					X	X			
Bayes-Klassifikation					X	X			
Random Forest						X			
Diskriminanzanalyse				x	x	x			
Logistic Regression							X	X	
Linear Regression								X	
Harmonic Regression									X
Nächste-Nachbar-Klassifikation	K-NN	x	x						

Hauptkomponentenanalyse (s. 6.2a)

Konstruktive Datenaufbereitung
Data Preparation

- Die Hauptkomponentenanalyse (Principal Component Analysis – PCA) ist eine variablenorientierte Methode, mit der versucht wird, einen hochdimensionalen Datensatz in einen niederdimensionalen Raum zu projizieren.



- Die zentrale Idee dabei ist, die Daten so auf eine 2-dimensionale Ebene zu projizieren, dass die gesuchten Zusammenhänge sichtbar werden.
  - Die sichtbar werdende Struktur der projizierten Daten hängt von der Richtung der Projektion ab.
  - Es stellt sich die Frage, wie eine Rotation der Daten (oder der Achsen - was für diese Zwecke dasselbe ist) gefunden werden kann, die ein Maximum an Information im projizierten Bild darstellt.

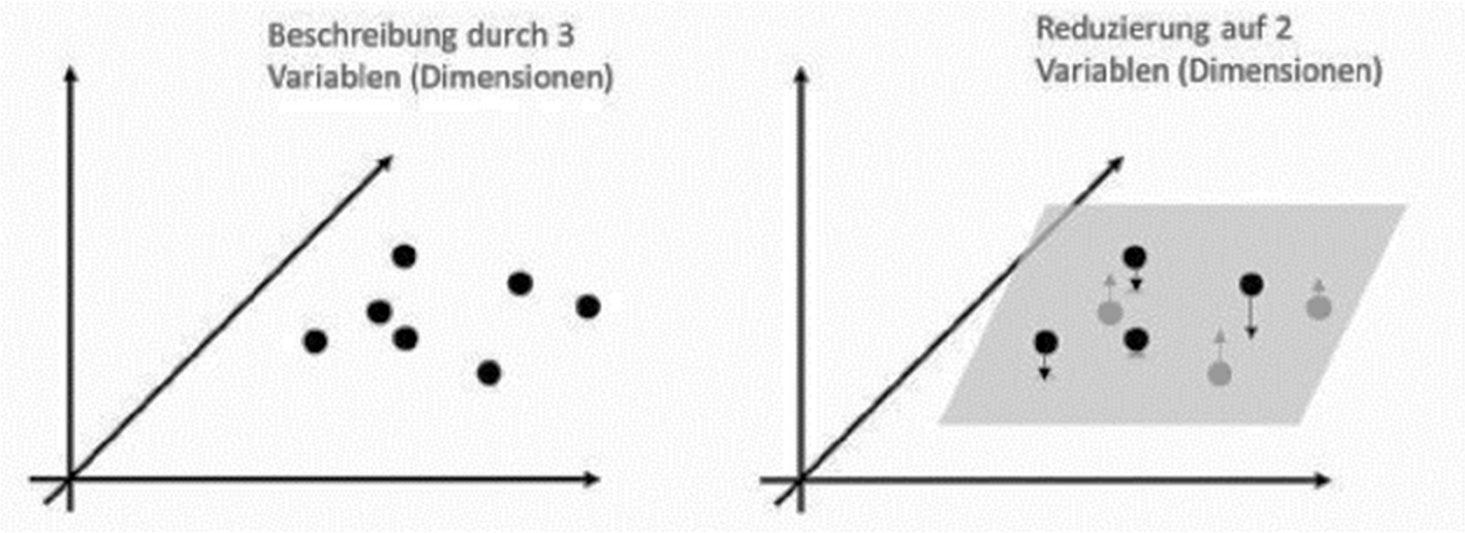
Faktorenanalyse

- Bei der Faktorenanalyse handelt es sich um ein Verfahren zur Zusammenfassung von Variablen.
  - Ziel ist es, die Anzahl der erklärenden Variablen zu verringern und „ähnliche“ Variablen zu Faktoren zusammenzufassen, die dann weitgehend voneinander unabhängig sind.
  - Die Bedeutung der Faktoren ist dabei nicht immer offensichtlich und muss interpretiert werden.
- Die Faktorenanalyse findet also dann Anwendung, wenn eine Vielzahl von Variablen vorhanden sind und davon ausgegangen werden kann, dass diese Variablen oft das Gleiche oder Ähnliches „aussagen“ und sich deshalb auf eine deutlich kleinere Anzahl aussagekräftiger Faktoren reduzieren lassen.
- Ein einfaches Beispiel hierzu bildet die Verdichtung der zahlreichen technischen Eigenschaften von PKWs auf wenige Dimensionen, wie z. B. Größe, Leistung, Prestige und Sicherheit.
- Dem Ziel der Faktorenanalyse – nämlich die Reduktion der Variablen und damit die Reduktion der Komplexität des Modells – steht auf der anderen Seite die Gefahr des Informationsverlustes gegenüber.
  - Diese Aspekte müssen gegeneinander abgewogen werden.

Prinzip der Faktorenanalyse

Bildlich kann man das an diesem Beispiel veranschaulichen:

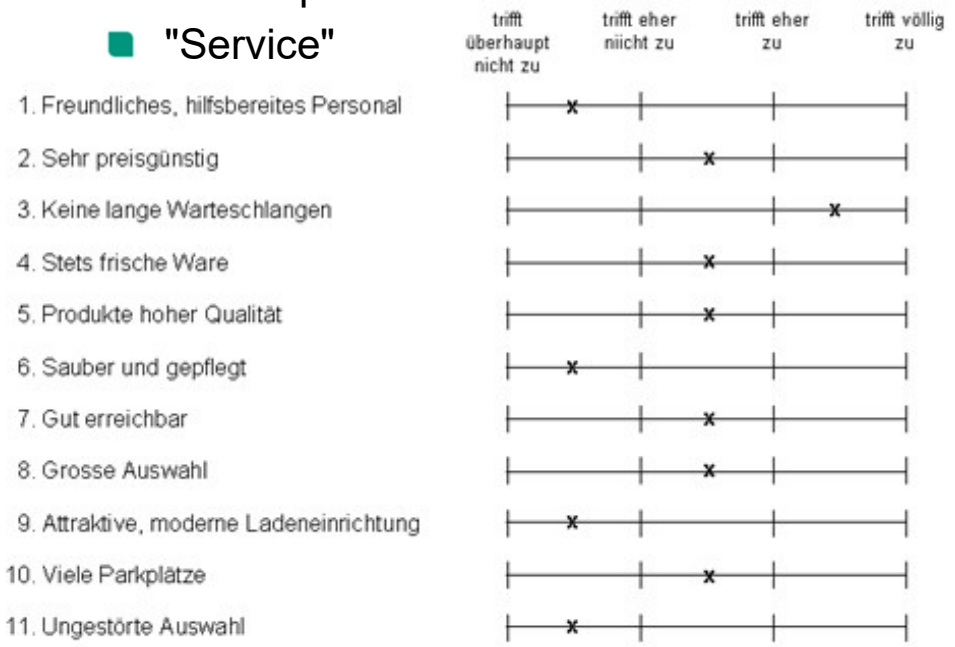
- Mit der Faktorenanalyse wird nun also versucht, eine zweidimensionale Fläche (zwei Faktoren) so in den Raum einzupassen, dass die Punkte mit möglichst wenig Verlust (Minimierung der Pfeillängen) auf dieser Fläche dargestellt werden können.
- Im Ergebnis können die Punkte auf der zweidimensionalen Fläche dargestellt werden, wobei die Reduktion der Komplexität mit einem gewissen Informationsverlust einhergeht.

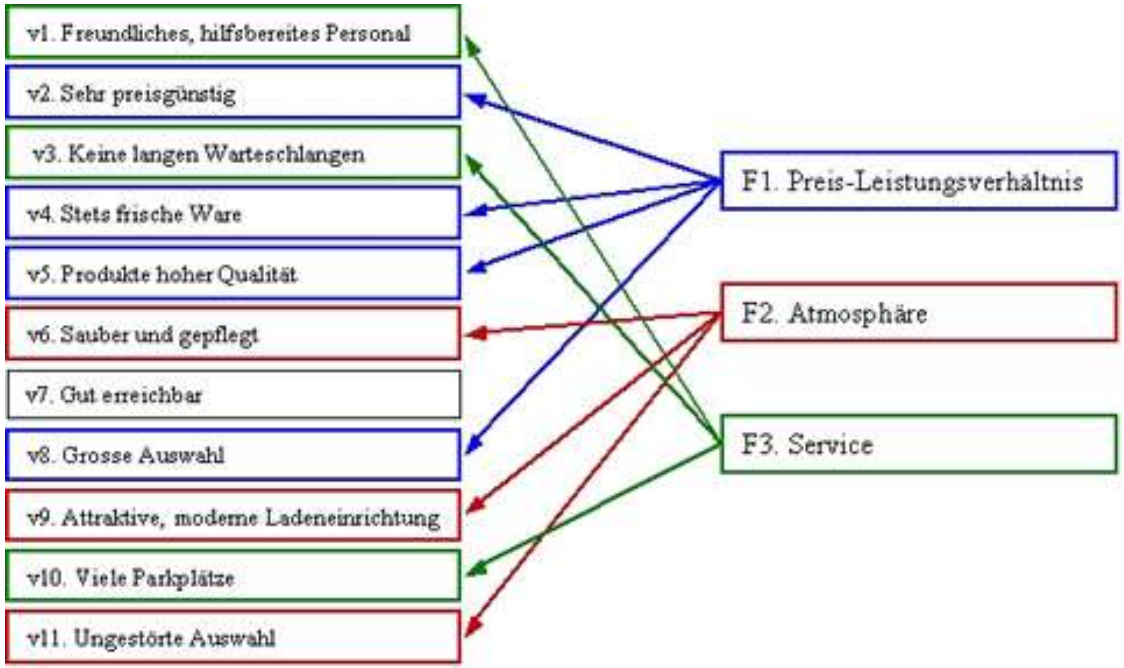


Faktorenanalyse

Illustratives Beispiel

- Bei der Untersuchung zum Image eines Ladengeschäftes wurde die Einschätzung von Kunden bezüglich elf Eigenschaften zum Image mit einer 4-stufigen Skala abgefragt.
- Ziel: die 11 Items zum Thema Image auf wenige Faktoren auf folgende 3 zu reduzieren:
  - "Preis-Leistungsverhältnis“
  - "Atmosphäre"
  - "Service"





s. http://www.mri.imh.unisg.ch/analysemethoden/datenanalyse/deskriptiv/multivariat/Faktorenanalyse.html

Hauptkomponentenanalyse vs. Faktorenanalyse

Zusammenfassung

- Die Hauptkomponentenanalyse (PCA) ist eine variablenorientierte Methode, mit der – vergleichbar zur Faktorenanalyse – versucht wird, einen hochdimensionalen Datensatz in einen niederdimensionalen Raum zu projizieren.
- Dabei versucht die PCA, die Varianzen der Objekte im ursprünglichen Raum möglichst gut mit dem neuen niederdimensionalen Raum abzudecken.
- Die PCA besteht darin, eine orthogonale Transformation der ursprünglichen Variablen in eine neue Menge unkorrelierter Variablen, die Hauptkomponenten, vorzunehmen.
  - Im Gegensatz dazu sind die Faktoren bei der Faktorenanalyse nicht zwingend orthogonal.
- Die Hauptkomponenten werden nacheinander in absteigender Bedeutung konstruiert.
- Die Hauptkomponenten sind Linearkombinationen der ursprünglichen Variablen.
  - Die erste Hauptkomponente wird so konstruiert, dass sie für den größten Teil der Variation verantwortlich ist.

Karlsruher Institut für Technologie



Fragen?

Inhalt IT2

6. Big Data & Maschinelles Lernen

- Definitionen, Verwendung (Charakteristik, Risiken, Chancen)
- Motivation und Anwendungsfälle
- Data Science Prozesse:
  - KDD
  - CRISP-DM
- CRISP-DM im Detail
  - Business Understanding
  - Data Understanding
  - Data Preparation
  - Modeling
  - Evaluation
- Infrastruktur für Big Data
  - Verfahren zur Datenanalyse
  - Beispiel zu ML-Bibliotheken: Tensorflow
  - Hardware für ML



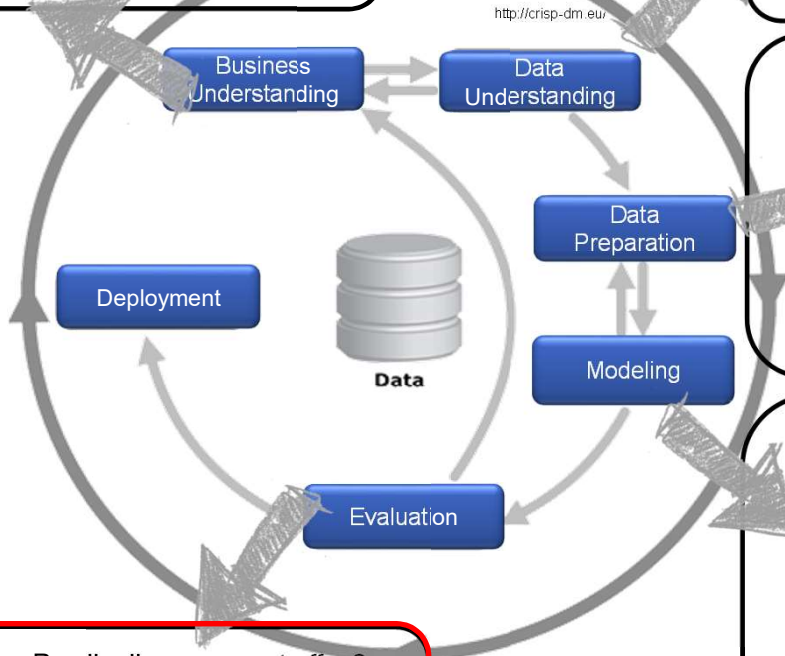
Big Data & Maschinelles Lernen

CRISP-DM im Detail

- Zielsetzung
- Rand-bedingungen

Analytic Aproach

- Regression
- Klassifikation
- Clustering
- Recommendation
- Outlier



Daten
- Struktur
- Statistik
- Visualisierung

Collect Initial Data
- Feature
- Type
- Struktur

Describe & Explore Data
- Statistik: Mean,Var, Median
- Visualisierung: Plots

Daten
Vor- & Aufbe-reitung

Select & Clean Data
- Auswahl
- Konsistenz
- Bereinigen

Construct Data
- Aggregation
- Aufteilung

Format Data
- Syntaktische Modifizierung
- Transformation
- Scaling

Algorithmus Selektion

Modeling Technique
- Group by Learning Style
  - Un- vs Supervised
  - Reinforcement
- Group by Similarity
  - Regression, Neighbor, Clustering, Decision Tree, NN, Bayes

Testdesign
- Splitting
  - Test, Train, Validate

Build & Assess Model
- ROC
- Confusion Matrix
- Cross Validation
- Precision/Recall
- AUC

Ergebnisse

Evaluate Results
- Randbedingungen getroffen? HW Constraints
- Summarize assessment results

Review of process
- Summarize process and highlight activites

Evaluation
(Data Mining-) Resultate auswerten

Resultate = (trainierte) Modelle + neue Erkenntnisse/Wissen

- Resultate verstehen und interpretieren
- Finales Statement:
  - Ursprüngliche Fragestellung gelöst?
  - Gibt es einen geschäftlichen Grund, warum das Modell unzureichend ist?
- Test auf Zielsystem → alle Anforderungen erfüllt? Deployment möglich?
- Neue Erkenntnisse müssen nicht mit dem (ursprünglichen) Ziel zusammenhängen
  - Bsp: mangelnde Datenqualität festgestellt
  - Neue Fragestellungen / Ziele

Evaluation

Beurteilung aller Prozessschritte

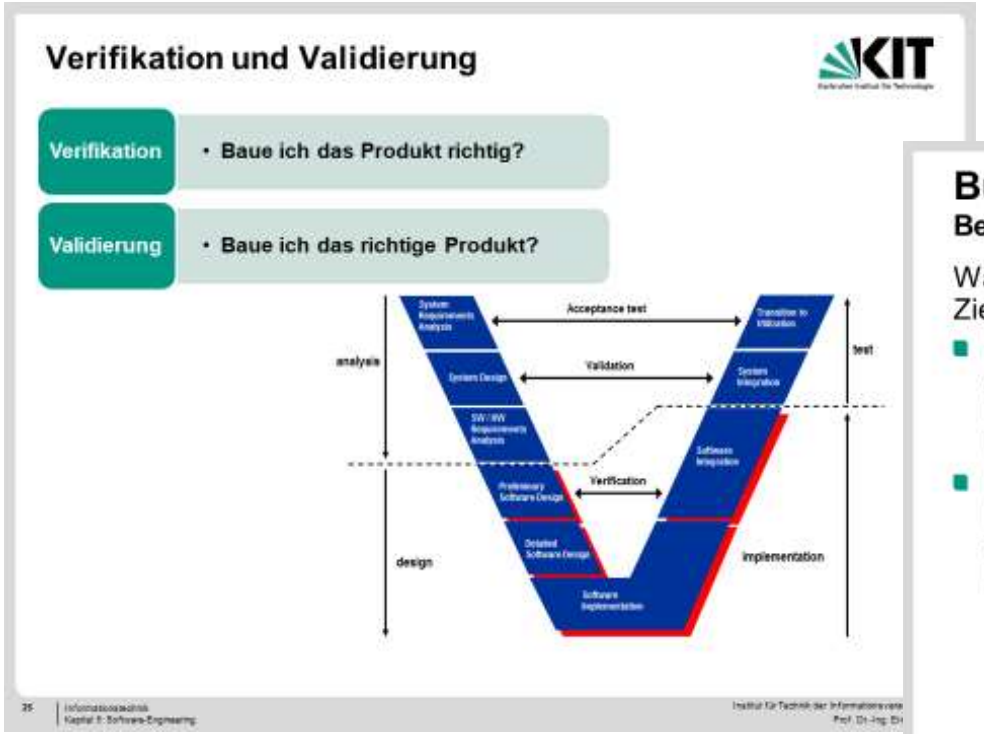
- Business-Ziele erreicht, trotzdem Review zur Qualitätssicherstellung nötig
- Alle getroffenen Entscheidungen überprüfen:
  - War es notwendig?
  - Wurde es optimal ausgeführt?
  - Inwiefern könnte es verbessert werden?
  - Beispiel: richtige Merkmalsauswahl, alle Algorithmen betrachtet, passende Daten gesammelt?



Fehler identifizieren und Alternativen für die nächste Iteration / Projekt finden

Evaluation

- Validierung entspricht der Prüfung der Business-Ziele
- Verifikation wird bei der Überprüfung des Prozesses durchgeführt



Business Understanding -
Bestimmung der Data Mining Ziele

Während betriebswirtschaftlicher Erfolg im Sinne der wirtschaftlichen Terminologie Ziele beschreibt, definiert ein Data Mining Erfolg ein Ziel im technischen Sinne

- Data Mining Ziele
  „Übersetzung" der wirtschaftlichen Ziele in Data Mining Ziele
  Data Mining Ziel spezifizieren durch z.B. Klassifikation, Vorhersage, Beschreibung oder Ähnliches
- Data Mining Erfolgskriterien
  Definition der Kriterien für einen erfolgreichen Projektausgang im technischen Sinn wie z.B. Genauigkeit von Vorhersagen
  Kriterien zur Modellbewertung

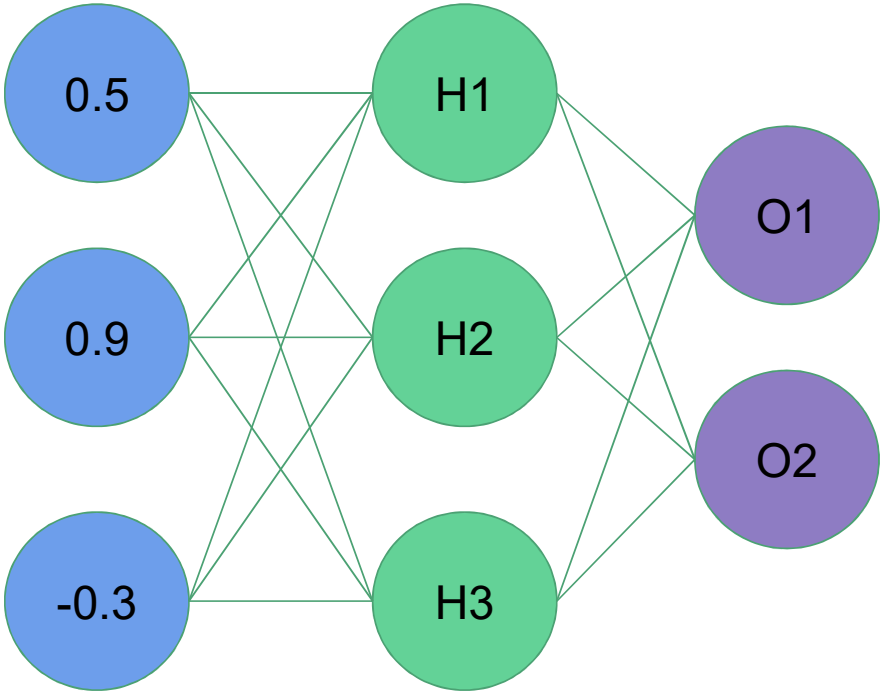
Backup

Beispiele

- Im Haushalt können Neuronale Netze bei Regelungsaufgaben wie der Benutzung einer Waschmaschine oder bei der Steuerung eines Staubsaugerroboters zum Einsatz kommen. Dieser kann mit Hilfe eines solchen Netzes seine Umgebung erlernen und kann bei jedem Saugvorgang seine abgefahrenen Pfade verbessern. Natürlich bleibt Ihre Wohnung nicht vollständig gleich. Hier wird einmal ein Tisch verrückt. Dort bekommt das Sofa einen neuen Standort. Da sich jedoch meist nicht alles gleichzeitig verändern wird, kann die Robotersteuerung durch ein Neuronales Netz die alten Kenntnisse, die sich noch verwenden lassen, weiter behalten und die neuen Kenntnisse zusätzlich lernen. Das heißt, mit Hilfe eines Neuronalen Netzes verblasst seine Erinnerung mit der Zeit und neue Informationen stellen sich in den Vordergrund, so dass er stets das momentan optimale Ergebnis liefert.

- Neuronale Netze können zur Optimierung einer Robotersteuerung eingesetzt werden. Industrieroboter z.B. bei der Produktion von Automobilen haben die Eigenschaft, dass sie aus einer Vielzahl von Gelenken bestehen, mit denen auch ein kompliziertes hineinreichen z.B. in eine Karosserie durchführbar ist. Die Gelenke können jedoch schnell unter Materialermüdung leiden, besonders wenn sie ruckartig bewegt werden und hohe Fliehkräfte wirken. Es geht darum, den Roboterarm möglichst schnell und möglich schonend zu bewegen, so dass alle Arbeiten geleistet werden können. Hierfür können neuronale Netze genutzt werden. Während des Betriebes verändert aufgrund von Abnutzungen der Roboterarm seine Eigenschaften. Eine Korrektur in dem Computerprogramm muss anhand verschiedener Parameter nachgehalten werden. Da sich die Parameter nur langsam ändern, kann auch hierfür ein Neuronales Netz genutzt werden.

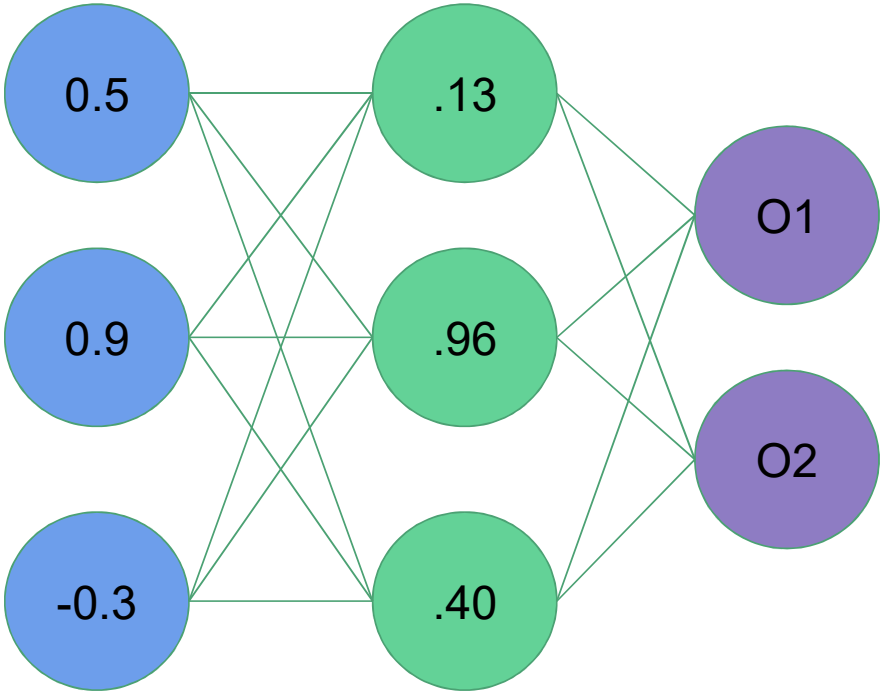
Inferenz
Neuronale Netze



H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

O1 Weights = (-3.0, 1.0, -3.0)
O2 Weights = (0.0, 1.0, 2.0)

Inferenz
Neuronale Netze

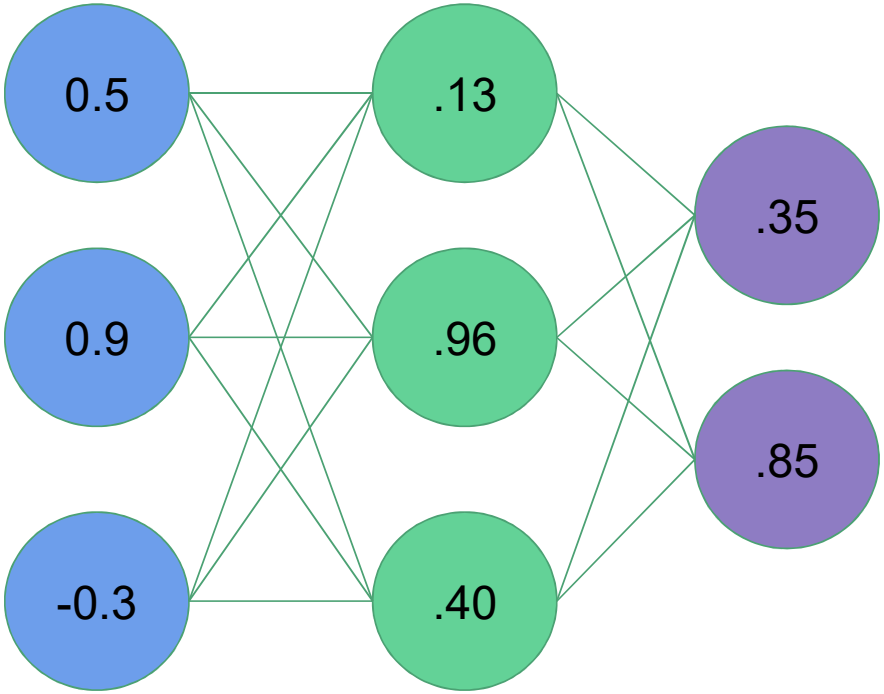


H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

O1 Weights = (-3.0, 1.0, -3.0)
O2 Weights = (0.0, 1.0, 2.0)

H1 = S(0.5 * 1.0 + 0.9 * -2.0 + -0.3 * 2.0) = S(-1.9) = .13
H2 = S(0.5 * 2.0 + 0.9 * 1.0 + -0.3 * -4.0) = S(3.1) = .96
H3 = S(0.5 * 1.0 + 0.9 * -1.0 + -0.3 * 0.0) = S(-0.4) = .40

Neuronale Netze
Inferenz



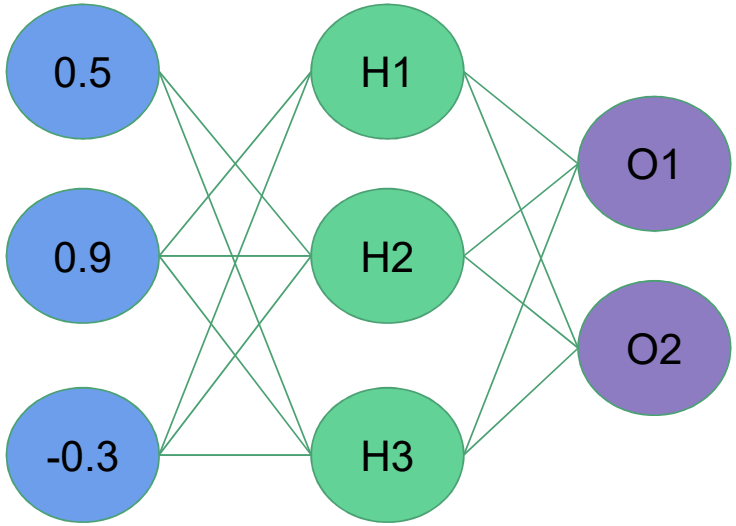
H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

O1 Weights = (-3.0, 1.0, -3.0)
O2 Weights = (0.0, 1.0, 2.0)

O1 = S(.13 * -3.0 + .96 * 1.0 + .40 * -3.0) = S(-.63) = .35
O1 = S(.13 * 0.0 + .96 * 1.0 + .40 * 2.0) = S(1.76) = .85

Matrix-Formulierung der Inferenz

Neuronale Netze



H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

Hidden Layer Weights

Inputs

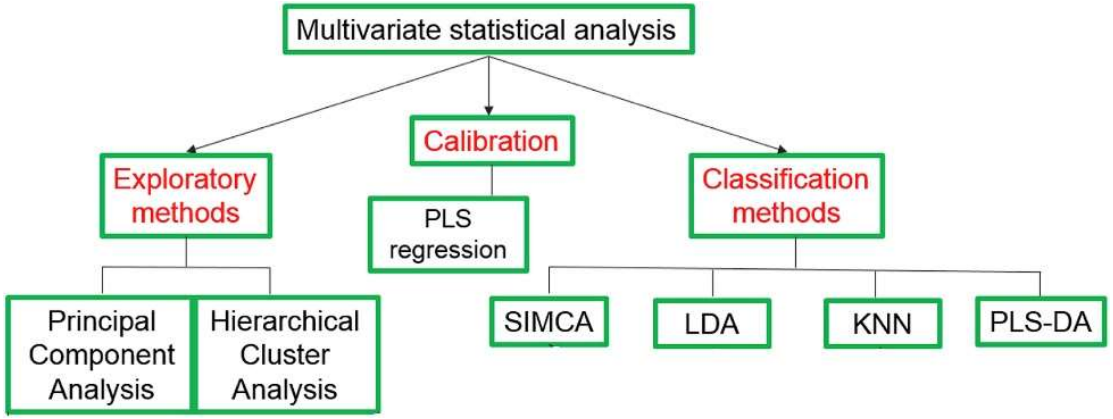
$$S\left(\begin{bmatrix}1.0 & -2.0 & 2.0\\ 2.0 & 1.0 & -4.0\\ 1.0 & -1.0 & 0.0\end{bmatrix} * \begin{bmatrix}0.5\\ 0.9\\ -0.3\end{bmatrix}\right) = S\left(\begin{bmatrix}-1.9 & 3.1 & -0.4\end{bmatrix}\right) = \begin{bmatrix}.13 & .96 & 0.4\end{bmatrix}$$

Hidden Layer Outputs

Unüberwachtes Lernen

„unsupervised learning“

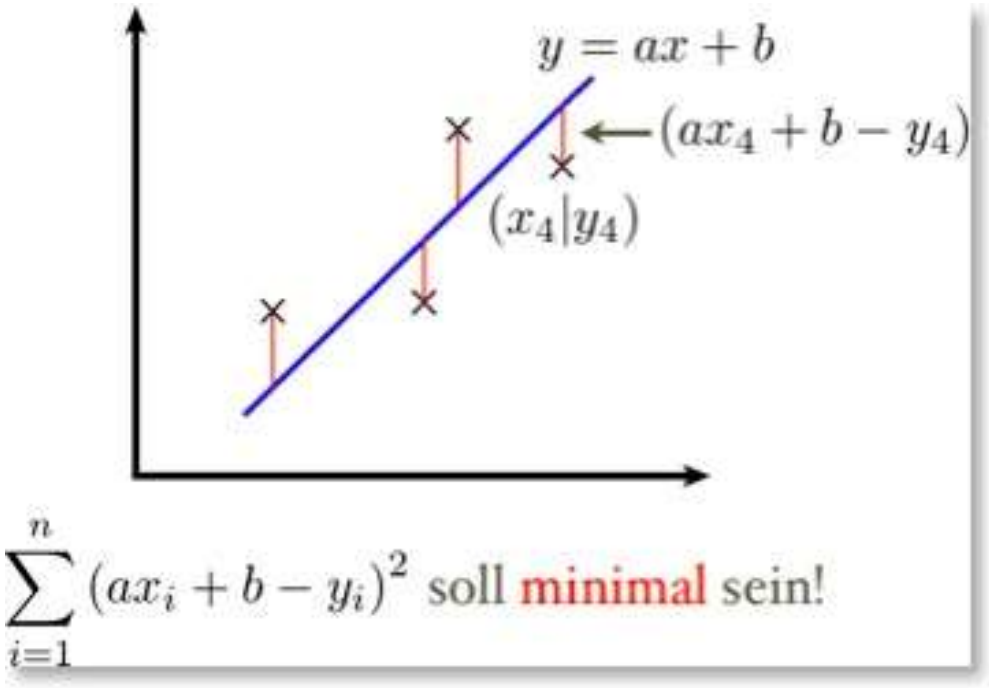
- Klassische Ballungen
  - k-means-Clustering
  - Agglomerative Hierarchical Clustering
- Begriffliche Ballungen („Conceptual Clustering“)
  - CLUSTER/2
  - Bildung von Begriffshierarchien („Concept Formation“)
- COBWEB
- CLASSIT
- Lernen durch Entdeckung
- BACON
- ABACUS
- ...
- Ohne Belohnung
- Ohne Vorwissen



Gauß'schen Methode der kleinsten Quadrate

Am Beispiel von 4 Messwerten

- Es soll eine passende Gerade (blau) gesucht werden.
- Nach einer Idee von **Carl-Friedrich Gauß** verwendet man als Maß für die Anpassung zunächst die vertikalen Abstände zwischen den tatsächlichen Messwerten und den entsprechenden Punkten auf der blauen Geraden.
- Für den vierten Punkt ist dieser Abstand angegeben.
- Diese Abstände werden quadriert und dann für alle Messwerte aufsummiert.
- Nach Gauß ist die beste Ausgleichslinie die, für die **die Summe der Quadrate der Abweichungen minimal ist**.
- Diese Idee heißt:
  - **Methode der kleinsten Fehlerquadrate**.
  - Das Stichwort „minimal“ deutet darauf hin, dass es sich um eine **Extremwertaufgabe** handelt.
  - Die Parameter a und b der Geraden müssen so gewählt werden, dass die bewusste Summe minimal wird.



Gütemaße für die Evaluation von Modellen

- Es existiert eine Reihe von Verfahren zum Testen der Güte statistischer Modelle.
- Die meisten solcher Verfahren liefern Kennzahlen, deren Quantität eine Aussage über die Modellgüte treffen. Ein Beispiel hierfür ist das Bestimmtheitsmaß R^2.
  - Dieses Maß erklärt den Anteil einer Variablen Y an der durch ein statistisches Modell erklärten Varianz.
  - Andere Gütetests beziehen sich auf die Beurteilung binärer Klassifikatoren.
  - Als Beispiel sei die Verifikation numerischer Wettervorhersagemodelle genannt.
  - An ihnen wird u.a. untersucht, wie genau ein vorhergesagter Parameter mit tatsächlich eingetretenen Ereignissen übereinstimmt (z.B. Niederschlag ja/nein).
  - Es werden Häufigkeitszahlen bestimmt, die angeben, wie oft eine Vorhersage bei eingetretenen Ereignissen korrekt war und wie oft inkorrekt.
  - Äquivalent dazu das Ganze für Nicht-Ereignisse.
  - Aus den erhaltenen vier Häufigkeiten lassen sich kategorische Gütemaße definieren.
  - Als Beispiele seien die Kennzahlen
    - Probability Of Detection
    - False Alarm Rate
    - True Skill Statisticsgenannt
  - Kontinuierliche Gütemaße beziehen sich dagegen auf kontinuierliche Parameter wie z.B. die Temperatur.
  - Hier findenübliche statistische Maße wiemittlerer Fehler,mittlerer absoluter Fehler,RMSE[1]unddieStandardabweichung Anwendung
  - Die Methodik, von der einige Verfahren erwähnt wurden, bezieht sich u.a. immer auf die Analyse eines kompletten Datensatzes, auf dem ein statistisches Modell beruht.
- Einen anderen Ansatzverfolgt dagegen die Kreuzvalidierung.
  - Ein Datensatz wird in mehrere Teile aufgeteilt.
  - Auf Basis eines Teils des Datensatzes wird ein statistisches Modellabgeleitet.
  - Das Modell wird auf den Rest des Datensatzes angewendet und die Abweichung zu den tatsächlichen Werten untersucht.
  - Das Verfahren dient also in erster Linieder Überprüfung derPrognosegüteeines Modells.

Gütemaße für die Evaluation von Modellen

Regression

- **R^2- Metric (R-Squared)**
  - Der Wert der R^2 kann Werte zwischen 0 und 1 annehmen
  - Nahe 1: Vorhersagen des Models sind gut
  - Nahe 0: Vorhersagen sind nicht besser als raten
  - $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$

$y_i = vorhergesagte\ Werte$

$\tilde{y}_i = korrekte\ Werte$

$\bar{y} = Median\ der\ Werte$