Fachhochschule Südwestfalen
University of Applied Sciences
Standort Hagen
Fachbereich Technische Betriebswirtschaft

Maschinelles Lernen

Masterstudiengänge Informatics and Business (Wirtschaftsinformatik) & Wirtschaftsingenieurwesen

Vorlesungsskript

Andreas de Vries

Version: 12. April 2024

Inhaltsverzeichnis

I Grundlagen **6**

1 Wahrscheinlichkeitsstheorie **7**
1.1 Zufallsvariablen und Wahrscheinlichkeiten 7
1.2 Der Satz von Bayes 10
1.3 * Bedingte Unabhängigkeit 13
1.4 Übungsaufgabe 15

2 Modelle und Theorien **17**
2.1 Methoden des logischen Schließens 18
2.2 Theorien 19
2.3 Was eigentlich ist ein Modell? 23
2.4 Ockhams Rasiermesser und Modellauswahl 27
2.5 Übungsaufgaben 31

3 Grundlagen des maschinellen Lernens **32**
3.1 Was ist maschinelles Lernen? 32
3.2 Statistische Variablen 34
3.3 Statistische Modelle 36
3.4 Verfahren des maschinellen Lernens 37
3.5 Probleme des maschinellen Lernens 41

4 Kurzeinführung in Python **44**
4.1 Grundlegende Sprachelemente 45
4.2 Kontrollstrukturen 47
4.3 Bibliotheken und Module 50
4.4 Übungsaufgaben 55

II Datenanalyse **58**

5 Regression **59**
5.1 Residuen und Bewertung von Regressionsmodellen 60
5.2 Lineare Regression in einer Dimension 62
5.3 Mehrdimensionale lineare Regression 64
5.4 Nichtlineare Regression 68
5.5 Übungsaufgaben 78

6 Datenanalyse mit Python **80**
6.1 Parametrische statistische Modelle in Python 80
6.2 Hauptkomponentenanalyse 83
6.3 Die Pipeline: Automatisierung der Datenanalyse 90

III Zeitreihen **95**

7 Zeitreihenanalyse **96**
7.1 Einführung 96
7.2 Stochastische Prozesse als Grundlage von Zeitreihen 98
7.3 Kausale lineare Prozesse 99
7.4 Die Random-Walk-Hypothese in der Ökonomie 101
7.5 Übungsaufgaben 102

8 Autoregressive Modelle **103**
8.1 Stochastische Prozesse in der Ökonomie 103
8.2 Definition und Eigenschaften autoregressiver Prozesse 104
8.3 Kausalität autoregressiver Prozesse 105
8.4 Übungsaufgaben 108

9 Autoregressive Modelle mit gleitendem Durchschnitt **109**
9.1 MA-Modelle 110
9.2 ARMA 112
9.3 Schätzung der Ordnung von ARMA-Modellen 114

10 Trends und Perioden: SARIMA-Modelle **119**
10.1 Zeitreihen mit Trends: Integrierte Prozesse 119
10.2 Vorgehensweise bei Trends 120
10.3 SARIMA 121
10.4 SARIMA-Modelle in `statsmodels` 122
10.5 Parameter zur Erzeugung eines SARIMAX-Modells 123
10.6 Fitten eines SARIMAX-Modells 124
10.7 Prognosen eines SARIMAX-Modells 124
10.8 Wahl eines SARIMA-Modells 125
10.9 Übungsaufgabe 126

A Appendix **127**
A.1 Solutions to selected problems 127
A.2 Heuser über Samuelsons Multiplikator 136

Literatur **137**

Internetquellen **140**

Vorwort

Das vorliegende Skript dient als Grundlage für den Kurs *Machine Learning* der Masterstudiengänge Wirtschaftsinformatik (Informatics and Business) und Wirtschaftsingenieurwesen der Fachhochschule Südwestfalen am Standort Hagen. Maschinelles Lernen ist als Begriff ein Hype. Es gibt eine Fülle an guter Literatur, Fachbücher ohne Ende – warum dann dieses Skript? Nun, genau weil es so viel gute Literatur und Quellen gibt. Eines der Hauptprobleme bei der Konzipierung dieses Kurses war, den Stoff so zusammen zu stellen, dass er in einem Semester behandelt werden kann. Ein anderes Problem war es, die vielen Zugänge aus ganz verschiedenen Schulen und Disziplinen in einen einheitlichen Formalismus zu bringen. Letzteres hört sich ja eigentlich recht einfach an – ist es aber nicht.

Ein einsemestriger Kurs über machinelles Lernen kann nur einen ersten Einblick in das weite Feld liefern. Dennoch war es von Anfang an das Ziel, dieses aktuell essenzielle und für die künftige Entwicklung in Wirtschaft und Wissenschaft vermutlich immer wichtiger werdende Gebiet den Studierenden angemessen zu vermitteln. Dazu gehört zunächst eine Theorie, die in ausreichendem Maß zu vermitteln ist. Lange und mühsame Literaturrecherchen sind dazu als Einstieg nicht effektiv, die Theorie sollte so präzise und knapp wie möglich, aber so viel wie nötig dargestellt werden. Zur Präsentation des Stoffs wurde zudem auf einen einheitlichen Formalismus Wert gelegt. Der Teufel steckt hier im Detail, schließlich kann die Variable x im in der Datenanalyse üblichen Formalismus eine ganz andere Bedeutung haben als in der Zeitreihenanalyse.[1]

Kurzum: Daher ein Skript! Beurteilen Sie als Lesende am Ende, ob es gelungen ist.

Der Inhalt. Der Kurs ist im Wesentlichen in die folgenden Teile gegliedert, wobei die ersten Teile das methodische und theoretische Rüstzeug liefern, um sie in den letzten Teilen anzuwenden:

1. Einführung in Python
2. Statistische Modelle
3. Datenanalyse (vor allem Regression, Hauptkomponentenanalyse, Naive-Bayes-Klassifikation)
4. Zeitreihenanalyse

Die Rolle statistischer Modelle kann dabei aus meiner Sicht gar nicht hoch genug bewertet werden, denn sie bilden die logische und formale Klammer für alle Ansätze des maschinellen Lernens. Auch jenen, die hier nicht erwähnt werden.

Die Lehrform. Die Theorie ist recht umfangreich und wird in einigen Vorlesungen skizziert. Im wesentlichen Teil des Kurses aber wird stark projektbezogen gelehrt, ein Ansatz, der sich allgemein für programmiernahe Fächer sehr bewährt hat.

[1] Falls dennoch Widersprüche im Formalismus dieses Skripts beobachtet werden, geben Sie mir Bescheid!

Literatur. Die dieses Skript ergänzende Literatur umfasst Werke zur Einführung in die zugrunde liegende Mathematik und zur Programmierung der mathematischen Verfahren in Python. Hier seien vor allem empfohlen:

- Für die Mathematik Backhaus et al. (2016), Backhaus et al. (2015), Cowpertwait und Metcalfe (2009), Downey (2011), Handl und Kuhlenkasper (2017), Hastie et al. (2009), James et al. (2013), K. P. Murphy (2012), Palma (2016), Sen und Srivastava (1990) und Tabachnick und Fidell (2018);
- für die Programmierung VanderPlas (2018), as well as the API documentations of the Python libraries Scikit-Learn (`https://scikit-learn.org/`) and Statsmodels (`https://www.statsmodels.org/`);
- und für beides Denis (2021) und Subasi (2020).

Hagen,
im April 2024 Andreas de Vries

Teil I

Grundlagen

1 Wahrscheinlichkeitsstheorie

Kapitelübersicht

1.1 Zufallsvariablen und Wahrscheinlichkeiten 7
1.2 Der Satz von Bayes 10
1.3 * Bedingte Unabhängigkeit 13
1.4 Übungsaufgabe 15

Dieses Kapitel widmet sich den Bezeichnungen und grundlegenden Begriffen der Wahrscheinlichkeitstheorie, die wir im Folgenden verwenden werden. Im besten Fall ist es eine einfache Wiederauffrischung Ihrer Kenntnisse über Stochastik, zumindest aber klärt es die Notationen.

Historisch nahm die Wahrscheinlichkeitstheorie 1654 ihren Anfang, als der französische Salontheoretiker Antoine Gombaud, der sich Chevalier de Méré nannte, einige Fragen über das Glücksspiel stellte, die daraufhin von beiden Mathematikern Pierre Fermat und Blaise Pascal in einer Reihe von Briefen betrachtet wurden. Auf Grundlage ihrer Ergebnisse veröffentlichte Christiaan Huygens die erste systematische Abhandlung über Wahrscheinlichkeitstheorie im Jahre 1657. Etwa ein halbes Jahrhundert später setzte Jakob Bernoulli Huygens Werk fort und veröffentlichte 1713 eine vorwiegend der Kombinatorik gewidmete Abhandlung. Kurz nach der französischen Revolution brachte Pierre-Simon Laplace 1795 sein bahnbrechendes Lehrbuch *Théorie analytique des probabilités* heraus.

Die strenge mathematische Fundierung der Wahrscheinlichkeitsrechnung wurde als sechstes der berühmten 23 Probleme von David Hilbert im Jahr 1900 gestellt, und Kolmogorow entwickelte den endgültigen axiomatischen Ansatz in seinem Artikel "Grundbegriffe der Wahrscheinlichkeitsrechnung“ in der Zeitschrift *Ergebnisse der Mathematik* **2**, Heft 3 (1933), der die Wahrscheinlichkeitstheorie als ein Spezialgebiet der Maßtheorie etablierte.[1]

1.1 Zufallsvariablen und Wahrscheinlichkeiten

Der Begriff der *Zufallsvariable* ist zentral in der Wahrscheinlichkeitstheorie. Vereinfacht und etwas salopp gesagt ist eine Zufallsvariable X eine Funktion, die einem zufälligen

Zufallsvariable X

[1] Bandelow (1989):§1.

Ereignis eine reelle Zahl x zuordnet.[2] Oder anders ausgedrückt: die Werte einer Zufallsvariablen X hängen ab von den Ereignissen ω eines Zufallsexperiments mit dem Ereignisraum Ω: [3]

$$X : \Omega \to \mathbb{R}, \qquad \omega \mapsto X(\omega) = x \tag{1.1}$$

Wahrscheinlichkeit $P(X = x)$

Somit werden die Wahrscheinlichkeiten der Ereignisse auf die Zufallsvariable vererbt, und wir bezeichnen mit $P(X = x)$ die Wahrscheinlichkeit, dass die Zufallsvariable X den Wert x annimmt. Also gilt insbesondere $0 \leqq P(X = x) \leqq 1$. Im maschinellen Lernen hat der Ereignisraum üblicherweise endlich viele Elemente, also auch der Wertebereich der Zufallsvariable. Die Werte X können daher als $\mathcal{X} = \{x_1, \ldots, x_N\}$ dargestellt werden, und die jeweiligen Wahrscheinlichkeiten $p_1, \ldots, p_N$ lauten

$$p_i = P(X = x_i) \qquad \text{und genügen} \qquad p_i \geqq 0, \qquad \sum_{i=1}^{N} p_i = 1. \tag{1.2}$$

kurze Notation $P(x)$, $P(A)$

Wir werden oft eine kürzere Notation verwenden und statt $P(X = x)$ einfach $P(x)$ oder $P(A)$ schreiben, wobei A die Aussage „$X = x$“ bedeutet, oder äquivalent $A =$ „X nimmt den Wert x an“. Wir werden manchmal auch komplexere Aussagen durch einen einzelnen Großbuchstaben wie $A, B, \ldots$ abkürzen. Eine Aussage kann hierbei die Werte wahr oder falsch annehmen.

Beispiel 1.1. Ein typisches Beispiel für eine Zufallsvariable ist die Augenzahl beim Werfen eines Würfels: Hier besteht der Ereignisraum aus den sechs verschiedenen Würfelergebnissen, die bei einem der sechs möglichen Ereignisse beim Werfen eines Würfels obenliegen. Für uns wesentlich ist, dass wir für eine Zufallsvariable X die Wahrscheinlichkeit

$$P(X = x)$$

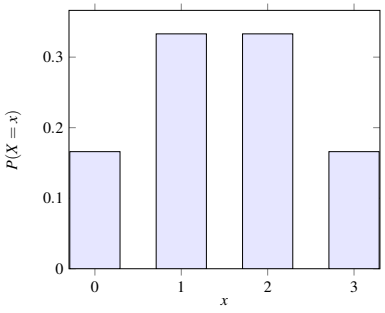
bestimmen können, das sie den Wert x annimmt. Ist zum Beispiel X die Zufallsvariable, die jedem Würfelwurf die Augenzahl ω modulo 4 zuordnet, so erhalten wir für sie die folgende Wertetabelle:

Augenzahl ω	1	2	3	4	5	6
$X(\omega)$	1	2	3	0	1	2

(1.3)

Damit lautet ihre Wahrscheinlichkeitsverteilung

x	0	1	2	3
$P(X = x)$	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$



(Wie kommt man auf die Werte? Vgl. dazu auch Übungsaufgabe 1.1.) □

Wahrscheinlichkeit einer Teilmenge

Ist $\mathcal{V}$ eine Teilmenge des Ergebnisraums, so gilt

$$P(\mathcal{V}) = P(x \in \mathcal{V}) = \sum_{x \in \mathcal{V}} P(X = x). \tag{1.4}$$

Ist $\mathcal{V}$ der gesamte Ergebnisraum, so gilt $P(\mathcal{V}) = 1$.

[2] Bauer (1991):S. 14.

[3] Im Englischen bezeichnet man ω üblicherweise mit *outcome* und Ω mit *sample space*.

Beispiel 1.2. *(Buchstaben in einem Dokument)* Ein weiteres Beispiel für eine Zufallsvariable ist ein Buchstabe, der zufällig aus einem englischen Text ausgewählt wird. Es gibt 27 Buchstaben a–z und ein Leerzeichen '–'. Die sich ergebenden Wahrscheinlichkeiten sind grafisch in Abbildung 1.1 illustriert. Hier werden die Wahrscheinlichkeiten $p_i = P(x_i)$

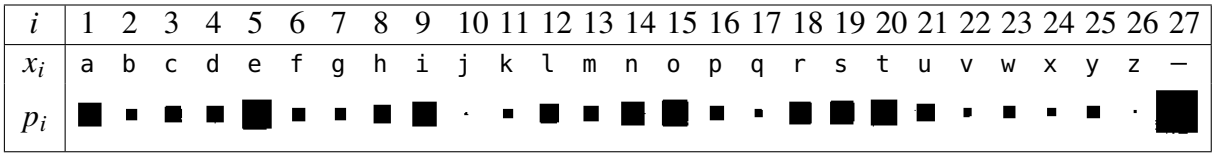


Abbildung 1.1. Wahrscheinlichkeitsverteilung über die Buchstaben in einem englischen Text. Die Größe eines Quadrats bezieht sich auf die Wahrscheinlichkeit $p_i = P(x_i)$. Modifiziert nach MacKay (2003:S. 22)

durch Quadrate dargestellt, deren Größe dem Wert von p_i entspricht. Wenn wir $\mathscr{V}$ als Vokale aus Abbildung 1.1 definieren, d.h., $\mathscr{V} = \{\mathtt{a}, \mathtt{e}, \mathtt{i}, \mathtt{o}, \mathtt{u}\}$, so gilt

$$\begin{aligned} P(\mathscr{V}) &= P(\mathtt{a}) + P(\mathtt{e}) + P(\mathtt{i}) + P(\mathtt{i}) + P(\mathtt{o}) + P(\mathtt{u}) \\ &= 0{,}058 + 0{,}091 + 0{,}060 + 0{,}069 + 0{,}033 = 0{,}311. \end{aligned} \tag{1.5}$$

(Die Wahrscheinlichkeitswerte sind entnommen aus MacKay (2003:S. 22)) □

Bei *zwei* Zufallsvariablen X und Y ist jedes Ergebnis ein geordnetes Paar (x, y), mit $x \in \mathscr{X} = \{x_1, \ldots, x_N\}$ und $y \in \mathscr{Y} = \{y_1, \ldots, y_M\}$. Wir nennen $P(x, y)$ die *gemeinsame Wahrscheinlichkeit* von X und Y, oder auch *bivariate Wahrscheinlichkeit* oder *Verbundwahrscheinlichkeit*. Das Komma und die Klammern sind bei geordneten Paaren optional, d.h. $xy \Leftrightarrow (x, y)$. Beachte, dass die beiden Zufallsvariablen X und Y hier nicht unabhängig sein müssen.

gemeinsame Wahrscheinlichkeit $P(x, y)$

Beispiel 1.3. Ein Beispiel für eine gemeinsame Wahrscheinlichkeit ist das geordnete Paar xy zweier aufeinander folgender Buchstaben („Bigramme") in einem englischen Text.[4] Die möglichen Stichprobenwerte sind geordnete Buchstabenpaare wie aa, ab, ac, ..., zz. Wir würden intuitiv erwarten, dass ab und ac wahrscheinlicher sind als aa und zz. Eine Schätzung der Bigrammhäufigkeiten ist grafisch in Abbildung 1.2 dargestellt. □

Mit der gemeinsamen Wahrscheinlichkeit $P(x, y)$ eines Zufallsraums mit endlich vielen Werten können wir die *Randwahrscheinlichkeit (marginal probability)* $P(x)$ von ihr dutch Summation über alle Werte von y:

Randwahrscheinlichkeit

$$P(x) = \sum_{y \in \mathscr{Y}} P(x, y) \tag{1.6}$$

Entsprechend erhalten wir die Randwahrscheinlichkeit $P(y)$ durch Summation über alle Werte von x:

$$P(y) = \sum_{x \in \mathscr{X}} P(x, y) \tag{1.7}$$

Mit der gemeinsamen Wahrscheinlichkeit $P(x, y)$ definieren wir dann die *bedingte Wahrscheinlichkeit* $P(x \mid y)$ als den Quotienten der gemeinsamen Wahrscheinlichkeit $P(x, y)$ und der Randwahrscheinlichkeit $P(y)$,

bedingte Wahrscheinlichkeit $P(x \mid y)$

$$P(x \mid y) = \frac{P(x, y)}{P(y)} \tag{1.8}$$

[4] MacKay (2003):S. 23.

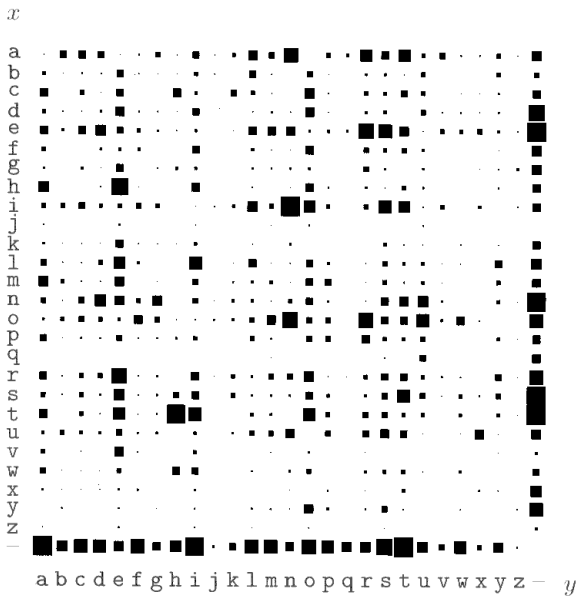


Abbildung 1.2. Die gemeinsame Wahrscheinlichkeitsverteilung $P(x, y)$ der 27×27 möglichen Bigramme xy in einem englischen Text. Die Größe des Quadrats (i, j) gibt die Wahrscheinlichkeit $p_{ij} = P(x_i, y_j)$ wieder. Modifiziert nach MacKay (2003:S. 23)

Wir sagen zu $P(x \mid y)$: „die Wahrscheinlichkeit, dass X gleich x ist, wenn Y gleich y gegeben“, oder kürzer „die Wahrscheinlichkeit von x gegeben y“.

A-Priori- / A-Posteriori-Wahrscheinlichkeit

Die Randwahrscheinlichkeit $P(x)$ wird oft auch die *A-Priori-Wahrscheinlichkeit (prior probability)* von x, und die bedingte Wahrscheinlichkeit $P(x \mid y)$ die *A-Posteriori-Wahrscheinlichkeit (posterior probability)* von x unter der Bedingung y.[5]

Beispiel 1.4. *(Beispiel 1.3 noch einmal)* Mit der gemeinsamen Wahrscheinlichkeitsverteilung $P(x, y)$ in Beispiel 1.3 und der Randwahrscheinlichkeitsverteilung $P(x)$ in Beispiel 1.2 kann die bedingte Wahrscheinlichkeitsverteilung $P(x \mid y)$ bestimmt werden. Sie ist in Abbildung 1.3 (a) skizziert. Zum Beispiel ist $P(x \mid y = \mathtt{u})$ die Wahrscheinlichkeit, dass der erste Buchstabe x ist, wenn der zweite ein u ist. Wie wir in Abbildung 1.3 (a) sehen können, sind die wahrscheinlichsten Werte o and n für x gegeben $y = \mathtt{u}$.

Entsprechend ist $P(y \mid x = \mathtt{q})$ die Wahrscheinlichkeit, dass der zweite Buchstabe y ist, wenn der erste ein q ist. Wie wir in Abbildung 1.3 (b) sehen, sind die wahrscheinlichsten Werte u und – für y, wenn $x = \mathtt{q}$ gegeben ist. □

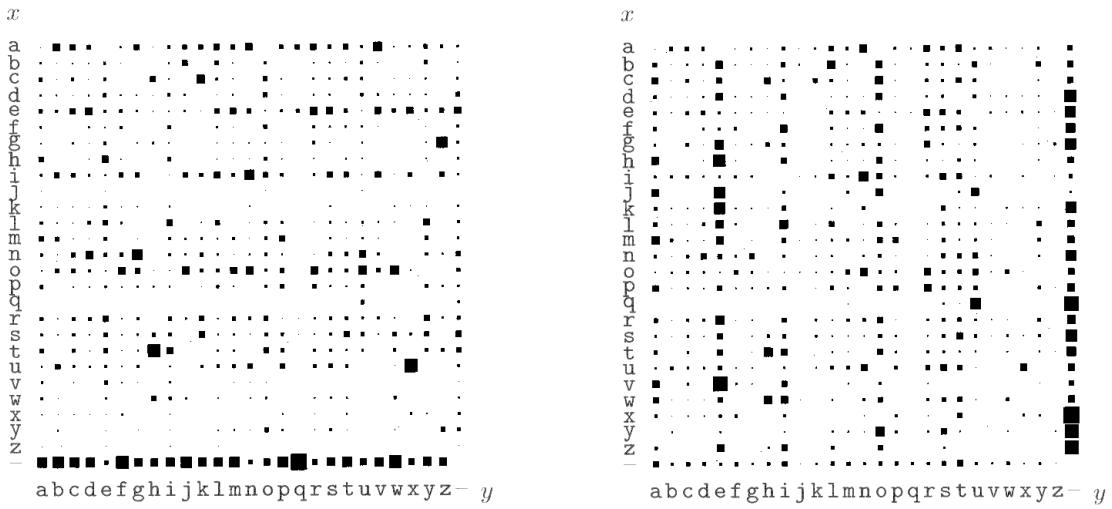
1.2 Der Satz von Bayes

Der Satz von Bayes ist eine direkte mathematische Konsequenz der Definition der bedingten Wahrscheinlichkeit.

Satz 1.5 (Bayes 1763). *Sei $P(y \mid \boldsymbol{x})$ die bedingte Wahrscheinlichkeit, den Wert y zu beobachten, wenn die beobachteten Werte $\boldsymbol{x} = (x_1, \ldots, x_n)$ sind, $P(\boldsymbol{x} \mid y)$ die bedingte Wahrscheinlichkeit, die Werte $\boldsymbol{x}$ zu beobachten, wenn der Wert y beobachtet wurde, und seien $P(\boldsymbol{x})$ und $P(y)$ die A-Priori-Wahrscheinlichkeiten, die Werte $\boldsymbol{x}$ bzw. y zu beobachten. Dann gilt*

$$P(y \mid \boldsymbol{x}) = \frac{P(y)\, P(\boldsymbol{x} \mid y)}{P(\boldsymbol{x})} \tag{1.9}$$

[5] MacKay (2003):S. 6; Wermuth und Streit (2007):S. 137.



(a) $P(x \mid y)$ (b) $P(y \mid x)$

Abbildung 1.3. Die bedingten Wahrscheinlichkeitsverteilungen (a) $P(x \mid y)$ und (b) $P(y \mid x)$ über die 27×27 möglichen Bigramme xy in einem englischen Text. Die Größe des Quadrats (i, j) entspricht der Wahrscheinlichkeit (a) $p_{ij} = P(x_i \mid y_j)$ bzw. (b) $p_{ij} = P(y_i \mid x_j)$. Modifiziert aus MacKay (2003:S. 24)

Beweis. Mit (1.8) gilt

$$P(\boldsymbol{x}, y) = P(y)P(\boldsymbol{x} \mid y) \qquad \text{und} \qquad P(\boldsymbol{x}, y) = P(\boldsymbol{x})P(y \mid \boldsymbol{x}), \tag{1.10}$$

d.h. $P(y)P(\boldsymbol{x} \mid y) = P(\boldsymbol{x})P(y \mid \boldsymbol{x})$. □

Figure 1.4 depicts the situation and illustrates the proof above: For instance, the leftmost branch in the first probability tree must equal the leftmost branch in the second one.

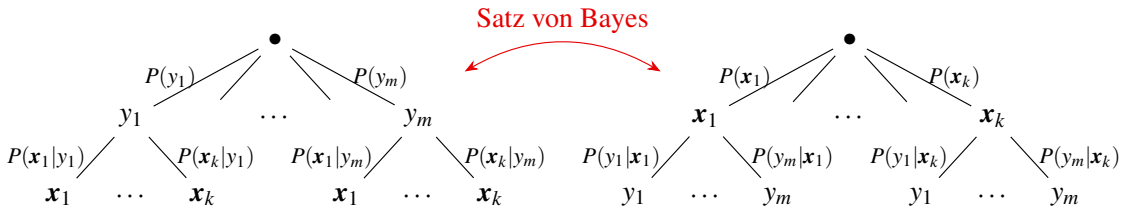


Abbildung 1.4. Wirkung des Satzes von Bayes: Änderung der Informationslage $P(y \mid \boldsymbol{x}) \leftrightarrow P(\boldsymbol{x} \mid y)$

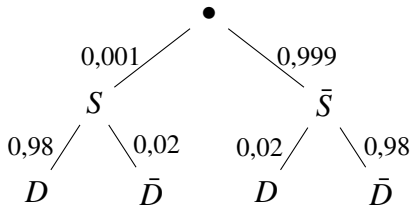
Oberflächlich betrachtet scheint das Bayes'sche Theorem nicht besonders nützlich zu sein. Mit ihm lässt sich die Einzelwahrscheinlichkeit $P(y \mid x)$ in Form von drei anderen Wahrscheinlichkeiten berechnen $P(x \mid y)$, $P(x)$, und $P(y)$. Na und? Schlimmer noch, das scheinen eher zwei Schritte zurück zu sein. Das Bayes'sche Theorem ist jedoch in der Praxis sehr nützlich, da es viele Fälle gibt, in denen wir gute Schätzungen für diese drei Wahrscheinlichkeiten haben und die vierte berechnen müssen. Oftmals beobachten wir nur die *Wirkung* einer unbekannten *Ursache*, möchten aber die Ursache bestimmen. In diesem Fall lautet der Satz von Bayes

Ursache und Wirkung

$$P(\text{Ursache} \mid \text{Wirkung}) = \frac{P(\text{Wirkung} \mid \text{Ursache})P(\text{Ursache})}{P(\text{Wirkung})}. \tag{1.11}$$

Die bedingte Wahrscheinlichkeit $P(\text{Wirkung} \mid \text{Ursache})$ quantifiziert die Beziehung in der *kausalen* Richtung, während $P(\text{Ursache} \mid \text{Wirkung})$ die *diagnostische* Richtung beschreibt, also die Richtung der Beobachtung oder Messung. Bei Problemen wie der medizinischen Diagnose haben wir normalerweise bedingte Wahrscheinlichkeiten in kausalen Beziehungen. Der Arzt kennt $P(\text{Symptome} \mid \text{Krankheit})$, aber sowohl Arzt als auch Patient suchen ja tatsächlich eine Diagnose $P(\text{Krankheit} \mid \text{Symptome})$.[6]

Beispiel 1.6 (Sie Sicht des Patienten auf eine Diagnose)**.** Es sei $P(S)$ die A-Priori-Wahrscheinlichkeit, eine gegebene Krankheit zu haben, $P(D \mid S)$ die bedingte Wahrscheinlichkeit einer positiven Diagnose, wenn der Patient die Krankheit hat, und $P(S \mid D)$ die bedingte Wahrscheinlichkeit, die Krankheit zu haben, wenn die Diagnose positiv ist.



$$\begin{aligned} P(\bar{S} \mid D) &= \frac{P(\bar{S})\,P(D \mid \bar{S})}{P(D)} \\ &= \frac{0{,}999 \cdot 0{,}02}{0{,}999 \cdot 0{,}02 + \mathbf{0{,}001} \cdot 0{,}98} \\ &= \frac{0{,}01998}{0{,}02096} = 0{,}95324 \end{aligned} \tag{1.12}$$

Entsprechend impliziert der Satz von Bayes

Beschreibung	**Bezeichnung**	**Wahrscheinlichkeit**
gesund trotz positiver Diagnose	$P(\bar{S} \mid D)$	0,95324
krank bei positiver Diagnose	$P(S \mid D)$	0,04676
kran trotz negativer Diagnose	$P(S \mid \bar{D})$	0,00002
gesund bei negativer Diagnose	$P(\bar{S} \mid \bar{D})$	0,99998

(1.13)

Eine positive Diagnose ist zu 95,3 % *falsch*! (Allerdings ist eine negative Diagnose zu 99,998 % korrekt ...) Die Ursache für dieses unerwartete Ergebnis ist die geringe Krankheitshäufigkeit („Prävalenz“) **1 ‰**. □

Definition 1.7 (**Likelihood-Funktion**)**.** Der Satz von Bayes wird oft als eine Aussage darüber interpretiert, wie gut beobachtete Daten $\boldsymbol{X}$ ein statistisches Modell mit Parametern $\boldsymbol{\theta}$ stützen:

$$P(\boldsymbol{\theta} \mid \boldsymbol{X}) = P(\boldsymbol{\theta})\,\frac{P(\boldsymbol{X} \mid \boldsymbol{\theta})}{P(\boldsymbol{X})} \tag{1.14}$$

Likelihood der Parameter

Hier ist $P(\boldsymbol{\theta})$ die A-Priori-Wahscheinlichkeit, dass die Hypothese zutrifft, und $P(\boldsymbol{\theta} \mid \boldsymbol{X})$ ihre A-Posteriori-Wahrscheinlichkeit unter der Voraussetzung, dass die Daten $\boldsymbol{X}$ beobachtet wurden. Allerdings kann $P(\boldsymbol{X} \mid \boldsymbol{\theta})$ auch als eine Funktion der Daten $\boldsymbol{X}$ und der Parameter $\boldsymbol{\theta}$ betrachtet werden. Für feste Parameter $\boldsymbol{\theta}$ ist sie eine Wahrscheinlichkeit über $\boldsymbol{X}$, aber für gegebene Daten $\boldsymbol{X}$ definiert sie die *Likelihood* der Parameter $\boldsymbol{\theta}$ bei beobachteten Daten $\boldsymbol{X}$. Die Likelihood wird mit L bezeichnet,

$$L(\boldsymbol{\theta} \mid \boldsymbol{X}) = P(\boldsymbol{\theta} \mid \boldsymbol{X}). \tag{1.15}$$

In der Folge werden wir uns in der Regel für die maximale Wahrscheinlichkeit L_* eines statistischen Modells und gegebenen Daten $\boldsymbol{X}$ interessieren, mit $L_* = L(\boldsymbol{\theta}_* \mid \boldsymbol{X})$. □

Im Prinzip beschreibt der Satz von Bayes, wie die Wahrscheinlichkeit einer Hypothese mit immer neueren Beobachtungen über die Zeit immer wieder aktualisiert wird. In der Praxis ist allerdings die Randwahrscheinlichkeit $P(\boldsymbol{X})$ sehr schwer zu bestimmen, die

[6] Russell und Norvig (2022):S. 417.

die Wahrscheinlichkeit ausdrückt, $\boldsymbol{X}$ unter allen Umständen zu beobachten. Wir werden jedoch weiter unten sehen (Abschnitt 2.4), dass sie gar nicht notwendig ist, um die A-Posteriori-Wahrscheinlichkeiten verschiedener Hypothesen bei gegebenen Beobachtungen $\boldsymbol{X}$ zu *vergleichen*.[7]

1.3 * Bedingte Unabhängigkeit

Zwei Zufallsvariablen heißen *(unbedingt) unabhängig*, wenn

$$P(x, y) = P(x)\ P(y). \tag{1.16}$$

In der Realität sind beobachtete Größen meist jedoch nicht unabhängig. Wenn es tatsächlich doch der Fall sein sollte, ist die gewählte Definition wahrscheinlich nicht relevant, so dass es meist sinnvoller ist, zwei separate Modelle zu betrachten. Ein in der Realität viel häufiger auftretendes Phänomen ist die bedingte Unabhängigkeit.[8]

Definition 1.8. *(Bedingte Unabhängigkeit)* Seien X, Y, Z Zufallsvariablen mit Ergebnissen x, y, z. Dann heißen X und Y *bedingt unabhängig* gegeben Z genau dann, wenn $P(z) > 0$ und

$$P(x \mid y, z) = P(x \mid z). \tag{1.17}$$

Manchmal wird diese Eigenschaft $(X \perp Y \mid Z)$ geschrieben. □

Lemma 1.9. *Seien X, Y, Z Zufallsvariablen mit Ergebnissen x, y, z. Dann sind X und Y genau dann bedingt unabhängig gegeben Z, wenn*

$$P(x, y \mid z) = P(x \mid z)\ P(y \mid z). \tag{1.18}$$

Hier ist $P(x, y \mid z)$ die gemeinsame Wahrscheinlichkeit von X und Y gegeben Z. Diese alternative Formulierung besagt, dass X und Y unabhängige Zufallsvariablen sind, wenn Z gegeben ist.

Beweis. Mit Gleichung (1.18) können wir die folgenden Umformungen berechnen,

$$\begin{aligned}
P(x, y \mid z) = P(x \mid z)\ P(y \mid z) \quad &\overset{(1.8)}{\iff} \quad \frac{P(x, y, z)}{P(z)} = \frac{P(x, z)}{P(z)}\ \frac{P(y, z)}{P(z)} \\
&\iff \quad P(x, y, z) = \frac{P(x, z) P(y, z)}{P(z)} \\
&\iff \quad \frac{P(x, y, z)}{P(y, z)} = \frac{P(x, z)}{P(z)} \\
&\overset{(1.8)}{\iff} \quad P(x \mid y, z) = P(x \mid z),
\end{aligned}$$

indem wir Definition (1.8) zweimal anwenden. Die letzte Gleichung ist genau die Definition (1.17), das heißt sie ist äquivalent zu Gleichung (1.18). □

Stellen wir nun die bedingte Abhängigkeit zweier Zufallsvariablen durch einen Pfeil dar, der den Einfluss oder die kausale Beziehung ausdrückt. Sind zwei Zufallsvariablen bedingt unabhängig, so werden sie nicht durch einen Pfeil verbunden. Grundsätzlich gibt es dann nur zwei Möglichkeiten, die bedingte Abhängigkeit zweier Zufallsvariablen X und Y gegeben Z auszudrücken, wie in Abbildung 1.5 dargestellt. Bedingte Unabhängigkeit wird

Bayes-Netz

[7] Downey (2011):S. 75.

[8] Pourret et al. (2008):S. 9; Wermuth und Streit (2007):S. 152.

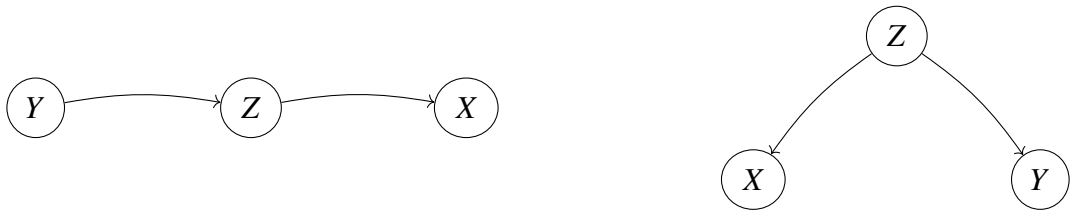


Abbildung 1.5. Die beiden grundlegenden Beziehungen, die eine bedingte Unabhängigkeit zweier Zufallsvariablen X und Y gegeben Z darstellen.

demnach entweder durch zwischenliegende („die direkte Verbindung unterbrechende“) Zufallsvariablen induziert, oder durch eine einzelne Zufallsvariable, die andere beeinflusst. Erweitert man dieses Vorgehen auf mehrere Zufallsvariablen, so entsteht ein ein gerichteter Graph („Digraph“), ein sogenanntes *Bayes-Netz*.

Beispiel 1.10. *(Der Lkw-Fahrer)*[9] Ein Lkw-Fahrer hat eine Fahrt von 600 Meilen zu absolvieren. Um das Risiko X: „er schläft während der Fahrt ein“ zu analysieren, betrachten wir, ob Y: „er in der Nacht zuvor gut geschlafen hat“ und Z: „er zu Beginn der Fahrt müde ist“. Offensichtlich gibt es kausale Zusammenhänge zwischen dem Schlaf des Fahrers, seiner empfundenen Müdigkeit und dem Risiko des Einschlafens. Daher können die drei (binären) Zufallsvariablen nicht unabhängig sein. Nehmen wir nun an wir wissen, dass der

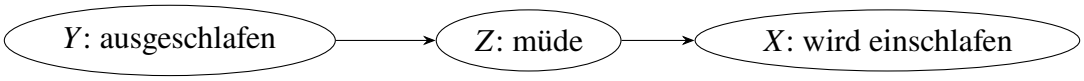


Abbildung 1.6. Die gegenseitige Beeinflussung der drei Zufallsvariablen in Beispiel 1.10. Variable X ist bedingt unabhängig von Y gegeben Z.

Lkw-Fahrer zu Beginn der Fahrt müde ist. Dann spielt es für die Bewertung des Risikos keine Rolle, ob dies auf einen schlechten Schlaf in der Nacht zuvor oder auf einen anderen Grund zurückzuführen ist. Fühlt sich der Lkw-Fahrer zu Beginn der Fahrt nicht müde, kann man davon ausgehen, dass die Qualität seines Schlafs keinen Einfluss auf das Risiko hat. Somit ist das Risiko des Einschlafens (X) bedingt unabhängig von der Qualität des Schlafs (Y), wenn der Lkw-Fahrer müde ist (Z). In Bezug auf die Wahrscheinlichkeiten der jeweiligen Ergebnisse x, y, z gilt daher

$$P(x \mid y, z) = P(x \mid z). \tag{1.19}$$

Die Kenntnis der Werte von y und z ist also nicht wertvoller als nur die Kenntnis des Wertes von z. Es ist nutzlos, das Verhalten von X, Y, Z durch eine Funktion von drei Variablen zu beschreiben, stattdessen können wir aus (1.19) ableiten, dass

$$P(x, y, z) = P(y)\ P(z \mid y)\ P(x \mid z). \tag{1.20}$$

So kann das Risikomodell aufgebaut werden, indem nacheinander die Qualität des Schlafs, dann sein Einfluss auf die Müdigkeit und schließlich der Einfluss der Müdigkeit auf das Risiko des Einschlafens untersucht wird (Abbildung 1.6). □

Beispiel 1.11. *(Der gedopte Athlet)*[10] Während eines Sportwettkampfs wird jeder Athlet zwei Dopingkontrollen unterzogen, um festzustellen, ob er eine bestimmte verbotene

[9] Pourret et al. (2008):S. 9.
[10] Pourret et al. (2008):S. 10.

Substanz eingenommen hat: ein Bluttest und ein Urintest. Beide Tests werden in zwei verschiedenen Labors durchgeführt, ohne dass es zu einer Absprache kommt.

Natürlich sind die beiden Tests nicht unabhängig voneinander. Fällt der Bluttest positiv aus, so ist der Sportler wahrscheinlich gedopt, und der Urintest wird wahrscheinlich ebenfalls positiv ausfallen. Ist der Sportler jedoch gedopt, können beide Tests als unabhängig

Abbildung 1.7. Die Beeinflussungen der drei Zufallsvariablen in Beispiel 1.11. Variable X ist bedingt unabhängig von Y gegeben Z.

angesehen werden, da sie unterschiedliche Methoden verwenden. Das Gleiche gilt, wenn der Sportler nicht gedopt ist: Die beiden Tests sind also bedingt unabhängig, wenn man den Zustand des Sportlers schon kennt. Wenn z das Ergebnis der binären Zufallsvariablen Z ist, ob der Sportler gedopt ist oder nicht, x das Ergebnis des Bluttests und y das Ergebnis des Urintests, so können wir formal schreiben

$$P(x \mid y, z) = P(x \mid z), \qquad \text{und symmetrisch} \qquad P(y \mid x, z) = P(y \mid z) \tag{1.21}$$

Beide Gleichungen besagen, dass die Information, ob der Sportler die Substanz eingenommen hat, ausreicht, um die Wahrscheinlichkeit eines positiven Testergebnisses zu bestimmen, vgl. Abbildung 1.7. Es ist nutzlos, das Verhalten von X, Y, Z durch eine Funktion von drei Variablen zu beschreiben. Stattdessen ergeben die Gleichungen (1.21):

$$P(x, y, z) = P(z)\ P(x \mid z)\ P(y \mid z). \tag{1.22}$$

Dies bedeutet, dass Überlegungen zum Anteil der gedopten Athleten $P(z)$ und zu den Zuverlässigkeiten $P(x \mid z)$, $P(y \mid z)$ der beiden Tests ausreichen, um ein Modell zu erstellen. □

1.4 Übungsaufgabe

Aufgabe 1.1 (**Random variables**). (a) Wir haben den Begriff der Zufallsvariablen auf Seite 7 oben etwas ungenau eingeführt. Wie lautet die korrekte formale Definition einer Zufallsvariablen X, wenn wir den Stichprobenraum mit Ω bezeichnen?

- □ $X : \Omega \to \Omega$
- □ $X : \Omega \to \mathbb{R}$
- □ $X : \mathbb{R} \to \Omega$
- □ $\mathbb{R} : X \to \mathbb{R}$

(b) Sei $\Omega = \{1, 2, 3, 4, 5, 6\}$ der Stichprobenraum eines Würfelwurfs. Wie lautet die Wertetabelle der Zufallsvariablen X, die den Wert 1 ergibt, wenn eine gerade Anzahl von Punkten gewürfelt wird, und ansonsten 0?

Augenzahl ω	1	2	3	4	5	6
$X(\omega)$						

Wie groß ist die Wahrscheinlichkeit $P(X = 1)$, dass die Zufallsvariable den Wert 1 annimmt? Welche implizite Annahme wird bei der Berechnung dieser Wahrscheinlichkeit getroffen?

2
Modelle und Theorien

Kapitelübersicht

2.1 Methoden des logischen Schließens 18
2.2 Theorien 19
2.3 Was eigentlich ist ein Modell? 23
2.4 Ockhams Rasiermesser und Modellauswahl 27
2.5 Übungsaufgaben 31

> Erst die Theorie entscheidet, was beobachtet werden kann.
>
> Einstein zu Heisenberg (Quelle: von Weizsäcker, 1985:S. 331)

Ein grundlegende Begriff des maschinellen Lernen und der Datenanalyse ist der des Modells. Die Wahl eines Modells entscheidet darüber, wie die beobachteten Daten analysiert und interpretiert werden können. Der Wort „Modell“ in dem Sinne, wie wir es heute verwenden, taucht erst im 16. oder 17 Jahrhundert auf.[1] und wurde in den Wissenschaften erst seit Entstehung der Quantentheorie zu Beginn des 20. Jahrhunderts verwendet. Die großen Schwierigkeiten, die Quantenphänomene zu verstehen, veränderten damals die Sicht auf das, was bis dahin als „Theorie“ bezeichnet wurde, indem sie den kategorischen Anspruch einer Theorie, die Wahrheit darzustellen, relativierten. Insbesondere bildeten sich so im Laufe der Zeit verschiedene Modelle des Atoms heraus, wobei jedes einzelne sich jeweils auf einen bestimmten Geltungsbereich beschränkte, d.h. auf einen begrenzten Ausschnitt der Realität, der damit erklärt werden konnte. Dieser Ansatz war so erfolgreich, dass wir noch heute das Bohr'sche Modell zum Verständnis der Spektrallinien des Wasserstoffatoms und das Orbitalmodell zur Erklärung komplexerer Atome und Moleküle verwenden.

Ziel dieses Kapitels ist es, die Zusammenhänge zwischen den Begriffen Theorie und Modell aus moderner Sicht zu verdeutlichen und sie mit Beobachtungen von Phänomenen der realen Welt und mit der Aufstellung von Hypothesen in Beziehung zu setzen. Alle diese Begriffe und ihre Verhältnisse zueinander beruhen auf den Methoden des logischen Schließens. Wir beginnen mit einem kurzen Überblick über diese Methoden und fahren dann mit den Definitionen von Theorien und Modellen fort. Das Kapitel endet mit der

[1] von Latein *modulus* – „Maß, Maßstab“, `https://en.wiktionary.org/wiki/model`, `https://de.wiktionary.org/wiki/Modell`

Einführung des Ockham'schen Rasiermessers, einem Prinzip zur Auswahl eines optimalen Modells aus einer Sammlung von mehreren Modellen für ein bestimmtes Problem.

2.1 Methoden des logischen Schließens

Es gibt verschiedene Formen des logischen Schließens. Im antiken Griechenland unterschied Aristoteles zwei Formen, die Deduktion und die Induktion. Doch seit den Arbeiten des Logikers und Philosophen Charles Sanders Peirce zu Beginn des 20. Jahrhunderts wird die Induktion von einer dritten Form, der Abduktion, abgegrenzt.[2] Der Begriff wird in der Wissenschaftstheorie heute jedoch etwas abgewandelt zu Peirce ursprünglicher Definition benutzt.

Deduktion

Deduktion ist die Schlussfolgerung aus einer oder mehreren Aussagen, den Prämissen, auf einer logisch zwingende Konsequenz. Das deduktive Denken verknüpft also Prämissen mit Konklusionen: Wenn alle Prämissen wahr sind, die Begriffe klar sind und die Regeln der deduktiven Logik befolgt werden, dann ist die Konklusion zwingend wahr. Anders ausgedrückt schließt eine Deduktion aus gegebenen Voraussetzungen auf einen speziellen Fall.

Induktion

Deduktion steht im Gegensatz zur Induktion. *Induktion* ist eine Methode der Argumentation, bei der die Prämissen als ein plausibler Beleg für die Wahrheit der Schlussfolgerung angesehen werden, ohne aber logisch zwingend zu sein. Induktion kann als eine Methode beschrieben werden, bei der aus Erfahrungen und Beobachtungen eine allgemeine Regel, oder ein Gesetz, abgeleitet wird. Beim deduktiven Schließem wird eine Schlussfolgerung

Deduktion	**Induktion**	**Abduktion**
Regel: $A \Rightarrow B$	*Prämisse:* A	*Regel:* $A \Rightarrow B$
Prämisse: A	*Konklusion:* B	*Konklusion:* B
Konklusion: B	*Regel:* $A \Rightarrow B$	*Hypothese* H_1*:* A_1
		$\vdots$ $\quad$ $\vdots$
		Hypothese H_n*:* A_n
		Beste Prämisse H_**:* A

Tabelle 2.1. Prinzip von Deduktion, Induktion und Abduktion.

durch die Reduktion allgemeiner Regeln erreicht, wobei der betrachtete Bereich so weit eingeschränkt wird, bis nur noch die Schlussfolgerung übrig bleibt. Beim deduktiven Schließen gibt es keine Unsicherheit. Beim induktiven Schließen wird die Schlussfolgerung durch Verallgemeinerung oder Extrapolation von spezifischen Fällen auf allgemeine Regeln erreicht, was zu einer Konklusion führt, die erkenntnistheoretisch gesehen unsicher ist.

Es gibt eine dritte Form der logischen Schlussfolgerung, die „Abduktion". *Abduktion* ist eine Methode der Argumentation, um aus Beobachtungen verschiedene Hypothesen zu gewinnen und die einfachste oder wahrscheinlichste davon auszuwählen. Eine *Hypothese* ist eine unbewiesene, aber plausible Annahme oder ein Erklärungsvorschlag für ein Phänomen. Sie dient oft als *Arbeitshypothese*, die vorläufig als Grundlage für weitere Forschung

Hypothese

[2] Die Idee der Abduktion geht strenggenommen auf Aristoteles zurück, der sie mit dem Begriff *Apagoge* erwähnt (Erste Analytik II, 25, 69a, `https://logicalstudy.ihcs.ac.ir/article_5171.html`)) und auch bereits der Induktion gegenüberstellt. Die Übersetzung des Begriffs *Apagoge* mit *Abduktion* erfolgte 1597 erstmals durch den italienischen Rechtsgelehrten Julius Pacius. Peirce hat den Ausdruck also zwar nur wieder aufgegriffen, aber präziser definiert und so als wissenschaftlichen Erkenntnisprozess für die Wissenschaftstheorie und die Informatik nutzbar gemacht.

akzeptiert wird. Eine wissenschaftliche Hypothese muss vor allem durch Beobachtungen oder Experimente überprüfbar sein, d.h. verifizierbar oder falsifizierbar sein. Abduktives Schließen kann also als Rückschließen auf die beste Erklärung für ein beobachtetes Phänomen verstanden werden. Im Gegensatz zum deduktiven Schließen bleibt bei abduktiven Schlussfolgerungen ein Rest von Unsicherheit oder Zweifel. Im 20. Jahrhundert wurde die Abduktion weitgehend ignoriert, doch mit der wachsenden Rechenleistung wurde sie seit den 1990er Jahren in den Bereichen Recht, Informatik und künstliche Intelligenzforschung wiederentdeckt.[3]

Abduktion

Die Beziehungen zwischen Deduktion und Abduktion als Methoden der Inferenz sind wie folgt. Beide gehen von der Regel aus, nehmen aber entweder das Ergebnis oder die Prämisse an. Die Deduktion schließt auf das Ergebnis, während die Abduktion die beste Hypothese als Prämisse auswählt. Darüber hinaus gehen Induktion und Abduktion gleichzeitig von der Schlussfolgerung aus, wechseln aber die Rollen von Regeln und Prämisse als Ergebnis ihrer Schlussfolgerung.

Beispiel 2.1. Das folgende Beispiel ist leicht modifiziert eines, dass Peirce 1902 veröffentlichte.[4] Hier werden den Begriffen in Tabelle 2.1 ie folgenden Aussagen zugeordnet:

Regel $A \Rightarrow B$*:*	„Alle Bohnen aus dieser Tüte sind weiß.“
Prämisse A:	„Diese Bohnen sind aus dieser Tüte.“
Conclusion B:	„Diese Bohnen sind weiß.“

Für ie Abduktion haben wir formal die zwei Hypothesen H_1: A und H_2: $\neg A$, von denen wir die wahrscheinlichste auswählen. Sowohl die induktiven als auch die abduktiven

Deduktion	**Induktion**	**Abduktion**
Alle Bohnen aus dieser Tüte sind weiß Diese Bohnen sind aus dieser Tüte Diese Bohnen sind weiß.	TDiese Bohnen sind aus dieser Tüte Diese Bohnen sind weiß Alle Bohnen aus dieser Tüte sind weiß	Alle Bohnen aus dieser Tüte sind weiß Diese Bohnen sind weiß H_1: Diese Bohnen sind aus dieser Tüte H_2: Diese Bohnen sind nicht aus dieser Tüte H_1: Diese Bohnen sind aus dieser Tüte

Tabelle 2.2. Beispiele für Deduktion, Induktion und Abduktion.

Schlussfolgerungen können falsch sein. Nur die Deduktion ist immer logisch korrekt. □

2.2 Theorien

> Wie ist Theorie möglich? Sie folgt niemals mit logischer Notwendigkeit aus der Erfahrung. Aus Gesetzen, die sich in der Vergangenheit bewährt haben, folgt nicht mit logischer Notwendigkeit, was in Zukunft geschehen wird.
>
> Carl Friedrich von Weizsäcker (1985:S. 24)

Eine Theorie ist eine durch rationales Nachdenken gewonnene Erkenntnis über ein Phänomen der Realität.[5] *Realität* bedeutet hier eine konsistente Welt außerhalb und unabhängig

[3] Flach und Kakas (2000).

[4] https://books.google.com/books?id=E7fnCAAAQBAJ&pg=PA13&redir_esc=y#v=onepage&q&f=false

[5] Im Altgriechischen bedeutete $\theta\varepsilon\omega\rho\acute{\iota}\alpha$ (theoria – „beobachten, betrachten, [an]schauen“) die Betrachtung der Wahrheit durch reines Denken, unabhängig von ihrer Realisierung.

von dem Beobachter.[6] In der modernen Wissenschaft ist eine *Theorie* eine Erklärung der Natur, einzelner Aspekte der natürlichen Welt oder eines Phänomens er realen Welt; sie muss verifizierbar gemäß der wissenschaftlichen Methode, die allgemein akzeptierte Protokolle von Beobachtungen, Messungen and Ergebnisberwertungen anwendet.[7] Eine Theorie kann bestimmte Modelle für Phänomene der realen Welt entwerfen.

Beispiel 2.2. *(Quantentheorie)* Eines der spektakulärsten Beispiele für eine Theorie mit umstrittenem Bezug zur Realität ist die Quantentheorie. Ihr Ziel ist es, die Dynamik von Elementarteilchen und Atomen, d. h. die mikroskopische Welt, zu erklären. Sie lässt sich durch Diracs fundamentales Überlagerungsprinzip in einem abstrakten Hilbert-Raum formulieren, wobei die Schrödinger-Gleichung die Dynamik der Vektoren des Hilbert-Raums beschreibt, die die möglichen Quantenzustände repräsentieren.[8] Die Anwendung der Theorie auf lokale Teilchen führt jedoch zu fundamentalen Paradoxien, von denen das berühmteste das EPR-Paradox ist.[9] Dieses kann nur gelöst werden, indem man mindestens eines der drei Konzepte Lokalität, Realität oder den freien Willen aufgibt, die normalerweise für eine physikalische Theorie vorausgesetzt werden:[10]

- *Keine Lokalität:* Lokalität ist das Prinzip, dass sich Wechelwirkungen zwischen Teilchen nicht schneller als das Licht ausbreiten können. Nach der allgemeinen Relativitätstheorie ist dies Voraussetzung für die Kausalität, also dafür, dass eine Ursache immer zeitlich vor ihrer Wirkung liegt. Eine Auflösung des EPR-Paradoxons wäre jedoch die Annahme, dass sich Wechselwirkungen zwischen Teilchen „instantan“, also augenblicklich un damit schneller als das Licht ausbreiten.

- *Keine Realität: Realität* bedeutet eine konsistente Welt außerhalb des Beobachters und unabhängig von diesem. Eine Lösung des EPR-Paradoxons besteht jedoch darin, anzunehmen dass mikroskopische Teilchen keine Eigenschaften haben, die unabhängig vom Beobachter gemessen werden können. Dies ist der Standpunkt des Modells der „Kopenhagener Deutung“, das von Bohr und Heisenberg in den 1920er Jahren entwickelt wurde.

- *Kein freier Wille:* An observer cannot configure a measurement apparatus independently from the objects to be observed. Any experiment is correlated to an unknown cause in the past or to some mysterious “conspiracy” due to the quantum states of the objects. Ein Beobachter kann einen Messapparat nicht unabhängig von den zu beobachtenden Objekten konfigurieren, jedes Experiment ist mit einer unbekannten Ursache in der Vergangenheit oder auf eine mysteriöse „konspirative Weise“ immer mit den Quantenzuständen der Objektpaare verknüpft.

Jede dieser Optionen hat ernsthafte Auswirkungen auf die grundsätzliche Gültigkeit der physikalischen Theorien im Allgemeinen. Was ist falsch? Ein Ausweg aus diesem Dilemma besteht darin, das zugrunde liegende Konzept der Teilchen zu hinterfragen: Es gibt überhaupt kein Paradoxon, wenn das Superpositionsprinzip und die nichtlokale Schrödinger-Wellenfunktion als real akzeptiert werden, nicht aber Teilchen.[11] Nach diesem Modell

[6] cf. Zeh (2012):S. 50.

[7] Für weitere Details vgl. `https://plato.stanford.edu/archives/sum2021/entries/scientific-method/`

[8] Zeh (2012):S. 52.

[9] Zeh (2012):S. 15.

[10] Zeh (2012):S. 16.

[11] “I think that it has to be the wavefunction [...] that describes quantum reality.” (Penrose, 2004:S. 508). “We have to think of the entire wave as describing (or ‘being’) just a single particle. [...] We must think

bzw. dieser „Interpretation“ sind Teilchen Realisierungen durch Messungen des nichtlokalen Quantenfeldes ψ, ähnlich dem (weithin akzeptierten) Modell, dass Elektronen Realisierungen des elektromagnetischen Feldes sind.[12] Born war der erste, der die komplexwertige Wellenfunktion ψ als "Wahrscheinlichkeitswelleßo interpretierte, dass die Wahrscheinlichkeit $P(x,t)$, ein Quantenteilchen am Punkt x zur Zeit t zu messen, durch ihr Modulusquadrat gegeben ist:[13]

$$P(x,t) = |\psi(x,t)|^2. \tag{2.1}$$

In diesem Modell ist der Impuls p durch den Term $i\hbar\frac{\partial}{\partial x}\psi(x,t)$ gegeben, womit die Heisenberg'sche Unschärferelation $\Delta x \Delta p \geqq \hbar$ nurmehr eine rein mathematische Konsequenz des Fourier-Theorems ist. Diese Ideen gehen insbesondere auf de Broglie, Schrödinger, Born, Bohm, Everett und Zeh zurück.[14] So klar und konsistent die Quantentheorie auch sein mag, bis heute gibt es keine allgemein akzeptierte Ontologie für sie. *„It is a common view among many of today's physicists that quantum mechanics provides us with* no *picture of 'reality' at all!“*[15] Der Fall bleibt ungelöst. □

> I do not believe that we have yet found the true 'road to reality', despite the extraordinary progress that has been made over two and one half millenia.
>
> Roger Penrose in *The Road to Reality* Penrose (2004:S. 1027)

Paradigmen und Revolutionen

Nach Heisenberg (1948) wird eine Theorie als *abgeschlossen*, wenn sie nicht durch geringfügige Änderungen verbessert werden kann. Eine neue Theorie, die eine abgeschlossene Theorie verbessert, unterscheidet sich in bestimmten grundlegenden Begriffen radikal von ihr, wobei der Umfang der erfolgreichen Teile der alten Theorie erhalten bleibt. Kuhn (1962) sieht die Entwicklung solcher Theorien in der Wissenschaft durch ständig wechselnde Paradigmen bestimmt. Solche Paradigmenwechsel nennt er „Revolutionen.“[16]

Beispiel 2.3. *(Die Kopernikanische Revolution)* Ein berühmtes Beispiel für eine Revolution im wissenschaftlichen Denken ist die kopernikanische Revolution. In der Denkschule des Ptolemäus wurden Zyklen und Epizyklen (mit einigen zusätzlichen Konzepten) verwendet, um die Bewegungen der Planeten in einem Weltall zu modellieren, in dessen Zentrum eine stationäre Erde steht. Als die Genauigkeit der Himmelsbeobachtungen zunahm,

---

of a wavefunction as one entire thing. If it causes a spot at one place [by a measurement], then it has done its job.” (Penrose, 2004:S. 512). “Die Realität neuer physikalischer Konzepte war häufig anfangs umstritten. Galilei wurde angeklagt, weil er das kopernikanische Weltbild als real und nicht nur als eine Rechenmethode ansah. [...] Im neunzehnten Jahrhundert wurde auch das elektrische Feld zunächst als ein rein formales Hilfskonstrukt zur Berechnung von Kräften auf Ladungen angesehen. Da wir das Feld nicht »direkt erblicken« können, fragt sich, was wir unter seiner Realität verstehen. Bei der Begründung eines realen elektrischen Feldes spielt die konsistente »Denkbarkeit« kleiner Probeladungen, mit deren Hilfe man es überall operationell nachweisen *könnte*, ohne es merklich zu stören, eine wesentliche Rolle.” [...] Ein weiteres Objekt von strittiger Realität in der Physikgeschichte ist der Lichtäther als ursprünglich vermutetes materielles Medium für elektromagnetische Schwingungen. Er gilt gemeinhin als gescheitertes Konstrukt [...], da er nicht zu mit den Experimenten konsistenten Vorhersagen geführt hat. (Zeh, 2012:S. 48–49)

[12] Zeh (2012):S. 52.

[13] Goswami (1997):S. 13; Scheck (2013):S. 39.

[14] von Weizsäcker (1985):S. 492–499; Zeh (2012):S. 60; vgl. auch Penrose (2004):S. 507.

[15] Penrose (2004):782: „Es ist eine verbreitete Ansicht unter vielen der heutigen Physiker*innen, dass die Quantenmechanik uns überhaupt kein Bild von der , Realität‘ liefert!“

[16] Vgl. auch von Weizsäcker (1985):S. 219.

musste auch die Komplexität der ptolemäischen epizyklischen Mechanismen zunehmen, um die berechneten Planetenpositionen mit den beobachteten Positionen in Übereinstimmung zu bringen. Kopernikus schlug stattdessen eine Kosmologie vor, in der die Sonne im Mittelpunkt stand und die Erde einer der Planeten war, die sich um sie drehten. Beide

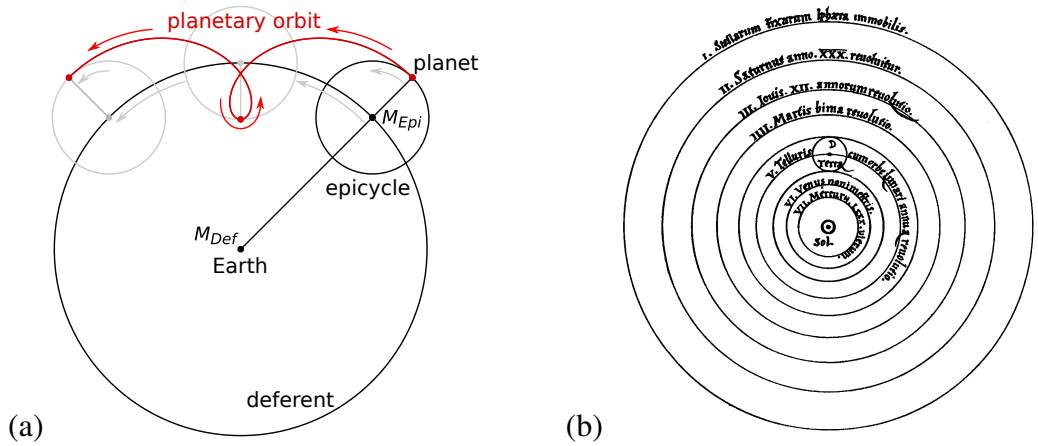


Abbildung 2.1. The epicycle theory and Copernicus cosmology. Modified from: Koyré (1992:S. 53)

Theorien sind in der Tat mathematisch äquivalent. Aber während das ptolemäische Modell als Parameter zwei Radien pro Planet erfordert (den Radius des Deferenten und den Radius des Epizykels), benötigt die kopernikanische Theorie nur einen Radius pro Planet (den Radius seiner Umlaufbahn). Daher ist die kopernikanische Theorie eleganter.

Dennoch wurde die Kopernikus' Theorie zunächst abgelehnt. Ein schwerwiegender Einwand war, dass eine Bewegung der Erde in Bezug auf die „Himmelssphäre", zu der die Sterne gehören sollten, beobachtbar sein müsste. Da solche Relativbewegungen der Sterne aber nicht beobachtet wurden, war klar, dass sie vernachlässigbar klein sein mussten, also Lichtjahre entfernt – eine Entfernung, die damals als unmöglich galt. Außerdem passten die kopernikanischen Kreisbahnen nicht zu den immer genauer werdenden Beobachtungsdaten der Planetenbewegungen, ein Fehler, den erst Kepler eineinhalb Jahrhunderte später durch die Annahme *elliptischer* Planetenbahnen beheben konnte. Es war am Ende Newton, der die Revolution mit seiner Gravitationstheorie vollendete, indem er die kopernikanisch-keplersche heliozentrische Kosmologie mit Galileis Vermutung über die Trägheit der Masse vereinte und mit dem aristotelischen Paradigma brach, dass Massen immer dazu neigen, zur Ruhe zu kommen.[17] □

[17]"Aristotle said what daily experience teaches us and gave a plausible explanation. Example 'rolling ball': *A moving body has the tendency to take up the state of rest. The movement therefore comes to a halt by itself; if it is to be maintained, the constant action of a force is required.* Example 'falling body': *Heavy bodies fall faster than light ones.* Aristotle describes what he observes. He gives a plausible, but false, explanation." Mein ehemaliger Kollege an der Fachhochschule in Hagen, Dieter Bangert, sagte in einem Vortrag: „Aristoteles sagte, was uns die tägliche Erfahrung lehrt und gab dafür eine plausible Erklärung. Beispiel ‚rollende Kugel': Ein bewegter Körper hat das Bestreben, den Zustand der Ruhe einzunehmen. Die Bewegung kommt daher von selbst zum Erliegen. Soll sie aufrechterhalten werden, so ist die ständige Einwirkung einer Kraft erforderlich. Beispiel ‚Fallender Körper': Schwere Körper fallen schneller als leichte. Aristoteles beschreibt, was er beobachtet. Er gibt dafür eine zwar plausible, aber falsche Erklärung." (`http://haegar.fh-swf.de/spielwiese/ART/2_Bangert_Modelle-und-Wirklichkeit.pdf`)

2.3 Was eigentlich ist ein Modell?

Die Welt um uns herum ist komplex. Um sie zu verstehen und mit ihr umzugehen, verwenden wir – trotz unserer Einschränkungen und Neigungen – Darstellungen der Realität, die wir Modelle nennen. Im Allgemeinen ist ein *Modell* eine informative Darstellung eines Systems oder eines Objekts.[18] Modelle lassen sich grob unterteilen in konkrete Modelle (z. B. Prototypen in der Technik, maßstabsgetreue Modelle wie Architekturmodelle oder Modelleisenbahnen) und abstrakte Modelle (z. B. konzeptionelle Modelle, oft in mathematischer oder schematischer Form). Begriffliche Modelle sind für die Wissenschaftsphilosophie von zentraler Bedeutung, da fast jede wissenschaftliche Theorie heutzutage eine Art von Modell der physikalischen oder menschlichen Sphäre enthält.

Modelle im Allgemeinen

Beispiel 2.4. *(Atommodelle)* Im Laufe der Geschichte der Naturwissenschaften haben sich verschiedene Modelle über die Struktur der Materie herausgebildet. Im antiken Griechenland stellten die Philosophen Leukipp und Demokrit die Theorie auf, dass die natürliche Welt aus Atomen – d.h. unteilbaren Teilchen – und der Leere besteht.[19] Doch während sie ihre Theorie als real ansahen, haben die im 20. Jahrhundert entwickelten Atommodelle einen genau definierten Geltungsbereich, der nur begrenzte Aspekte der Realität abbildet. So kann das Bohr'sche Modell, das davon ausgeht, dass sich die Elektronen auf diskreten stabilen Bahnen mit Drehimpulsen $n\hbar$, $n = 1, 2, \ldots$, um den zentralen Kern drehen, gut die Spektrallinien des Wasserstoffatoms erklären, bei komplexeren Atomen jedoch versagt sie. Zur Erklärung der Anordnung mehrerer Elektronen um den Kern kann das auf der Schrödinger'sche Wellenfunktion basierende Orbitalmodell verwendet werden, dieses kann jedoch nicht zur Berechnung von Molekülorbits eingesetzt werden. Das wiederum ist der Anwendungsbereich des Molekülorbitalmodells, bei dem die Elektronen nicht den einzelnen chemischen Bindungen zwischen den Atomen zugeordnet werden, sondern sich unter dem Einfluss der Atomkerne im gesamten Molekül bewegen. □

Bemerkung 2.5. Sehr wichtige Modelle beim maschinellen Lernen sind statistische Modelle. Dabei handelt es sich um spezielle Formen mathematischer Modelle, die aus funktionalen Gleichungen bestehen und Variablen von Modellparametern unterscheiden. Wir werden diese Modelle weiter unten in Abschnitt 3.3 untersuchen. □

Um die komplexen Zusammenhänge zwischen Theorien, Modellen und Beobachtungen zu beschreiben, ist der Begriff der Hypothese wichtig. Normalerweise wird eine Hypothese durch Abduktion aus Beobachtungen erstellt. Sie kann ein Modell spezifizieren, das ein Phänomen der realen Welt erklärt oder darstellt. Eine Hypothese kann auch Teil einer Theorie sein, die dann ihrerseits durch Beobachtungen überprüft werden kann.

Die komplexen Beziehungen zwischen Realität, Theorie, Modellen, Hypothesen und Beobachtungen sind in Abbildung 2.2 dargestellt. Wie wir sehen, basieren sie hauptsächlich auf den Methoden des logischen Schließens, nämlich Deduktion, Induktion und Abduktion. Insbesondere ist ein *deduktives Modell* eine logische Struktur, die auf einer Theorie beruht, während ein *induktives Modell* sich aus empirischen Erkenntnissen und deren Verallgemeinerung ergibt.

Deduktive und induktive Modelle

[18] Der Begriff „Modell" bezeichnete im späten 16. Jahrhundert im Englischen ursprünglich die Pläne eines Gebäudes und wurde über das Französische und Italienische letztlich vom lateinischen *modulus*, einem Maß, abgeleitet.

[19] https://plato.stanford.edu/entries/atomism-ancient/#LeucDemo

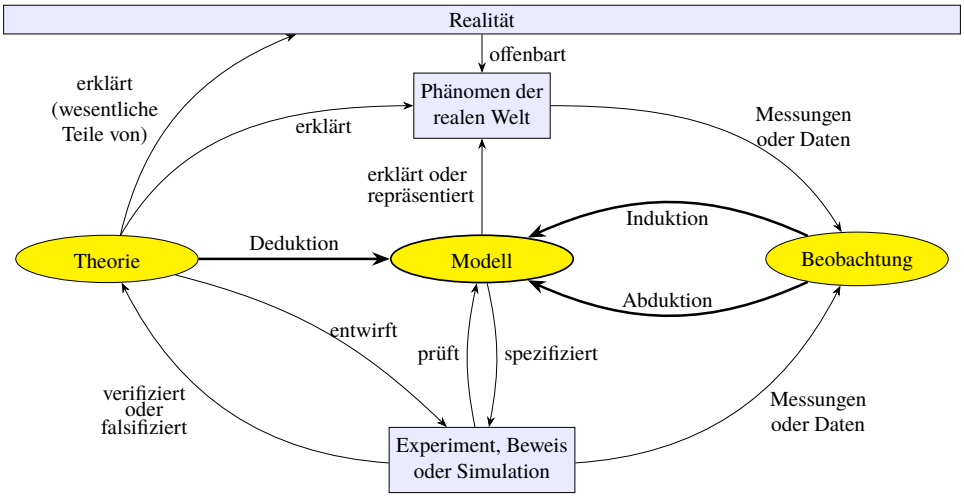


Abbildung 2.2. A model as a representation of reality and its relationships to theory, hypotheses, and observation.

Mathematische Modelle

Sehr wichtige Modelle im Zusammenhang mit maschinellem Lernen sind statistische Modelle. Sie sind spezielle Formen mathematischer Modelle, die aus Funktionsgleichungen bestehen und Variablen von Modellparametern unterscheiden.

mathematische Modelle

Definition 2.6. Ein *mathematisches Modell* ist eine Darstellung eines Systems, d.h. einer Menge von zusammenhängenden Objekten der realen Welt, die durch eine funktionale Gleichung ausgedrückt wird

$$y = f(x, \theta) \tag{2.2}$$

aus einigen Modellvariablen y und x, einem oder mehreren Modellparametern θ und einer Funktion f. Normalerweise sind $y \in \mathbb{R}$, $x \in \mathbb{R}^n$, $\theta \in \mathbb{R}^k$ und $f : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}$, für gegebene Zahlen $n, k \in \mathbb{N}$. □

Beispiel 2.7. *(Atommodelle, Fortsetzung)* Die atomaren Modelle in Beispiel 2.4 sind allesamt mathematische Modelle, jedes mit bestimmten Parametern und Variablen. Der Parameter des Bohr-Modells ist z.B. die Planck-Konstante $\hbar$, während die Hauptquantenzahl n und der Energiegewinn ΔE von der Umlaufbahn $n+1$ zur Umlaufbahn n Variablen sind, die den Drehimpuls $L = n\hbar$ des Elektrons auf der n-ten Schale und die Frequenz $\omega = \Delta E/\hbar$ des emittierten Photons bestimmen. Formal lautet (2.2) damit

$$\begin{pmatrix} L \\ \omega \end{pmatrix} = f(n, \Delta E; \hbar) = \begin{pmatrix} n\hbar \\ \Delta E/\hbar \end{pmatrix}, \tag{2.3}$$

d.h. $x, y \in \mathbb{R}^2$, $\theta \in \mathbb{R}$ und $f : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$, mit $x = \binom{n}{\Delta E}$, $y = \binom{L}{\omega}$, und $\theta = \hbar$. □

Beispiel 2.8. *(Klimamodelle)* Zu den komplexesten mathematischen Modellen, die auch beträchtliche soziale und wirtschaftliche Auswirkungen haben, gehören die Klimamodelle des IPCC (auf Deutsch oft: Weltklimarat), einem zwischenstaatlichen Gremium der Vereinten Nationen, das den Auftrag hat, wissenschaftliche Informationen zum Verständnis des menschlichen Einflusses auf den Klimawandel auf der Erde bereitzustellen.[20] Der

[20] Die Vereinten Nationen befürworteten offiziell die Gründung des IPCC (Zwischenstaatlicher Ausschuss für Klimaänderungen) im Jahr 1988. Er betreibt keine eigene Forschung, sondern erstellt umfassende Sachstandsberichte die jeweils auf früheren Berichten aufbauen und den aktuellen wissenschaftlichen Kenntnisstand beleuchten. Insbesondere die Arbeitsgruppe 1 hat ihren Beitrag zum sechsten Sachstandsbericht über die physikalischen Grundlagen im Jahr 2021 (IPCC, 2021) veröffentlicht.

Klimawandel ist gut dokumentiert und wird hauptsächlich durch den stetig steigenden Ausstoß von Treibhausgasen verursacht,[21] siehe auch Abbildung 7.1 auf Seite 97. Die wirtschaftlichen Verluste aufgrund des Klimawandels haben in den letzten 50 Jahren stetig zugenommen (Abb. 2.3). Die Klimamodelle, die in den IPCC-Berichten verwendet

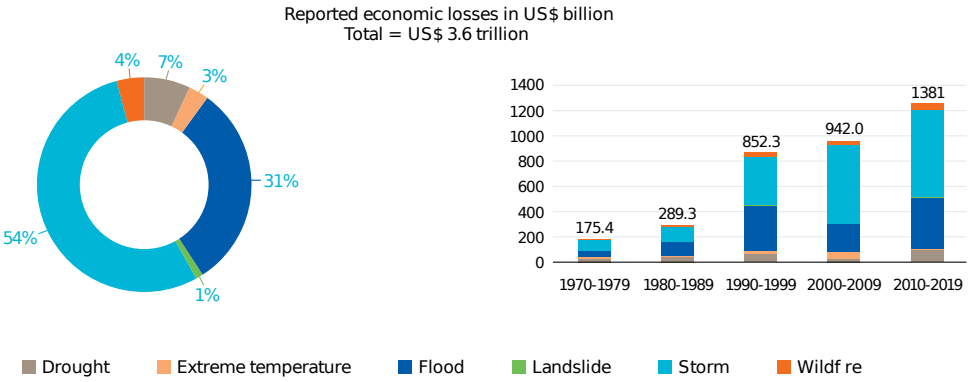


Abbildung 2.3. Globale ökonomische Verluste aufgrund von Katastrophen nach Jahrzehnt. Source: WMO (2021:S. 19)

werden, zielen darauf ab, die wesentlichen Einflüsse auf das Klima darzustellen.[22] Der sechste Sachstandsbericht des IPCC verwendet CMIP6, ein System, das verschiedene Klimamodelle miteinander koppelt, sogenannte Allgemeine Zirkulationsmodelle *(general circulation models, GCM)*. Sie modellieren die Zirkulationen der Ozeane und der Atmosphäre und basieren auf einem System physikalischer Gleichungen.[23] CMIP6 dient der Berechnung von fünf möglichen sozioökonomischen Entwicklungspfaden als *Szenarios*. Sie werden als *shared socio-economic pathways* (SSP) bezeichnet, sind von SSP1 bis

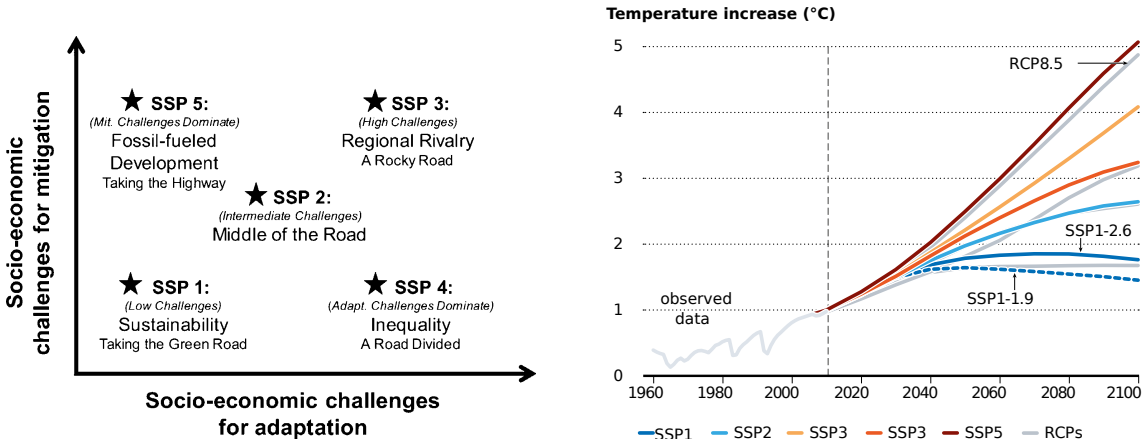


Abbildung 2.4. Die fünf von IPCC (2021) betrachteten SSPs und ihre jeweiligen Vorhersagen zum Temperaturanstieg bis 2100. Quellen: O'Neill et al. (2017) und Tollefson (2020)

SSP5 nummeriert und erweitern die früheren Szenarien RPCs *(repräsentative Konzentrationspfade)*, die eine Reihe alternativer Pfade für die atmosphärischen Konzentrationen der wichtigsten Treibhausgase beschreiben.[24] Jedes dieser fünf SSP-Szenarien stellt ein

[21] e.g. Feldman et al. (2015); IPCC (2021):§1.2, §1.4.1.
[22] IPCC (2021):§1.5.3.
[23] cf. Schönwiese (2008):S. 245.
[24] O'Neill et al. (2017); Tollefson (2021); IPCC (2021):TS.1.3.1, §1.6.1.1; zur Methodik siehe z.B. Rotmans et al. (2000).

eigenes Narrativ dar. Sie werden wie folgt beschrieben.

- *SSP1: Nachhaltigkeit.* Die Welt blickt weniger auf Wirtschaftswachstum und mehr auf globales Wohlergehen und einen ressourcenschonenden Konsum. Investitionen in Bildung und Gesundheit fördern den demografischen Übergang zu einer global bald wieder sinkenden Bevölkerungsgröße. Ungleichheiten in sowie unter den Staaten sinken.

- *SSP2: Der mittlere Weg.* Bisherige gesellschaftliche, ökonomische und technologische Muster setzen sich ähnlich in die Zukunft fort. Die Nationen machen ungleichmäßig verteilte Fortschritte. Trotz allmählich weniger intensiver Nutzung von Rohstoffen kommt es zu weiteren Umweltveränderungen. Soziale und ökologische Verwundbarkeiten bestehen länger fort und sind schwerer zu beseitigen.

- *SSP3: Regionale Rivalitäten.* Aufkommende nationalistische Tendenzen lassen regionale Sicherheitsfragen wichtiger erscheinen als globale Probleme. Der Wunsch zur lokalen Selbstversorgung mit Energie und Nahrungsmitteln hemmt zwischenstaatliche Kooperation, Entwicklungen sind langsamer und kostspieliger, protektionistische Maßnahmen behindern den Handel und stehen gemeinsamen Zielen entgegen. In weniger wohlhabenden Teilen der Welt gibt es starkes Bevölkerungswachstum und Umweltzerstörungen.

- *SSP4: Ungleichheiten.* Die ökonomische und politische Macht verteilt sich zunehmend verschieden, sowohl innerhalb von Staaten als auch zwischen ihnen. Es entsteht eine Kluft: einerseits eine gut vernetzte, internationalisierte Gesellschaftsschicht als Treiberin eines wirtschaftlichen und wissenschaftlichen Aufschwungs, andererseits global fragmentierte Milieus mit schlechter Ausbildung und begrenzten finanziellen Möglichkeiten, die davon kaum profitieren. Der soziale Zusammenhalt schwindet, und es kommt häufig zu Unruhen. Umweltschutzmaßnahmen konzentrieren sich auf das Umfeld der Elite.

- *SSP5: Fossil befeuerte Entwicklung.* Die Welt setzt darauf, mit Innovationen, gemeinsamen Wirtschaftsräumen und gesellschaftlicher Teilhabe schnelle Fortschritte hervorzubringen, die wiederum eine nachhaltige Entwicklung ermöglichen. Die Prozesse werden jedoch von der intensiven Nutzung fossiler Energiequellen sowie einer energieintensiven Lebensführung angetrieben. Die Weltwirtschaft wächst rasch. Lokale Umweltprobleme wie Luftverschmutzung lassen sich bewältigen, und auch Risiken durch einen starken Klimawandel sollen technologisch in den Griff bekommen werden.

Diese Szenarien wurden bereits 2014 entwickelt, und seither scheint das Szenario SSP3 dem Weg, den die Großmächte der Welt einschlagen, sehr nahe zu kommen. Es ist geprägt von einem Wiederaufleben des Nationalismus. Die Sorge um die wirtschaftliche Wettbewerbsfähigkeit und Sicherheit führt zu Handelskriegen. Im Laufe der Jahrzehnte werden die nationalen Bemühungen, die Energie- und Nahrungsmittelversorgung zu sichern, die globale Entwicklung behindern. Investitionen in Bildung und Technologie gehen zurück. In einer solchen Welt wäre es schwierig, die Treibhausgase einzudämmen, und auch die Anpassung an den Klimawandel wäre nicht einfacher. In diesem Szenario wird die globale Durchschnittstemperatur voraussichtlich um mehr als 4°C über das vorindustrielle Niveau ansteigen. Als das Szenario entwickelt wurde, schien es eher unwahrscheinlich. Aber leider ist es das nicht. □

Mathematische Modelle können unterteilt werden in deterministische Modelle und statistische Modelle. In einem *deterministischen Modell* ist jeder Zustand der Variablen eindeutig durch die Parameter im Modell und möglicherweise durch frühere Zustände dieser Variablen bestimmt. Im Gegensatz dazu spielt in einem statistischen Modell der Zufall eine Rolle und die Variablen werden nicht genau und eindeutig durch die Parameterwerte beschrieben, sondern durch bestimmte Wahrscheinlichkeitsverteilungen.

deterministische vs. statistische Modelle

2.4 Ockhams Rasiermesser und Modellauswahl

Ein häufiges Problem beim maschinellen Lernen und in der Statistik besteht darin, aus mehreren Modellkandidaten das beste Modell auszuwählen, das beobachtete Daten erklärt. Aber was ist ein „bestes“ Modell? Erstaunlicherweise gibt uns ein altes philosophisches Prinzip einen Leitfaden, um die Güte eines Modells mathematisch präzise zu formulieren. Dieses Prinzip werden wir hier betrachten. Beginnen wir mit einem einfachen „Modellproblem“ aus dem alltäglichen Leben, in dem wir im Allgemeinen nicht von einem „Modell“, sondern von einer „Hypothese“ sprechen. Daher werden wir die Begriffe in diesem Abschnitt austauschbar verwenden. Wie viele Kisten befinden sich hinter der Säule in Abbildung 2.5? Die übliche spontane Antwort lautet: Eine! Ist diese Vermutung rational? Oder ist sie völlig willkürlich? MacKay (2003:Chapter 28) argumentiert, dass

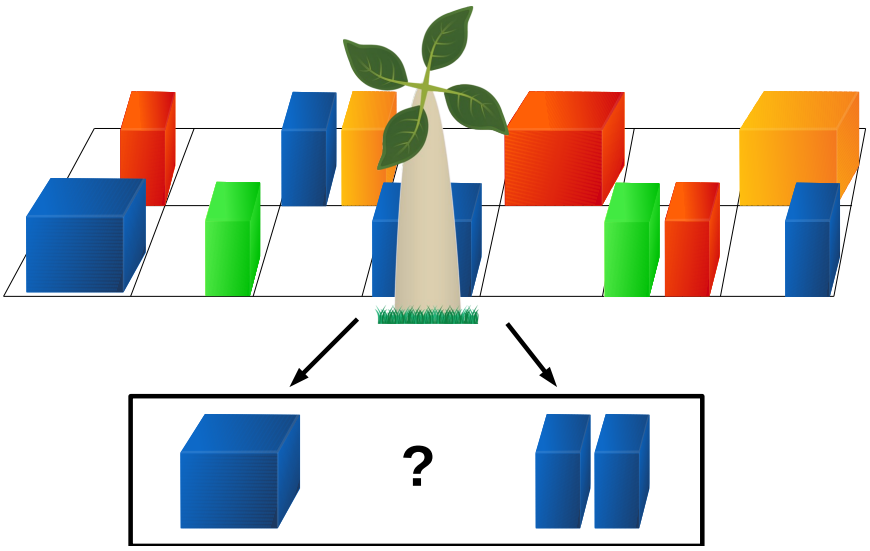


Abbildung 2.5. Wie viele Kisten befinden sich hinter der Säule? Eine oder zwei? Oder sogar mehr? Quelle: MacKay (2003:S. 343)

diese Vermutung auf dem Forschungsprinzip von *Ockhams Rasiermesser* beruht. Dieser Grundsatz besagt, dass einfache Modelle und Hypothesen zu bevorzugen sind: „Akzeptiere die einfachste Erklärung, die zu den Beobachtungsdaten passt.“ Gemäß Ockhams Rasiermesser sollten wir also zu dem Schluss kommen, dass sich nur eine Kiste hinter der Säule befindet.

Gibt es einen überzeugenden Grund für die Annahme, dass es höchstwahrscheinlich nur eine einzige Kiste gibt? Vielleicht hat Ihre Intuition das Argument verwendet: „Nun, es wäre ein bemerkenswerter Zufall, wenn zwei Kisten genau die gleiche Höhe und Farbe hätten“. Damit Algorithmen für maschinelles Lernen und künstliche Intelligenz diese Daten richtig interpretieren können, müssen wir dieses intuitive Gefühl in einen konkreten

mathematischen Formalismus übersetzen.[25]

Neben dem bisherigen empirischen Erfolg von Ockhams Rasiermesser – William von Ockham lebte vor etwa 700 Jahren! – gibt es zwei Gründe für seine Rechtfertigung. Der erste ist ästhetischer Natur: „Eine mathematisch schöne Theorie ist wahrscheinlicher richtig als eine hässliche, die mit einigen experimentellen Daten übereinstimmt“, wie es der große Physiker Paul Dirac ausdrückte.[26] Der zweite Grund ist die Wahrscheinlichkeit: Wenn wir die Wahrscheinlichkeit verschiedener möglicher Modelle zur Erklärung eines Phänomens abschätzen oder sogar berechnen können, dann ist das Modell mit der höchsten Wahrscheinlichkeit zu akzeptieren. Es ist nämlich wahrscheinlicher, dass sich hinter dem Baum eine Kiste befindet, als dass es zwei sind. Untersuchen wir das näher in folgendem Beispiel.

Beispiel 2.9. Wie viele Kisten befinden sich hinter der Säule in Abbildung 2.5?[27] Nehmen wir an, dass jede Kiste entweder ein dünner Quader oder ein Würfel ist, der auf äquipartitionierten Quadraten steht, die durch ein 7×2-Gitter angeordnet sind. Zwei Quader können sich ein einziges Quadrat teilen. Wir nehmen außerdem an, dass es 4 verschiedene Farben von Quadern gibt, und dass alle Eigenschaften der Quader – Form, Position und Farbe – die gleiche Wahrscheinlichkeiten haben. Dann hat das Modell M_1, das besagt, dass es nur eine Kiste hinter der Säule gibt, zwei freie Parameter, seine Position und seine Farbe. Die Position kann eines von $7 \cdot 2 = 14$ Quadraten einnehmen, und die Farbe ist eine von vier. Das Modell M_2, das besagt, dass sich hinter der Säule zwei Kästen befinden, hat dementsprechend vier freie Parameter, für jede Kiste deren Quadrat und deren Farbe. Welche Evidenz hat jedes dieser Modelle? Zunächst stellen wir fest, dass die A-Priori-Wahrscheinlichkeiten $P(M_i)$ der Hypothesen gleich sind,

$$P(M_1) = P(M_2) = \frac{1}{2}, \tag{2.4}$$

da eine zufällig gewählte Kiste mit der Wahrscheinlichkeit $\frac{1}{2}$ ein Würfel ist. Dann ist die Wahrscheinlichkeit für die Beobachtung der Daten X, dass sich ein grüner Würfel hinter der Säule befindet, bei gegebenem Modell M_1 gegeben durch

$$P(X \mid M_1) = \frac{1}{14} \cdot \frac{1}{4} = \frac{1}{56}, \tag{2.5}$$

da er mit der Wahrscheinlichkeit $\frac{1}{14}$ auf dem Platz hinter der Säule steht und mit der Wahrscheinlichkeit $\frac{1}{4}$ grün ist. Analog dazu gilt für das Modell M_2, welches zwei Quader hinter der Säule vermutet, dass die Parameter jedes einzelnen Quaders die gleichen Wahrscheinlichkeiten wie der Würfel für das Modell M_1 haben, aber es zwei verschiedene Kombinationen der beiden Farben gibt, d.h.

$$P(X \mid M_2) = \left(\frac{1}{14} \cdot \frac{1}{4}\right)^2 = \frac{1}{3136}, \tag{2.6}$$

d.h. $P(X \mid M_2) = P^2(X \mid M_1)$. Dann ist die Wahrscheinlichkeit $P(M_i \mid X)$ für die Gültigkeit des Modells i durch den Satz von Bayes gegeben, und somit lautet das Verhältnis der Wahrscheinlichkeiten der beiden Modelle

$$\frac{P(M_1 \mid X)}{P(M_2 \mid X)} = \frac{P(X \mid M_1)\, P(M_2)}{P(X \mid M_2)\, P(M_1)} = \frac{P(X \mid M_1)}{P(X \mid M_2)} = \frac{1}{P(X \mid M_1)} = 56. \tag{2.7}$$

[25] MacKay (2003):S. 343.

[26] https://www.jstor.org/stable/24936146

[27] modifiziert nach MacKay (2003):§28.2.

Daher ist die Wahrscheinlichkeit des Modells M_1 viel größer als die des Modells M_2, wenn man die Beobachtungsdaten X betrachtet. □

Definition 2.10. Das Verhältnis in Gleichung (2.7) nennen wir allgemein

Verhältnis der Wahrscheinlichkeiten

$$\boxed{\frac{P(M_1 \mid X)}{P(M_2 \mid X)} = \frac{P(X \mid M_1)\, P(M_2)}{P(X \mid M_2)\, P(M_1)},} \tag{2.8}$$

das *Wahrscheinlichkeitsverhältnis*,[28] unter der Voraussetzung der Hypothesen M_1 und M_2 und den Beobachtungsdaten X. Ist das Wahrscheinlichkeitsverhältnis größer als 1, ist die Hypothese M_1 zu bevorzugen, ist es kleiner als 1, die Hypothese M_2. □

Bemerkung 2.11. Bei der Auswahl statistischer Modelle sind die A-priori-Wahrscheinlichkeiten $P(M_1)$ und $P(M_2)$ üblicherweise unbekannt, weren aber als gleich angenommen. Das Wahrscheinlichkeitsverhältnis (2.8) ergibt dann

$$\frac{P(M_1 \mid X)}{P(M_2 \mid X)} = \frac{P(X \mid M_1)}{P(X \mid M_2)}, \tag{2.9}$$

d.h. der Quotient der Wahrscheinlichkeiten der Modelle ist gleich dem Quotienten ihrer Likelihood, gegeben die Daten X. □

Beispiel 2.12. *(Der Verdacht)* Eine Person hat am Tatort Spuren ihres eigenen Blutes hinterlassen. Ein Verdächtiger, Oliver, wird getestet und hat die Blutgruppe 0. Die Blutgruppe 0 ist in der örtlichen Bevölkerung mit einer Häufigkeit von 60% weit verbreitet. Was ist die Evidenz dafür, dass Oliver bei dem Verbrechen anwesend war?

Lösung: Wir haben zwei Hypothesen, nämlich M_1: Oliver war am Tatort anwesend, M_2: Oliver war nicht anwesend. Die Daten sind einfach X: Blutgruppe 0 wird gefunden. Dann sehen wir sofort die Wahrscheinlichkeiten

$$P(X \mid M_1) = 1, \qquad P(X \mid M_2) = 0.6, \tag{2.10}$$

da bei der Hypothese M_1, dass Oliver anwesend war, der Typ O mit Sicherheit gefunden wird, aber bei der Hypothese M_2, dass er nicht anwesend war, ist die Wahrscheinlichkeit genauso groß wie bei einer zufälligen Auswahl aus der Population. Der Satz von Bayes 1.5 impliziert dann

$$P(M_1 \mid X) = \frac{P(X \mid M_1)\, P(M_1)}{P(X)} = \frac{P(M_1)}{P(X)}, \qquad P(M_2 \mid X) = \frac{P(X \mid M_2)\, P(M_2)}{P(X)}. \tag{2.11}$$

Obwohl wir die Wahrscheinlichkeit $P(X)$, Typ 0 am Tatort zu beobachten, nicht kennen, können wir das Wahrscheinlichkeitsverhältnis (2.8) mit $P(X \mid M_1) = P(X \mid M_2) = \frac{1}{2}$ ableiten als

$$\frac{P(M_1 \mid X)}{P(M_2 \mid X)} = \frac{P(X \mid M_1)}{P(X \mid M_2)} = \frac{1}{0,6} = 1.667 > 1. \tag{2.12}$$

Daher ist die Hypothese M_1 leicht zu bevorzugen. Die Daten liefern also einen schwachen Beweis dafür, dass Oliver am Tatort anwesend war. □

Beispiel 2.13. *(Das Verbrechen mit zwei Verdächtigen)*[29] Zwei Personen haben am Tatort Spuren ihres eigenen Blutes hinterlassen. Die Blutgruppen der beiden Spuren sind 0 und AB. Ein Verdächtiger, Oliver, wird getestet und hat die Blutgruppe 0. Die Blutgruppe 0

[28] Brandt (1999):S. 185; MacKay (2003):S. 28, 344; Hastie et al. (2009):S. 234.

[29] MacKay (2003):S. 55–56.

ist in der örtlichen Bevölkerung weit verbreitet, mit einer Häufigkeit von $p_O = 60\%$, Typ AB ist ein seltener Typ mit einer Häufigkeit von $p_{AB} = 1\%$. Was ist die Evidenz dafür, dass Oliver bei dem Verbrechen anwesend war?

Lösung: Wir haben zwei Hypothesen, M_1: Oliver und eine unbekannte Person waren am Tatort anwesend, M_2: zwei unbekannte Personen waren anwesend. Die Daten sind einfach X: die Blutgruppen 0 und AB wurden gefunden. Dann sehen wir sofort die Wahrscheinlichkeiten

$$P(X \mid M_1) = p_{AB} = 0.01, \qquad P(X \mid M_2) = 2\, p_O p_{AB} = 0.012, \tag{2.13}$$

da bei der Hypothese M_1, dass Oliver anwesend war, der Typ O mit Sicherheit gefunden wird und der Typ AB durch p_{AB}, aber bei der Hypothese M_2, dass er nicht anwesend war, ist die Wahrscheinlichkeit genauso groß wie bei einer zufälligen Auswahl aus der Grundgesamtheit, dass die erste Person den Typ O und die zweite den Typ AB hat, sowie die Wahrscheinlichkeit, dass die erste Person den Typ AB und die zweite den Typ O hat. Dann ist das Wahrscheinlichkeitsverhältnis (2.8), mit $P(X \mid M_1) = P(X \mid M_2) = \frac{1}{2}$, gegeben durch

$$\frac{P(M_1 \mid X)}{P(M_2 \mid X)} = \frac{P(X \mid M_1)}{P(X \mid M_2)} = \frac{p_{AB}}{2\, p_O p_{AB}} = \frac{1}{2\, p_O} = 0.83 < 1. \tag{2.14}$$

Daher ist die Hypothese M_2 leicht zu bevorzugen. Somit liefern die Daten in der Tat einen schwachen Beweis gegen den Verdacht, dass Oliver anwesend war. □

Bemerkung 2.14. *(Bezug zu statistischen Hypothesentests)* Das in diesem Abschnitt beschriebene Verfahren zur Anwendung von Ockhams Rasiermesser ist Teil des sogenannten „Bayes'schen Schließens" *(Bayesian inference)*. Es stellt eine Alternative zu statistischen Standardhypothesentests oder „Nullhypothesentests" dar. Hier wird eine so genannte „Nullhypothese" H_0 gegen eine Alternativhypothese H_1 getestet. Die Alternativhypothese ist die zu prüfende Forschungshypothese.[30] Sie wird also angenommen, wenn die Nullhypothese abgelehnt wird. Aus Sicht der Forschungshypothese gibt es also zwei mögliche Fehler:

Nullhypothese

	H_0 ist wahr	H_1 ist wahr
H_0 wird akzeptiert	*Spezifizität* $(1-\alpha)$ „wahr negativ"	Fehler 2. Art (β) „falsch negativ"
H_0 wird abgelehnt	Fehler 1. Art (α) „falsch positiv"	*Sensitivität* $(1-\beta)$ „wahr positiv"

(2.15)

Ziel eines Hypothesentests ist es, die Wahrscheinlichkeit zu berechnen, einen Fehler 1. Art zu erhalten, d.h., dass Hypothese H_0 abgelehnt wird, obwohl sie richtig ist:

$$p = P(H_0 \text{ abgelehnt} \mid H_0 \text{ richtig}) < \alpha. \tag{2.16}$$

Diese Wahrscheinlichkeit ist der so genannte „p-Wert" und sollte kleiner als ein bestimmtes „Signifikanzniveau" α sein, um die Nullhypothese abzulehnen und die Alternativhypothese zu akzeptieren. Er sollte sehr klein sein, normalerweise $p < 0,05$. Das Testen der Nullhypothese ist also eine *reductio ad absurdum*, da die Forschungshypothese als gültig angenommen wird, wenn ihre Gegenbehauptung sehr unplausibel ist. Der große Nachteil gegenüber Ockhams Rasiermesser ist, dass wir keine Ahnung darüber haben, wie viel besser oder schlechter die Nullhypothese ist. Darüber hinaus ist Ockhams Rasiermesser sehr viel flexibler und ermöglicht, mehrere optionale Alternativen zu vergleichen und die plausibelste davon auszuwählen. Es liefert sogar ein Vergleichsmaß der Hypothesen, gegeben durch deren Wahrscheinlichkeitsverhältnis.[31] □

Die Forschungshypothese wird nur angenommen, wenn die Nullhypothese abgelehnt wird.

[30] vgl. Wermuth und Streit (2007):S. 192.

[31] für einen kritischen Vergleich siehe z.B. MacKay (2003):S. 458.

2.5 Übungsaufgaben

Aufgabe 2.1 (**Logisches Schließen**)**.** Betrachten Sie die drei Aussagen „Es regnet“, „Die Straße ist nass“, „Wenn es regnet, ist die Straße nass“. Formalisieren Sie diese Aussagen durch die Variablen A, B und $A \Rightarrow B$, und wenden Sie sie an, um eine Deduktion, eine Induktion und eine Abduktion gemäß Tabelle 2.1 zu bilden, wie es in Beispiel 2.1 geschieht.

Aufgabe 2.2 (**Modellauswahl durch Wahrscheinlichkeitsverhältnis**)**.** Es gibt zwei Arten von verbogenen Münzen, A und B. Eine Münze vom Typ A zeigt mit der Wahrscheinlichkeit $p_A = \frac{1}{3}$ Kopf, während eine Münze vom Typ B mit der Wahrscheinlichkeit $p_B = \frac{2}{3}$ Kopf zeigt.

	A	B
Kopf	$1/3$	$2/3$
Zahl	$2/3$	$1/3$

(2.17)

Nehmen wir an, wir wählen eine Münze nach dem Zufallsprinzip und werfen viermal Zahl und einmal Kopf. Ist es wahrscheinlicher, dass sie vom Typ A oder B ist? Oder anders ausgedrückt: Passt Modell A oder Modell B besser zu den Daten X?

3

Grundlagen des maschinellen Lernens

Kapitelübersicht

3.1 Was ist maschinelles Lernen? 32
3.2 Statistische Variablen 34
3.3 Statistische Modelle 36
3.4 Verfahren des maschinellen Lernens 37
3.4.1 Klassifikation nach Variablentyp 39
3.4.2 Einteilung nach Lernart 39
3.4.3 Einteilung nach Strukturerkennungsgrad 40
3.5 Probleme des maschinellen Lernens 41
3.5.1 Probleme der Wahl des Modells: Underfitting und Overfitting 41
3.5.2 Datenqualität 42

In diesem Kapitel werden die wichtigsten Begriffe, die in der Folge verwendet werden, kurz erläutert. Obwohl sie hauptsächlich Sachverhalte beschreiben, die aus Grundkursen der Statistik oder des maschinellen Lernens bekannt sein könnten, werden sie hier auch aufgeführt, um die in diesem Skript verwendeten Notationen vorzustellen.

3.1 Was ist maschinelles Lernen?

Das Konzept des maschinellen Lernens ermöglicht einen anderen Blickwinkel, unter dem statistische Analyseverfahren betrachtet und eingeteilt werden können. Das *maschinelle Lernen (Machine Learning)* ist die Technik der Programmierung von Computern, so dass sie aus Daten lernen können, ohne dass die zu erkennenden Regeln oder Muster explizit implementiert sind[1]. Das maschinelle Lernen ist ein Teilgebiet der Informatik und eng verknüpft mit den Begriffen Künstliche Intelligenz, Data Science und Data Mining, vgl. Tabelle 3.1. *Data Science* ist ein wichtiges Teilgebiet der Künstlichen Intelligenz und behandelt allgemein die Extraktion von Wissen aus Daten. Ihre wesentlichen Ansätze sind das maschinelle Lernen und das Data Mining. Während das Data Mining vorwiegend darauf abzielt, Hilfsmittel und Werkzeuge für die Anwender:innen zur Analyse von Datenbeständen zu liefern, steht beim maschinellen Lernen die automatisierte Erzeugung von Wissen im Vordergrund, also die Mechanisierung des Lernens: Das Erkennen von Re-

[1] Géron (2017):§1.

Künstliche Intelligenz (KI)			
Data Science		**Intelligenzforschung**	···
Maschinelles Lernen	**Data Mining**	···	···
Automatisiert aus Daten lernen	Muster und Regelmäßigkeiten in Daten erkennen	···	···
Neuronale Netze, Regression, Clusteranalyse, ...		···	···

Tabelle 3.1. Einordnung von Gebieten der künstlichen Intelligenz

geln oder Mustern soll nicht durch explizite Programieranweisungen geschehen, sondern durch Lernprozesse anhand von Trainings- oder Beobachtungsdaten. Allerdings ist die Grenze zwischen maschinellem Lernen und Data Mining fließend. Viele der Methoden des maschinellen Lernens und des Data Mining sind gleich, insbesondere ein großer Teil der verwendeten statistischen Verfahren. Die beiden Begriffe entstanden fast gleichzeitig in 1990er Jahren.[2]

Beispiel 3.1 (*Spamfilter*)**.** Ein Beispiel für eine Software des maschinellen Lernens ist der Spamfilter in Ihrem E-Mail-Programm. Er lernt permanent anhand von Trainingsbeispielen, die Sie als „Spam“ oder „Ham“ markieren, wird aber nicht aktiv upgedatet. □

Im Bereich des maschinellen Lernens gibt es drei wesentliche Lernarten, das überwachte Lernen, das unüberwachte Lernen und das bestärkende Lernen.

Überwachtes Lernen (supervised learning). Hierbei wird ein vorbereitetes Set von Trainingsdaten mit den jeweils bekannten Lösungen (*Labels*) erstellt, ähnlich wie die Musterlösungen von Probeklausuren. Das System stellt damit die Parameter seines (expliziten oder impliziten) Modells ein. Nach dieser Lernphase labelt das System dann einen neuen Datenpunkt *(Instance)*, dessen Lösung es nicht kennt, anhand der eingestellten Parameter. D. h. die „echte“ Klausur wird geschrieben. Ein Beispiel dafür ist ein Spamfilter, der jede Mail mit „Spam“ oder „Ham“ labelt, wie bereits in Beispiel 3.1 erläutert.

Unüberwachtes Lernen (unsupervised learning). Das System versucht, ohne Anleitung aus Trainingsdaten zu lernen. Das entspräche in etwa der Situation, mit Probeklausuren ohne Musterlösung zu lernen. Nach der Lernphase kann ein neuer Datenpunkt gelabelt werden, d.h. die „echte“ Klausur geschrieben werden.

Bestärkendes Lernen (reinforcement learning). Dies ist eine iterative Lernart. Das System *(„Agent“)* beobachtet seine Umgebung, wählt *„Aktionen“* aus einer *„Strategie“ (policy)* aus und führt sie aus. Jede Aktion wird mit einer gegebenen *Belohnungsfunktion* bewertet, d.h. belohnt oder bestraft. Die Aktion wird entsprechend in der Strategie berücksichtigt oder verworfen, bevor im nächsten Iterationsschritt erneut die Umgebung beobachtet wird und der Zyklus mit der veränderten Strategie von vorne beginnt. Das entspräche beim Klausurenlernen der Vorgehensweise, eine „echte“ Klausur mitzuschreiben, abhängig vom Klausurergebnis die gelernten Themen („Strategie“) anzupassen und die nächste Klausur wieder mitzuschreiben.

[2] Krahl et al. (1998); Mitchell (1997).

3.2 Statistische Variablen

Statistische Analysen beruhen auf beobachteten oder gemessenen Daten. Zur informationstechnischen Verarbeitung werden sie als Zahlen dargestellt. Grundsätzlich bestimmt dabei die betrachtete Eigenschaft, wie gut man ihre Ausprägung messen kann, d. h. wie gut man sie in Zahlen ausdrücken kann. So kann z.B. die Körpergröße eines Menschen sehr leicht in Zahlen ausgedrückt werden, seine Intelligenz, seine Motivation oder sein Gesundheitszustand dagegen nur sehr schwer[3].

Die Ausprägungen einer gemessenen Eigenschaft werden auf einer Skala abgetragen. Abhängig davon, wie die Eigenschaft eines Objektes ausgedrückt werden kann, unterscheidet man in der Statistik verschiedene Skalenniveaus. Es gibt zwei Hauptkategorien, die *kategorialen* und die *metrischen* Skalenniveaus. Wesentlich für die kategorialen Skalenniveaus ist, dass ihre Werte nicht Zahlen sein müssen oder, wenn sie Zahlen sind, dann nur als Ordnungsummern, nicht als Rechengröße. Demgegenüber kann man die

Skalenniveau

Skalenniveau		Merkmal	Messbare Eigenschaften	Beispiel	Berechenbare Größen
kategorial	**Nominalskala**	rein qualitativ	Häufigkeit	Blutgruppe, Geschlecht	Modus
	Ordinalskala	sortierbar	Rangfolge	Bildungsabschluss, Turnierplatzierung	Median, Quantil
metrisch	**Intervallskala**	subtrahierbar	Abstand	Datum, Temperatur in °C	Arithmetisches Mittel, Standardabweichung
	Verhältnisskala	dividierbar	natürlicher Nullpunkt	Alter, Preis, Prozent, Kelvin	Geometrisches Mittel, Variationskoeffizient

Tabelle 3.2. Skalenniveaus statistischer Variablen. Jede Skala enthält die Merkmale, messbaren Eigenschaften und berechenbaren Größen aller vorhergehenden Skalen.

Werte metrischer Skalen sinnvoll addieren und subtrahieren und entsprechend statistische Größen wie Mittelwert oder Standardabweichung berechnen. Einen Überblick über die verschiedenen Skalenniveaus gibt Tabelle 3.2. Sie sind dort so angeordnet, dass jedes Skalenniveau die Merkmale, messbaren Eigenschaften und berechenbaren statistischen Größen aller vorhergehenden Skalenniveaus enthält.

Nominalskala

Das niedrigste Skalenniveau ist die Nominalskala, deren Werte rein qualitativ sind und nur auf Gleichheit oder Ungleichheit geprüft werden können ($x = y$, $x \neq y$). Beispiele sind Skalen für Blutgruppen (0, A+, A-, ...) oder für das Geschlecht (m, w, d). Als einzige statistisch messbare Eigenschaft lassen sich bei nominalskalierten Daten lediglich die Häufigkeiten der einzelnen Merkmalsausprägungen ermitteln, aus der dann als statistisches Maß der Modus berechenbar ist, also der am häufigsten vorkommende Wert einer Stichprobe. Die Werte einer Nominalskala können aber nicht sinnvoll der Größe nach sortiert werden.

Ordinalskala

Dies gelingt jedoch bei dem nächsthöheren Skalenniveau, der Ordinalskala, deren Werte eine Rangordnung ausdrücken. Sie können zwar ihrer Größe nach verglichen werden

[3] Backhaus et al. (2016):S. 10.

($x < y$), die Abstände zwischen den Rangwerten oder den Ordnungskategorien sagen jedoch nichts über die Abstände der Merkmale der ihnen zugrunde liegenden Objekte aus. Ein Beispiel ist der Tabellenplatz in der Bundesliga, der zwar die Leistung einer Mannschaft im Vergleich zu den anderen ausdrückt, aber der Erstplatzierte ist sicher nicht um denselben Betrag besser als der Zweitplatzierte wie der Zweitplatzierte besser als der Drittplatzierte ist.

Intervallskala

Das nächsthöhere Skalenniveau ist die Intervallskala. Sie weist gleichgroße Skalenabschnitte aus und gehört damit zu den metrischen Skalenniveaus. Ein typisches Beispiel ist die Celsius-Skala zur Temperaturmessung, bei der der Abstand zwischen Gefrierpunkt und Siedepunkt des Wassers in hundert gleichgroße Abschnitte eingeteilt wird. Bei intervallskalierten Daten besitzen also auch die Differenzen zwischen einzelnen Messwerten Informationsgehalt – z.B. großer oder kleiner Temperaturunterschied –. Das ist bei nominalen oder ordinalen Daten ja genau nicht der Fall. Der Nullpunkt (Gefrierpunkt, „Christi Geburt“) ist bei einer Intervallskala aber grundsätzlich willkürlich, da er abhängig von der verwendeten Einheit ist (Celsius versus Fahrenheit, Gregorianischer versus jüdischer oder islamischer Kalender).

Verhältnisskala

Das höchste Skalenniveau stellt die Verhältnisskala dar. Sie unterscheidet sich von der Intervallskala dadurch, dass zusätzlich ein natürlicher Nullpunkt existiert. Er lässt für das betreffende Merkmal meist im Sinne von „nicht vorhanden“ interpretieren. Das ist z.B. bei der Kelvin-Skala oder dem Urknall als Beginn der Zeit der Fall, nicht jedoch bei der Celsius-Skala oder dem Gregorianischen Kalender. Dies trifft auch für die meisten physikalischen Größen (Länge, Gewicht, Geschwindigkeit) oder den meisten ökonomischen Merkmalen (Einkommen, Kosten, Preis) zu. Bei verhältnisskalierten Daten besitzen nicht nur die Differenz, sondern, infolge der Fixierung des Nullpunktes, auch der Quotient bzw. das Verhältnis der Messwerte Informationsgehalt. Verhältnisskalierte Daten erlauben die Anwendung aller arithmetischen Operationen wie auch die Anwendung aller obigen statistischen Maße. Zusätzlich sind z.B. das geometrische Mittel oder der Variationskoeffizient sinnvoll berechenbar.

Zusammenfassend gilt also: Je höher das Skalenniveau ist, desto größer ist der Informationsgehalt der betreffenden Daten und desto mehr Rechenoperationen und statistische Maße lassen sich auf die Daten anwenden. Es ist generell möglich, Daten von einem höheren Skalenniveau auf ein niedrigeres Skalenniveau zu transformieren, nicht aber umgekehrt. Dies kann sinnvoll sein, um die Übersichtlichkeit der Daten zu erhöhen oder um ihre Analyse zu vereinfachen. So werden z.B. häufig Einkommensklassen oder Preisklassen gebildet, die ursprünglich verhältnisskalierte Daten auf eine Intervall-, Ordinal- oder Nominalskala heruntertransformieren. Mit einer Transformation auf ein niedrigeres Skalenniveau ist natürlich immer auch ein Informationsverlust verbunden.

Bemerkung 3.2. So einleuchtend die Einteilung der Skalenniveaus auch ist, so gibt es Grenzfälle, die sich auf den ersten Blick nicht, oder nicht so einfach, zuordnen lassen. Beispielsweise sind Schulnoten („sehr gut“, . . . , „ungenügend“) nominalskaliert, d.h. es macht strenggenommen keinen Sinn, eine Durchschnittsnote zu berechnen; meist liegt der Bewertung aber implizit der Gedanke zugrunde, dass jede Note eine gleich große Leistungsdifferenz abdeckt und Schulnoten daher intervallskaliert sind. Wir werden in diesem Lehrbrief mit Verfahren zu tun haben, die als unabhängige Variable ein ganzes Netzwerk erwarten. Da dieses als Matrix codiert werden kann und Matrizen nicht sortierbar sind, sind solche Verfahren nur nominalskaliert. □

3.3 Statistische Modelle

Ein *statistisches Modell* nimmt a priori einen sachlogisch fundierten Kausalzusammenhang zwischen den Variablen an. Es setzt also eine Vorstellung davon voraus, welche Variablen („Ursache“) welche anderen Variablen („Wirkung“) beeinflussen. Diejenigen Variablen, die die anderen beeinflussen, heißen „unabhängig“, die beeinflussten Variablen heißen „abhängig“.

Definition 3.3. Können die Werte von $n+1$ beobachtbaren Merkmalen *(features)* $x_1, \ldots, x_n$, $y \in \mathbb{R}$, mit Unsicherheiten oder Fehlern gemessen werden, dann ist ein *statistisches Modell* mit den k Parametern $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$ ein funktionaler Zusammenhang

$$\boxed{y = f(\boldsymbol{x}, \boldsymbol{\theta}) + \varepsilon} \tag{3.1}$$

mit einer Funktion $f : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}$ und einem Fehlerterm ε, so dass die Werte des Merkmals y einer bedingten Wahrscheinlichkeit $P(y \mid f(\boldsymbol{x}, \boldsymbol{\theta}))$ gehorchen. Die Merkmale $x_1, \ldots, x_n$ heißen *unabhängige Variablen*, *Prediktoren* oder *exogene Variablen* des Modells, das Merkmal y heißt *abhängige Variable*, oft auch *Zielvariable (target)*, *Antwort (response)* oder *exogene Variable*. □

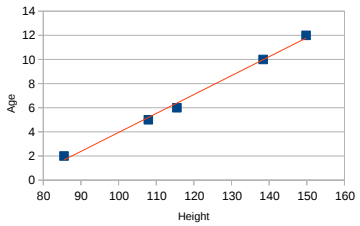
Bemerkung 3.4. Im maschinellen Lernen werden üblicherweise die Funktion f vorausgesetzt und die Parameter $\boldsymbol{\theta}$ anhand von m Beobachtungsdaten der Merkmale

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} \qquad \boldsymbol{X} = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix} \tag{3.2}$$

hergeleitet. Die Methoden zur Bestimmung der Parameterwerte in Abhängigkeit von den Daten variieren je nach Problemstellung. Viele Methoden beinhalten die Minimierung von Residuen, d. h. von geeigneten Abständen der beobachteten Daten von den Modellwerten. Manchmal werden Bayes'sche Methoden verwendet, die die Parameter mit der höchsten Wahrscheinlichkeit auswählen.[4] □

Beispiel 3.5 (*Lineare Regression: Alter und Größe von Kindern*)**.** Gegeben sei die in folgender Tabelle dargestellte kleine Stichprobe für Größe und Alter von $m = 5$ Kindern.

Größe x [cm]	Alter y [yrs]
107,9	5
149,8	12
115,5	6
85,5	2
138,4	10



Größe x und Alter y scheinen korreliert. Ein einfaches Modell für einen funktionalen Zusammenhang könnte lauten:

$$y = \theta_0 + \theta_1 x + \varepsilon \tag{3.3}$$

mit dem Störterm $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ und den Parametern $\theta = (\theta_0, \theta_1) \in \mathbb{R}^2$ (Hier: $\theta_0 = -11{,}75$, $\theta_1 = 0{,}157$).[5] □

Beispiel 3.6 (*Logistische Regression*)**.** Es wurden die in folgender Tabelle aufgeführten Einflüsse von Lernzeiten auf das Bestehen einer Klausur beobachtet.

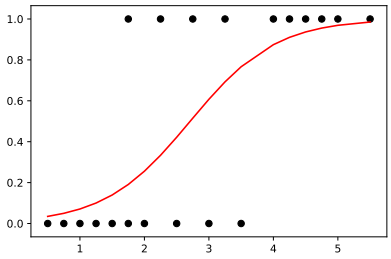
[4] MacKay (2003):S. 28, 347, 535; Hastie et al. (2009):S. 30, 34, 233.
[5] vgl. K. P. Murphy (2012):S. 19.

Stunden (x)	Bestanden (y)
0,50	0
0,75	0
1,00	0
1,25	0
1,50	0
1,75	0
1,75	1

Stunden (x)	Bestanden (y)
2,00	0
2,25	1
2,50	0
2,75	1
3,00	0
3,25	1
3,50	0

Stunden (x)	Bestanden (y)
4,00	1
4,25	1
4,50	1
4,75	1
5,00	1
5,50	1

Darauf lässt sich das folgende statistische Modell für y als Wahrscheinlichkeit des Bestehens bei x Stunden Lernen anwenden:



Modell:
$y = \text{sigm}\,(\theta_0 + \theta_1 x + \varepsilon)$

(Hier: $\theta_0 = -4{,}0777$, $\theta_1 = 1{,}5046$)

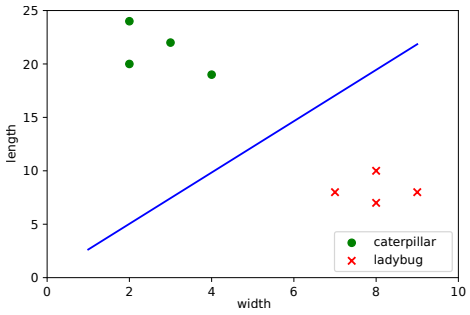
Hier bezeichnet $\text{sigm}\,(x)$ die *Sigmoid-* oder *logistische Funktion*. □

Beispiel 3.7 (*Klassifizierung*)**.** Die folgenden Daten von Insekten wurden beobachtet:

Breite x_1	Länge x_2	Label
9 mm	8 mm	Marienkäfer x
3 mm	22 mm	Raupe ●
8 mm	10 mm	Marienkäfer x
2 mm	20 mm	Raupe ●

Breite x_1	Länge x_2	Label
7 mm	8 mm	Marienkäfer x
2 mm	24 mm	Raupe ●
4 mm	19 mm	Raupe ●
8 mm	7 mm	Marienkäfer x

In ein Diagramm eingetragen, lassen sich die Daten in zwei Kategorien klassifizieren.



Modell:
$y = \text{sgn}\,(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \varepsilon)$

(Hier: $\theta_0 = -0{,}025$, $\theta_1 = -0{,}259$,
$\theta_2 = 0{,}108$)

Hier stellt $y = +1$ die Klasse „Raupe“ und $y = -1$ die Klasse „Marienkäfer“. □

3.4 Verfahren des maschinellen Lernens

Die Verfahren Datenanalyse und des maschinellen Lernens erleben ständig neue Entwicklungen. Darüber hinaus werden diese Methoden zunehmend durch die Verfügbarkeit effizienter und benutzerfreundlicher Software angewendet. Vor diesem Hintergrund wird hier versucht, eine Auswahl von multivariaten Analysemethoden vorzustellen, die sowohl in der universitären Ausbildung als auch in der praktischen Anwendung von besonderer Bedeutung sind. Tabelle 3.3 gibt eine Übersicht über gängige statistische Analyseverfahren, mit denen wir in diesem Skript zu tun haben werden. Für eine umfassendere Übersicht siehe beispielsweise Backhaus et al. (2016:S. 24), de Vries (2020:§1.2) und Hastie et al. (2009).

Verfahren	Typisches Anwendungsbeispiel
Regressionsanalyse	Einfluss von Preis, Werbeausgaben und Einkommen auf die Absatzmenge
Zeitreihenanalyse	Analyse und Prognose der zeitlichen Entwicklung des Absatzvolumens
Diskriminanzanalyse	Bonitätsklasse zur Krediterteilung hinsichtlich soziodemografischer und Merkmale (Alter, Einkommen usw.)
Logistische Regression	Ermittlung des Herzinfarktrisikos von Patienten in Abhängigkeit ihres Alters und ihres Cholesterin-Spiegels.
(Explorative) Faktorenanalyse	Verdichtung einer Vielzahl von Eigenschaftsbeurteilungen auf zugrunde liegende Beurteilungsdimensionen.
Support Vector Machine	Medizinische Diagnose „krank“ oder „gesund“ anhand mehrerer Symptome, z.B. Maximalpuls und Alter für eine Diagnose „herzkrank“[6].
Clusteranalyse	Bildung von Persönlichkeitstypen auf Basis psychografischer Merkmale
Louvain-Methode	Erkennen von Clustern in Netzwerken
Neuronale Netze	Mögliche Einflussfaktoren und Prognose von Aktienkursen.
Soziale Netzwerkanalyse	Messung des Einflusses von Akteuren in einem Netzwerk für effiziente Marketingkampagnen

Tabelle 3.3. Gängige Verfahren der statistischen Datenanalyse und des maschinellen Lernens

Jede dieser Methoden basiert auf einem bestimmten statistischen Modell. Bei der linearen Regression beispielsweise wird ein Modell mit einer linearen Funktion f angewendet, so wie in Gleichung (3.3). Die Wahl der geeigneten Methode zur Untersuchung gegebener statistischer Daten kann also auf die Auswahl „des“ geeigneten Modells reduziert werden. Die Auswahl eines Modell ist ein komplexes Thema und wird in Hastie et al. (2009:§7) ausführlich behandelt, speziell die Auswahl linearer Regressionsmodelle in James et al. (2013).

Um eine Orientierung im „Dschungel“ dieser Modelle zu bieten, klassifizieren wir in den folgenden Abschnitten die verschiedenen statistischen Methoden nach unterschiedlichen Kriterien, nämlich nach den Typ ihrer Variablen, nach der Art des Lernens und nach dem Grad der Strukturerkennung.

3.4.1 Klassifikation nach Variablentyp

Nach unserer obigen Diskussion lassen sich statistische Variablen nach ihren Skalenniveaus unterscheiden, wie in Tabelle 3.2 zusammengefasst. Die Hauptkategorien sind kategoriale und metrische Variablen. Darüber hinaus können wir unabhängige Variablen von abhängigen Variablen unterscheiden. Daher kann jede Methode des maschinellen Lernens durch das Skalenniveau der unabhängigen Variablen und das Skalenniveau der abhängigen Variablen spezifiziert werden.[7] So there are four logical categories which the

		Abhängige Variable	
		metrisch	**kategorial**
Unabhängige Variable	**metrisch**	Regressionsanalyse Hauptkomponentenanalyse Faktorenanalyse	Diskriminanzanalyse Logistische Regression
	kategorial	Varianzanalyse Soziale Netzwerkanalyse	Korrespondenzanalyse Log-lineare Analyse Kontingenzanalyse Auswahlbas. Conjoint-Analyse

Tabelle 3.4. Klassifikation nach Typ der Variablen

methods can be sorted in, as is shown in Table 3.4. This table can be used to decide from the pure structure of the data to be analysed which method can be applied.

3.4.2 Einteilung nach Lernart

Wie wir in Abschnitt 3.1 gesehen haben, gibt es drei Arten des Lernens: überwachtes, unüberwachtes und bestärkenes Lernen. Damit können Verfahren des maschinellen Lernens

Lernart	**Verfahren**
Überwachtes Lernen	Regressionsanalyse Zeitreihenanalyse Logistische Regression Diskriminanzanalyse Support Vector Machine Neuronale Netze
Unüberwachtes Lernen	Hauptkomponentenanalyse Clusteranalyse Faktorenanalyse Soziale Netzwerkanalyse Louvain-Methode
Bestärkendes Lernen	Tiefe Neuronale Netze A/B-Test

Tabelle 3.5. Einteilung statistischer Verfahren nach Lernart.

nach ihrer Art des Lernens eingeteilt werden, wie es in Tabelle 3.5 für einige von ihnen

[7] cf. Backhaus et al. (2015); Backhaus et al. (2016).

gezeigt wird. Die Regressionsanalyse beispielsweise kann als ein Verfahren überwachten Lernens angesehen werden, da ein Satz von Trainingsdaten mit Werten sowohl der unabhängigen als auch der abhängigen Variablen erforderlich ist. Andererseits sind Methoden wie die Hauptkomponentenanalyse oder die Clusteranalyse Verfahren des unüberwachten Lernens, da die Trainingsdaten das Ergebnis nicht explizit enthalten, sondern es erst durch die Methode selbst bestimmt wird. Verfahren des bestärkenen Lernens schließlich zeichnen sich dadurch aus, dass sie überhaupt keine Trainingsdaten benötigen, sondern „learning by doing“ agieren. *Das* prototypische Beispiel für diese Art des Lernens sind Tiefe neuronale Netze wie AlphaZero.[8]

3.4.3 Einteilung nach Strukturerkennungsgrad

Neben den oben genannten Klassifikationsschemata können Verfahren des maschinellen Lernens auch nach der Art unterteilt werden, wie sie mit den Strukturen des jeweiligen Anwendungsproblem umgehen. In diesem Sinne wird häufig eine Einteilung in strukturentdeckende Methoden und strukturprüfende Verfahren vorgeschlagen.[9] Diese beiden Kriterien werden wie folgt verstanden:

Strukturprüfende Verfahren sind statistische Verfahren, um primär die kausale Abhängigkeit einer abhängigen Variablen von einer oder mehreren unabhängigen Variablen (Einflussfaktoren) zu überprüfen. *Strukturentdeckende Verfahren* dagegen sind statistische Verfahren, um Zusammenhänge zwischen Variablen oder Objekten entdecken. Eine Einteilung verschiedener statistischer Verfahren anhand ihrer Fähigkeit zur Strukturerkennung ist in Tabelle 3.6 aufgelistet.[10]

Strukturprüfend	Strukturentdeckend
Regressionsanalyse	Hauptkomponentenanalyse
Zeitreihenanalyse	Clusteranalyse
Diskriminanzanalyse	Faktorenanalyse
Logistische Regression	Neuronale Netze
Varianzanalyse	Soziale Netzwerkanalyse

Tabelle 3.6. Einteilung statistischer Verfahren nach Strukturerkennungsgrad.

Strukturprüfende Verfahren werden vorwiegend zur Analyse von Kausalitäten eingesetzt. Beispiele dafür sind Analysen, ob und wie stark sich das Wetter, die Bodenbeschaffenheit und unterschiedliche Düngemittel und -mengen auf den Ernteertrag auswirken, oder wie stark die Nachfrage eines Produktes von dessen Qualität, dem Preis, der Werbung und dem Einkommen der Konsumenten abhängt. Diese Verfahren basieren auf einem statistischen Modell, das a priori einen sachlogisch fundierten Kausalzusammenhang zwischen den Variablen annimmt, also eine Vorstellung davon, welche Variablen („Ursache“) welche anderen Variablen („Wirkung“) beeinflussen. Diejenigen Variablen, die die anderen beeinflussen, sind dann die unabhängigen Variablen nach Definition 3.3, die beeinflussten Variablen die abhängigen.

Auch den strukturentdeckenden Verfahren liegt jeweils ein statistisches Modell zugrunde, dessen Parameter anhand von Trainingsdaten eingestellt werden. Bei der Hauptkomponentenanalyse beispielsweise ist dies die Kovarianzmatrix der beobachteten Daten, deren Eigenwerte als Parameter zu bestimmen sind; bei der Clusteranalyse das Distanz-

[8] Silver et al. (2017).

[9] Backhaus et al. (2016):pp. 15; Backhaus et al. (2015); Ng und Soo (2018).

[10] Backhaus et al. (2016); Backhaus et al. (2015); Ng und Soo (2018).

oder Ähnlichkeitsmaß der Datenobjekte, anhand dem die Clusterzuordnungen minimiert werden; und bei einem neuronalen Netz ist es sein gewichteter Graph, dessen Kantengewichte als Parameter zu optimieren sind.

3.5 Probleme des maschinellen Lernens

Das maschinelle Lernen läuft nicht unbedingt wie ein Uhrwerk, das immer perfekte Ergebnisse liefert. Vielmehr hat es seine Tücken und Vorbehalte, aber auch einige ernsthafte Probleme. Zwei Hauptprobleme des maschinellen Lernens betreffen die Wahl des angewandten Modells, das andere die Qualität der Trainingsdaten.

3.5.1 Probleme der Wahl des Modells: Underfitting und Overfitting

Die grundlegenden Probleme der Wahl eines falschen Modells sind Unteranpassung *(Underfitting)* und Überanpassung *(Overfitting)*. Um diese Begriffe zu klären, erinnern wir uns zunächst an das grundlegende Ziel des maschinellen Lernens: *Maschinelles Lernen ist ein Prozess zur Anpassung eines Modells an bestimmte beobachtete oder anderweitig eingegebene Daten.* Welches Modell angepasst werden soll, hängt von der angewandten Methode ab, vgl. Tabelle 3.3. Zwei typische Klassen von Verfahren sind Regressionen und Klassifikationen. Betrachten wir Underfitting und Overfitting für diese beiden Fälle.

Bemerkung 3.8. Grob gesagt passt eine Regression die Parameter $\boldsymbol{\theta}$ einer gegebenen stetigen Funktion $f(\boldsymbol{x}, \boldsymbol{\theta})$ an die Daten $\boldsymbol{x}$ und y an, vgl. Beispiel 3.5. Andererseits passt eine Klassifikation ein Modell mit einer diskreten (d.h. nichtstetigen) Funktion $f(\boldsymbol{x}, \boldsymbol{\theta})$ an, die die Datenpunkte durch eine Qualität y trennt, vgl. Beispiel 3.7. Nehmen die beiden Datensätze an, die in der Abbildung 3.1 dargestellt sind.

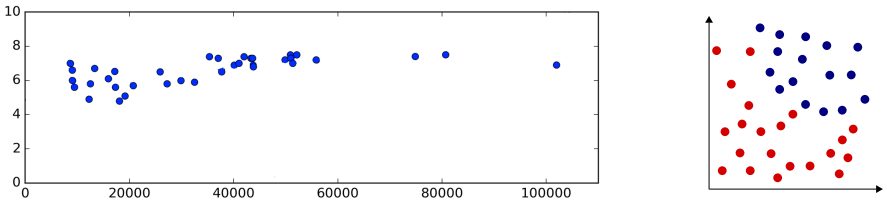


Abbildung 3.1. Given training data for a regression (left) and a classification (right).

Der erste entscheidende Schritt besteht darin, ein Modell zu wählen, das für die Daten geeignet ist. Sollten wir ein lineares Modell verwenden? In Abbildung 3.2 wird dieser Ansatz für den Fall einer Regression und einer Klassifizierung dargestellt.

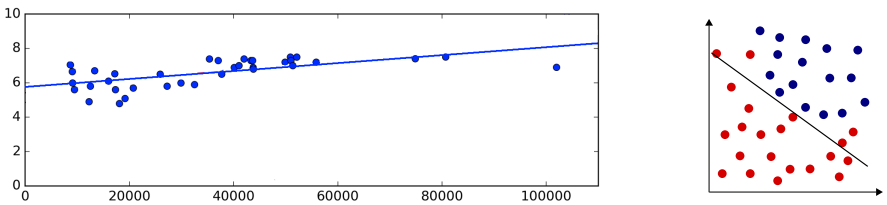


Abbildung 3.2. Lineare Modelle für Regression (links) und Klassifikation (rechts).

Oder sollten wir lieber ein nichtlineares Modell wie in Abbildung 3.3 wählen? Es gibt kein eindeutiges Kriterium für die Entscheidung, welches Modell verwendet werden sollte, um eine funktionale Beziehung zwischen den Daten $\boldsymbol{x}$ und y auszudrücken. Außerdem

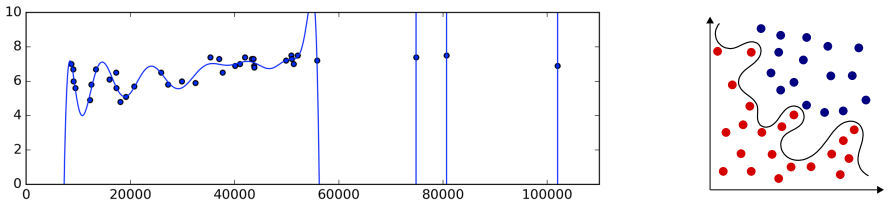


Abbildung 3.3. Nichtlineare Modelle für Regression (links) und Klassifikation (rechts).

ist nicht klar, wie viele Parameter $\boldsymbol{\theta}$ angegeben werden sollten. Das eigentliche Problem besteht darin, dass wir in der Regel das anzuwendende Modell nicht kennen. □

Die Probleme der Unteranpassung und Überanpassung sind in der Tabelle 3.7 zusammengefasst. Der Grund für diese beiden Phänomene ist derselbe, nämlich ein ungeeignetes Modell. Underfitting wird durch ein Modell verursacht, das zu einfach ist, um die wahre funktionale Beziehung darzustellen, während Overfitting durch ein Modell verursacht wird, das zu komplex ist, d. h. zu eng an die Trainingsdaten angepasst ist, so dass es nicht auf neue Daten verallgemeinert werden kann. Daher werden in beiden Fällen die

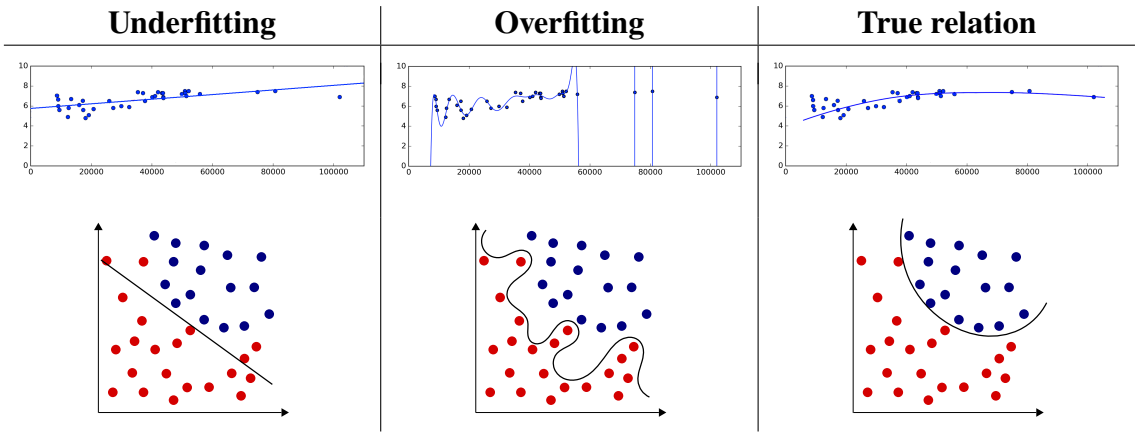


Tabelle 3.7. Underfitting und Overfitting von Regressionen und Klassifikationen.

statistischen oder kausalen Beziehungen nicht gut genug verstanden.

In der Praxis ist die Wahl eines Modells ein Kompromiss zwischen Modellkomplexität und Exaktheit, aber wir kennen in der Regel nicht den „Kipppunkt“. Das heißt, wir wissen in der Regel nicht: Wann genau ist das Optimum der Verallgemeinerung für ein möglichst einfaches Modell erreicht? In der Tat sollte die Wahl von *Ockhams Rasiermesser* geleitet werden, vgl. Abschnitt 2.4 auf S. 27: Von verschiedenen Modellen eines Phänomens ist das einfachste zu bevorzugen. Mit anderen Worten:

Ein „gutes“ Modell ist so einfach wie möglich und so komplex wie nötig.

Für Regression werden wir einige Maße kennenlernen, um verschiedene Modelle quantitativ zu vergleichen. Ein oft verwendetes Maß ist das BIC, das auf der Bayes'schen Version von Ockhams Rasiermesser in Gleichung (2.8) basiert. Wir werden es auf Seite 62 definieren.

3.5.2 Datenqualität

Ein weiteres Problem des maschinellen Lernens betrifft die Qualität der Trainingsdaten. Wir wissen in der Regel nicht, wie „gut“ oder „schlecht“ die von uns verwendeten Trainingsdaten sind. Schlechte Daten können durch folgende Probleme verursacht werden.

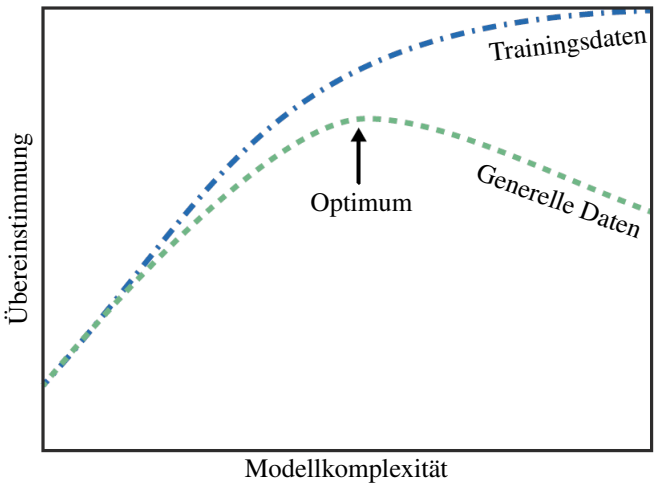


Abbildung 3.4. Die Wahl des besten Modells ist im Allgemeinen ein Kompromiss zwischen Modellkomplexität und Übereinstimmung mit den Trainingsdaten.

- Die Daten sind ungenau; dies kann auf ungenaue Messungen, missverständliche Fragebögen usw. zurückzuführen sein . . .
- Die Daten sind fehlerhaft oder falsch
- Die Daten sind nicht repräsentativ, „untypisch“.

Selbst wenn wir ein ideales Modell anwenden, lernt die Maschine mit falschen Daten das Falsche!

4
Kurzeinführung in Python

Kapitelübersicht

4.1 Grundlegende Sprachelemente 45
4.1.1 Ein- und Ausgaben 45
4.1.2 Elementare Operationen 46
4.2 Kontrollstrukturen 47
4.2.1 Bedingte Anweisungen mit `if` 47
4.2.2 Schleifen 48
4.2.3 Tiefe Schleifen 49
4.2.4 Funktionen 49
4.3 Bibliotheken und Module 50
4.3.1 Pandas 52
4.3.2 Scikit-Learn 54
4.3.3 Statsmodels 55
4.4 Übungsaufgaben 55

In diesem Skript werden wir viel mit Hilfe von Programmen der Programmiersprache Python arbeiten. Zu Python wird hier daher eine kurze Einführung gegeben.

Python ist eine dynamisch typisierte, objektorientierte Skriptsprache mit Unterstützung funktionaler Programmierung. Sie wurde 1991 von Guido van Rossum in Amsterdam entwickelt und nach den in den 1970er Jahren populären britischen Komikern Monty Python benannt. Die API Dokumentation befindet sich online unter

`https://docs.python.org/`

Eine Kurzanweisung, um auf verschiedenen Plattformen Python und die für diesen Lehrbrief wesentlichen Pakete zu installieren, findet sich auf meiner Webseite

`https://math-it.org/AI/installations.html`

Hier werden die Schritte zur Installation des für die Programmierung mit Python häufig verwendeten *Jupyter Notebook* angegeben, das interaktiv im Browser die Eingabe von Quelltext und dessen Ausführung ermöglicht.

4.1 Grundlegende Sprachelemente

Die reservierten Wörter und grundlegende Datentypen von Python sind in Abbildung 4.1 dargestellt. Python hat also mehrere grundlegende Datentypen wie Strings (`"..."` oder

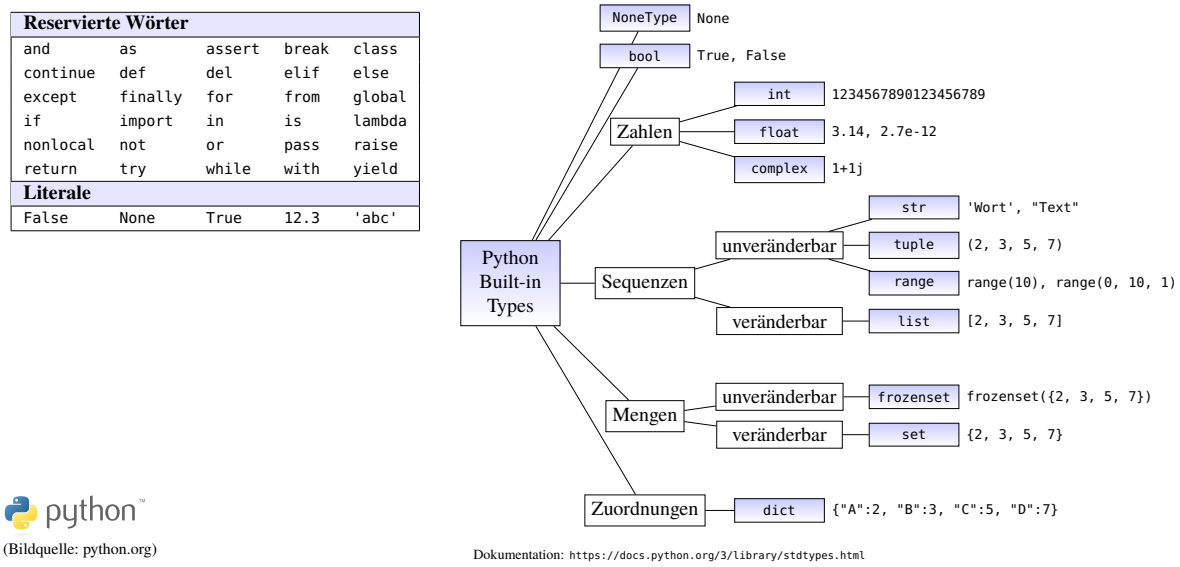


Abbildung 4.1. Reservierte Wörter und Standard-Datentypen von Python

`'...'`), Boole'sche Werte (`True`, `False`), beliebig große Ganzzahlen, Gleitkommazahlen und komplexe Zahlen, Listen `[..., ...]`. Anweisungen werden in Python durch einen Zeilenumbruch beendet. Blöcke von Anweisungen werden durch Einrückungen gebildet. Zeilenweise Kommentare werden durch `#` markiert. Der Zuweisungsoperator `=` bestimmt sowohl Wert als auch Typ einer Variablen:

```
a = 1                  # Zuweisung eines Ganzzahlwerts
a = 1.2                # Zuweisung eines Gleitkommawerts
a = 1 + 5j             # Zuweisung eines komplexen Werts
a = 'ein String'       # Zuweisung eines Strings
a = "auch " + a        # Zuweisung eines konkatenierten Strings
a = [2, 3, 5, 7]       # Zuweisung einer Liste
```

4.1.1 Ein- und Ausgaben

Die einfachsten Möglichkeiten zur Ein und Ausgabe in Python sind die Funktionen **`input`** und **`print`**. Ein String oder der Wert einer Variablen `a` wird mit dem Befehl

```
print(a)
```

ausgegeben, mehrere auszugebende Werte können mit Kommas getrennt werden:

```
print('a =', a)
```

Zahlen können als f-Strings mit dem Präfix `f` oder `F` formatiert ausgegeben werden:

```
print(f'{314.1592:.2f}')  # => 314.16
print(F"{31415.92:,.0f}") # => 31,416 mit Tausendertrenner
print(f'{0.314159:.2%}')  # => 31.42%
```

Direkt hinter dem Formatierungspräfix muss ein String mit einer geschweiften Klammer, dem zu formatierenden Wert, einem Doppelpunkt, der Formatspezifikation und einer

geschlossenen geschweiften Klammer folgen: `f"{`*wert*`:`*spec*`}"`. Mit der Spezifikation „`.2f`“ wird der Wert als Kommazahl mit zwei Nachkommastellen gerundet dargestellt, mit „`,.0f`“ (d.h. mit Komma vor dem Punkt!) wird er entsprechend auf 0 Nachkommastellen gerundet und zusätzlich mit einem Komma als Tausendertrenner für Zahlen größer gleich 1000 versehen.

Eingaben von der Tastatur können mit **`input`** eingelesen werden:

```
eingabe = input("Gib einen Text ein: ")
```

Es erscheint ein Textfeld, in das Text eingegeben und durch Drücken der Returntaste in der Variablen `eingabe` gespeichert werden kann.

4.1.2 Elementare Operationen

Wie in den meisten Programmiersprachen ermöglicht Python die Grundrechenarten mit einfachen Operatoren. Die Stringkonkatenation und gängige arithmetische Operationen in Python sind entsprechend in Tabelle 4.1 aufgeführt. Besonderheiten von Python sind die

Operator	Bedeutung	Beispiel	Ergebnis
`+`	Addition, Vorzeichen	`2 + 4, +1`	`6, 1`
`+`	Konkatenation (Strings)	`"ab"+"c", '12'+'3'`	`'abc', '123'`
`-`	Subtraktion, Vorzeichen	`2 - 4, -1`	`-2, -1`
`*`	Multiplikation	`2 * 4`	`8`
`/`	Division	`7 / 2`	`3.5`
`//`	ganzzahlige Division	`7 // 2`	`3`
`%`	modulo (Divisionsrest)	`7 % 2`	`1`
`**`	Potenz	`7 ** 2`	`49`

Tabelle 4.1. Konkatenation und arithmetische Operationen in Python

beiden Möglichkeiten zur Division zweier Zahlen, nämlich die stets reellwertige Division mit `/` nach IEEE 754) und die ganzzahlige Division mit `//`.

Auch logische (oder Boole'sche) Operatoren für NOT, UND sowie ODER gibt es in Python, sie sind in Tabelle 4.2 aufgeführt. Hierbei erhält die Verneinung nur einen

Operator	Bedeutung	Beispiel	Ergebnis
`not`	Verneinung	`not True, not False`	`True, False`
`and`	logisches UND	`True and False`	`False`
`or`	logisches ODER	`True or False`	`True`

Tabelle 4.2. Logische Operationen in Python

Boole'schen Wert **`True`** oder **`False`** und ergibt stets den jeweils anderen Wert davon. Die logischen Operatoren **`and`** und **`or`** dagegen verknüpfen stets zwei Boole'sche Werte, wobei **`and`** nur dann **`True`**, wenn beide Werte **`True`** sind, und umgekehrt **`or`** nur dann **`False`**, wenn beide Werte **`False`** sind.

Wichtige Standardfunktionen in Python sind in Tabelle 4.3 aufgelistet. Sie werden öfter in diesem Skript verwendet, insbesondere `len()` und `range()`. Die Funktion `len` erwartet eine Sequenz und gibt ihre Länge als Wert zurück. Die Funktion `range`$(x_0, x_{\max}, \Delta)$ erstellt eine Zahlenfolge (eigentlich einen „Iterator“)

$$x_0 \to x_0 + \Delta \to x_0 + 2\Delta \to \ldots \to x_0 + n\Delta,$$

so dass das letzte Folgenglied echt kleiner $x_{\max}$ ist. Alle Parameter müssen ganzzahlig sein. Wird der dritte Parameter weggelassen, ist die Schrittweite 1, wird auch der zweite

Funktion	Bedeutung	Beispiel	Ergebnis
`int()`	Konvertierung in eine ganze Zahl	`int(1.2)`, `int("123")`	1, 123
`float()`	Konvertierung in eine Gleitkommazahl	`float(12)`, `int("1E-12")`	12.0, 1e-12
`str()`	Konvertierung in einen String	`str(1.2)`, `str("123")`	"1.2", "123"
`len()`	Länge einer Sequenz oder Menge	`len([2, 3, 5])`	3
`sum()`	Summe einer Sequenz oder Menge	`sum([2, 3, 5])`	10
`list()`	Sequenz oder Menge als Liste	`list({2, 3, 5})`	[2,3,5]
`range()`	Erzeugen einer Range	`range(3,18,5)`	3⟶8⟶13
`round()`	Runden einer Gleitkommazahl	`round(3.1415,2)`	3.14

Tabelle 4.3. Wichtige Standardfunktionen in Python

weggelassen, so geht die Folge von 0 bis $x_0 - 1$. Will man die Folge als Liste haben, so kann man dies mit

```
list(range(4))  # ergibt [0,1,2,3]
```

erreichen. Ähnlich erhält man mit

```
sum(range(1,9,2))  # => 1 + 3 + 5 + 7 = 16
```

die Summe aller Zahlen der Folge.

4.2 Kontrollstrukturen

Python besitzt die in einer imperativen Programmmiersprachen üblichen Kontrollstrukturen, die den Ablauf der einzelnen Anweisungen steuern. Es sind dies die bedingte Anweisung, Schleifen und Subroutinen, in Python als Funktionen programmierbar.

4.2.1 Bedingte Anweisungen mit `if`

Die bedingte Ausführung eines Blocks von Anweisungen wird mit **`if`** programmiert, eine (optionale) Alternative mit **`else`**:

```
x = 6
if x < 0:
    print('negativ')
else:
    print('nicht-negativ')
```

Der Ausdruck hinter `if` ist eine Bedingung und daher entweder `True` oder `False`. Umfasst der if-Zweig nur eine einzelne Anweisung, so kann sie direkt nach dem Doppelpunkt geschrieben werden, mehrere Anweisungen müssen eingerückt werden, um einen Block zu bilden. Als Bedingungen sind z.B. folgende Vergleichsoperatoren für Zahlen möglich:

gleich: `a == b`	kleiner als: `a < b`	größer als: `a > b`
ungleich: `a != b`	kleiner gleich: `a <= b`	größer gleich: `a >= b`

Eine mehrfache Fallunterscheidung, also eine Auswahl aus mehreren Optionen, kann durch eine „Else-if-Leiter“ mit `elif` („else if“) programmiert werden:

```
monat = 4
if monat in (1, 3, 5, 7, 8, 10, 12):
    tage = 31
elif monat in (4, 6, 9, 11):
```

```
    tage = 30
else:  # bleibt nur noch: monat == 2
    tage = 28
print("Monat ", monat, " hat ", tage, " Tage")
```

Soll durch eine einfache Fallunterscheidung ein einzelner Wert ermittelt werden, so ermöglicht Python noch die Möglichkeit eines einzelnen bedingten Ausdruck mit nachgestelltem if – else:

```
x if bedingung else y
```

Abhängig von der Bedingung ergibt dieser Ausdruck entweder den Wert `x` oder den Wert `y`. Er kann in einer Variablen gespeichert oder auch mit `print` direkt ausgegeben werden. So wird beispielsweise mit den Anweisungen

```
from time import localtime
print("Guten " + ("Morgen" if localtime().tm_hour < 12 else "Tag"))
```

ein zur Tageszeit passender Gruß ausgegeben (. . . zumindest zwischen 2 und 18 Uhr).

4.2.2 Schleifen

In Python gibt es zwei Arten von Schleifen, mit denen sich wiederholende Anweisungsblöcke programmiert werden können:

listengesteuerte Schleife mit `for`

- Die listengesteuerte Schleife *(loop)*:

  ```
  for i in range(3):  # [start = 0,] end = 3-1 = 2
     print(i)         # -> 0, 1, 2
  ```

  Merkmal: Die Anzahl der Schleifendurchläufe steht schon beim Start fest.

While-Schleife

- Die While-Schleife:

  ```
  noch_nicht_erraten = True   # Boole'sche Variable
  while noch_nicht_erraten:   # wiederhole solange True ...
      tipp = input("Rate die Zahl: ")
      noch_nicht_erraten = (tipp != "7")
  ```

  Merkmal: Die Anzahl der Schleifendurchläufe muss beim Start nicht unbedingt bekannt sein.

Jede Schleife kann in Python durch die Anweisung **`break`** abgebrochen werden.

```
n = int(input("Gib eine ganze Zahl > 2 ein: "))
for i in range(2,n):
    if n % i == 0:  break   # i teilt n: Abbruch der Schleife!
if i == n - 1: print (n, " ist prim")
else: print (n, " ist nicht prim: ", n, "/", i, "=", n//i)
```

Will man in Python durch eine gegebene Liste `[...]` iterieren und aus ihren Elementen eine neue erzeugen, so kann die Konstruktion

```
neue_liste = [ausdruck for eintrag in liste]
```

verwendet werden, die *List Comprehension*. Die Einträge der neuen Liste werden dann in `ausdruck` berechnet. Ein Beispiel:

```
liste = [i**2 for i in [2,3,5,7,11]]  # => [4,9,25,49,121]
```

Man kann mit einem `if`-Ausdruck vor der äußeren schließenden Klammer auch eine Filterbedingung aufnehmen:

```
liste = [x for x in range(30) if x % 7 == 0]  # => [7,14,21,28]
```

4.2.3 Tiefe Schleifen

Schleifen kann man verschachteln, um damit z.B. zweidimensionale Datenstrukturen wie Matrizen zu durchlaufen:

```
for x in range(4):
    for y in range(5):
        print((x,y), end=" ")  # Leerzeichen statt Zeilenumbruch
    print()                    # drucke Zeilenumbruch
```

ergibt die Ausgabe:

```
(0, 0) (0, 1) (0, 2) (0, 3) (0, 4)
(1, 0) (1, 1) (1, 2) (1, 3) (1, 4)
(2, 0) (2, 1) (2, 2) (2, 3) (2, 4)
(3, 0) (3, 1) (3, 2) (3, 3) (3, 4)
```

Wir können mit einer tiefen Schleife auch Wertetabellen von Funktionen mit mehreren Variablen ausgeben:

```
print("a b c\t (a or b) and c")
for a in range(2):
    for b in range(2):
        for c in range(2):
            # 3 Boole'sche Variablen und ein logischer Ausdruck:
            print(a, b, c, "\t   ", int( (a or b) and c ))
```

Das ergibt die Wahrheitstabelle des logischen Ausdrucks $(a \vee b) \wedge c$. (Probieren Sie es aus!):

```
a b c    (a or b) and c
0 0 0       0
0 0 1       0
0 1 0       0
0 1 1       1
1 0 0       0
1 0 1       1
1 1 0       0
1 1 1       1
```

4.2.4 Funktionen

Funktionen werden in Python mit dem reservierten Wort **`def`** und einem Doppelpunkt definiert und mit ihrem Namen und eingesetzten Werten aufgerufen. Z.B. definieren wir die Funktion $f(x, y) = x^2 + y^2$ durch:

```
def f(x,y):
    return x**2 + y**2
f(3,4)  # -> 25
```

Die Variablen einer Funktion heißen *Parameter*, die beim Aufruf eingesetzten Werte *Argumente*. Man kann Parametern voreingestellte Werte (*Default-Werte*) geben, die verwendet werden, wenn sie beim Aufruf keinen Wert bekommen:

```
def f(x=3,y=4):
   return x**2 + y**2

f(), f(9, 12), f(y=0)   # -> (25, 225, 9)
```

starred expression * entpackt Parameterliste

Eine Besonderheit von Python ist, dass eine Liste als letzter Parameter einer Funktion als „gesternter Ausdruck“ *(starred expression)* mit dem „Splat-Operator“ entpackt werden kann:

```
def f(x, y):
    return (x**2 + y**2)**0.5

lst = (3, 4)
f(*lst) # -> 5
```

Hier wird also das Tupel `(3,4)` in die zwei separaten Parameter 3 und 4 zerlegt. Der Ausdruck funktioniert jedoch ausschließlich in Funktionsaufrufen.

In Python kann eine Funktion auch innerhalb einer Funktion definiert und damit als Variable zurückgegeben werden:

```
def f(x):
    def g(y): return x**y
    return g
f(2)(3)    # => 8
```

Kürzer können wir die innere Funktion als einen *Lambda-Ausdruck* definieren, also als eine anonyme Funktion:

```
def f(x):
    return lambda y : x**y
f(2)(3)    # => 8
```

4.3 Bibliotheken und Module

Modul
Library, Package

Neben den eingebauten Sprachelementen der Standardbibliothek von Python gibt es für spezielle Zwecke zahlreiche Module, die zu Bibliotheken („Libraries“ oder „Packages“) zusammengefasst werden können und spezielle Funktionen oder Objekte bereitstellen.[1] Sie können an beliebiger Stelle in einem Programm importiert werden.

Die in diesem Skript verwendeten Python-Bibliotheken und Module werden kurz in diesem Abschnitt beschrieben. Für Hinweise zur vollständigen Installation siehe auch unter `http://haegar.fh-swf.de/KI/Installationen.html`.

NumPy
Matplotlib

In dem folgenden Beispielprogramm werden zwei sehr häufig verwendete Bibliotheken importiert, NumPy (sprich: „nampei“) für numerische Berechnungen und Arrays (die es in Python als Standard-Datentyp ja nicht gibt, vgl. Abb. 4.1), sowie Matplotlib zur

[1] `https://docs.python.org/3/library/`

Visualisierung, speziell zur Ausgabe von Funktionsgraphen und Diagrammen. Meist wird dabei aber gar nicht die gesamte Bibliothek Matplotlib gebraucht, sondern nur das Modul pyplot:

```
import numpy as np              # Modul für numerische Berechnungen
import matplotlib.pyplot as plt # Modul für Funktionsgraphen

x = np.arange(0, 6*np.pi, 0.1)  # Array [0, 0.1, ..., 6*pi]
plt.plot(x, np.sin(x))          # Funktionsgraph (x, sin x)
plt.show()                      # schließt u. zeigt alle plt-Aktionen
```

Das Programm erstellt zunächst mit der Funktion `np.arange` ein Array `x` von 0 bis 6π (ausschließlich) in 0,1-Schritten; `np.arange` ist damit die „Array"-Variante der Python-Standardfunktion **`range`**, die nur ganzzahlige Werte annehmen kann. In der nächsten Anweisung werden die Werte des Arrays `x` gegen das Array `np.sin(x)` ihrer Sinuswerte aufgetragen. Das ist eine sehr bemerkenswerte Eigenschaft vieler NumPy-Funktionen: Wird ihr eine einzelne Zahl übergeben, so ergibt sie wieder eine Zahl; wird ihr jedoch ein Array übergeben, so werden die Funktionswerte der einzelnen Arraywerte berechnet und in einem Array gleicher Größe als Ergebnis zurückgegeben! Die Standardvariante von `plt.plot` stellt die beiden übergebenen Arrays als zweidimensionalen Funktionsgraphen dar, dessen Punkte durch eine Linie verbunden werden. Mit `plt.show()` werden alle bisherigen plt-Aktionen (hier nur eine) abgeschlossen, zusammengefasst und ausgegeben. Das ergibt in diesem Programm den in Abbildung 4.2 dargestellten Funktionsgraphen der Sinusfunktion. In dem Quelltext ist erkennbar, dass man mit dem reservierten Wort **`as`**

Arrays als Argument einer Funktion

`plt.plot`

`plt.show`

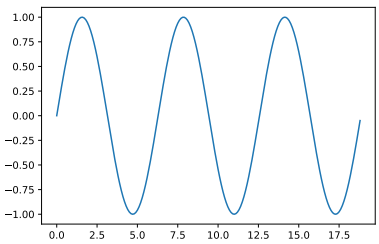


Abbildung 4.2. Der Funktionsgraph der Sinusfuktion mit dem Modul `matplotlib`

dem Modul einen Aliasnamen geben kann, den man statt des Modulnamens verwendet.

Aliasnamen mit **`as`**

`from`

In dem folgenden Beispielprogramm wird in eine Variante der **`import`**-Anweisung nur ein einzelnes Modul oder Bibliothekselemente mit **`from`** importiert und verwendbar:

```
%matplotlib inline
from numpy import pi, linspace, cos
import matplotlib.pyplot as plt

x = linspace(0, 6*pi, 100) # Array [0, 0.1, ..., 6*pi]
plt.plot(x, cos(x))        # Funktionsgraph (x, cos x)
plt.savefig("cos.png")     # speichert die Graphik als png
plt.show()                 # schließt u. zeigt alle plt-Aktionen
```

Allerdings sind nach einem Import von Elementen aus einer Bibliothek auch nur diese Elemente daraus verfügbar. Das Beispielprogramm gibt den Funktionsgraph der Cosinus-Funktion aus Abbildung 4.3 wieder. Aber es gibt in diesem Programm auch weitere Varianten zu dem vorherigen. So wird statt `np.arange` hier die ganz ähnliche Funktion `np.linspace` eingesetzt. Sie erwartet als Parameter Start- und Endwert des Arrays und

`np.linspace`

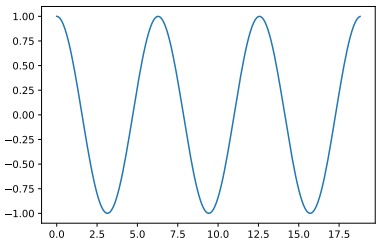


Abbildung 4.3. Funktionsgraph der Cosinusfunktion

die Anzahl der Einträge. Bei `np.linspace` steht also die Anzahl der Einträge im Vordergrund, bei `np.arange` die Schrittweite zwischen den Einträgen. In der vorletzten Zeile des Quelltextes wird mit `plt.savefig("cos.png")` die Grafik in einer Datei `cos.png` im Wurzelverzeichnis des Jupyter Notebooks gespeichert. Matplotlib erkennt automatisch an der Endung, welches Format zu verwenden ist, hier ist es PNG, möglich sind aber auch PDF, SVG, TIFF oder JPG.

`plt.savefig`

Anmerkung: Arbeitet man mit dem Jupyter Notebook, so bewirkt die Anweisung `%matplotlib inline` in der ersten Zeile eines Python-Programms, dass es die Grafik auch beim ersten Durchlauf des Programms anzeigt.

Bemerkung 4.1. Es gibt über Python einige gute Bücher. Eines, dass mir für die grundsätzlichen Sprachelemente immer wieder als Nachschlagewerk dient, ist[2]. Da sich bei Python aber von Anfang an, bedingt gerade durch die offene Architektur der Sprache, eine sehr rege Gemeinschaft von Entwicklerinnen und Entwicklern für neue Bibliotheken und Module gebildet hat, sind Bücher in diesen Punkten oft schnell nicht mehr aktuell. Die nach meiner Ansicht beste Möglichkeit, bei Python & Co auf dem Laufenden zu bleiben, sind Internetquellen, und hier insbesondere die *Scipy Lecture Notes* [SciPy], deren Link am Ende dieses Skripts aufgeführt ist. Sie enthalten gute Einführung in NumPy [NumPy] und in Matplotlib [plt]. □

4.3.1 Pandas

Als eine Art Basismodul für maschinelles Lernen in Python hat sich Pandas etabliert, `https://pandas.pydata.org/`. Es bietet mächtige und flexible Methoden zum Importieren, Transformieren und einfache Analysen externer Daten. Für unsere Zwecke besonders wichtig sind die unter `https://pandas.pydata.org/docs/user_guide/` aufgelisteten Importfunktionen für zahlreiche Datenformate, insbesondere textbasierte CSV-Dateien (mit Endung `.csv`, manchmal auch `.xls`), oder binäre Dateiformate wie Excel, OpenDocument Spreadsheets, oder SPSS.

DataFrames und Series. Die beiden zentralen Datenstrukturen in Pandas sind das DataFrame und die Series („Reihe“). Ein DataFrame ist eine zweidimensionale Datenstruktur, also eine Tabelle, in der die Daten als Spalten (`column`) organisiert sind und jede Zeile durch einen Index identifiziert wird.

Index	**Spaltenname 1**	**Spaltenname 2**	$\cdots$
i_0	x_{01}	x_{02}	$\cdots$
i_1	x_{11}	x_{12}	$\cdots$
$\vdots$	$\vdots$	$\vdots$	

[2] Weigend (2019).

Jede Spalte hat einen Namen, mit dem sie wie bei einer Liste mit eckigen Klammern adressiert werden kann:

```
df['spaltenname']
```

Die Indexwerte können natürliche Zahlen, Strings oder Zeiten (Datetime-Objekte) sein. Entsprechend erhält man einen einzelnen Wert, indem sein Indexwert mit einer zweiten eckigen Klammer adressiert wird:

```
df['spaltenname']['indexwert']
```

Den Namen der Zeile Nummer i erhält man mit `df.index[i]`, die Namen der Spalte Nummer j mit `df.columns[j]`.

Eine *Series („Reihe“)* ist eine eindimensionale Struktur, die in etwa einem DataFrame mit nur einer Spalte entspricht. Insbesondere hat auch sie einen Index. Wir werden Series in der dritten Lehreinheit dieses Skripts zur Speicherung von Zeitreihen verwenden.

Um Werte eines Dataframes `df` als numerisches Array zu erhalten, muss man die gewünschte Spalte (DataFrame oder Series) dem Numpy-Objekt `n_c` übergeben:

```
import numpy as np
X = np.c_[df['spaltenname']]
```

Das Objekt `c_` in Numpy erwartet eine Liste von Einträgen und erzeugt daraus eine numerische Spaltenmatrix (daher `c_` wie “column”). Beachte, dass `c_` direkt eine *eckige* Klammer erwartet, also nicht wie eine Funktion eine runde Klammer. Oft werden wir die Spaltenwerte eines Pandas-Objekts nicht als numerische Spaltenmatrix `array[[...]]` benötigen, sondern als eindimensionales Array. Dafür ist die Numpy-Funktion `ravel` hilfreich, die aus den Einträgen einer Matrix, aber auch eines Tensors höherer Stufe, ein eindimensionales Array erzeugt. Mit

```
X = np.array([[1, 2, 3], [4, 5, 6]])
x = np.ravel(X)
```

beispielsweise wird eine Matrix X und ein Array x erstellt:

```
X = [[1 2 3]
     [4 5 6]]          x = [1 2 3 4 5 6]
```

Beispiel 4.2. Im maschinellen Lernen und der Datenanalyse werden wir fast ausschließlich mit externen Daten arbeiten. Das folgende typische Fallbeispiel soll die ersten Schritte dafür zeigen. Gegeben sei eine Datei `children.csv` mit dem Inhalt

```
Größe;Alter
107.9;5
149.8;12
85.5;2
115.5;6
138.4;10
```

deren Spalten durch ein Semikolon getrennt sind. Um sie ins Verzeichnis `datasets` zu importieren, genügen die Anweisungen

```
import pandas as pd
df = pd.read_csv("./datasets/kinder.csv", sep=";")
```

Zu beachten ist dabei, dass sich der Ordner hier im Wurzelverzeichnis des Jupyter Notebooks befindet. Dann ist der gesamte Inhalt als DataFrame in der Variable `df` gespeichert. Mit `print(df)` kann man deren Inhalt kontrollieren:

```
   Größe  Alter
0  107.9      5
1  149.8     12
2   85.5      2
3  115.5      6
4  138.4     10
```

Der Parameter `sep` ist optional, der Standardwert ist `','` für das Komma als Spaltentrenner. Wichtig ist oft auch der optionale Parameter `encoding` für Textcodierung, hier ist der Standardwert `'UTF-8'`. Bei Windowsdateien ist sie häufig `'ISO-8859-1'`. □

Pandas DataFrames und Series stellen auch komfortable Funktionen zum Plotten ihrer Daten bereit. Mit dem optionalen Parameter `kind` kann man die Art des Plots festlegen, sein Standardwert ist `'line'`. Beispielsweise ergibt die Anweisung

```
df.plot(kind='bar')
```

den Graph in Abbildung 4.4. In der Standardvariante der Funktion für DataFrames wird eine Legende mit den Spaltennamen angezeigt. Will man sie nicht anzeigen lassen, übergibt

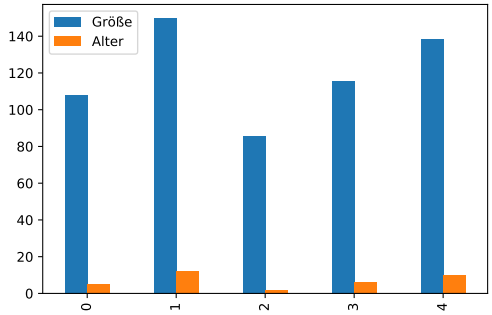


Abbildung 4.4. Graph der Anweisung `df.plot(kind="bar")`

man als Parameter `legend=None`. Weitere mögliche Darstellungsarten sind unter `https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html#other-plots` aufgeführt.

4.3.2 Scikit-Learn

Scikit-Learn ist eine Bibliothek für Python für maschinelles Lernen und wird importiert als `sklearn`. Die Bibliothek stellt sehr viele Modelle zur Datenanalyse als Klassen (auch „Estimators“, also Schätzer, genannt) zur Verfügung. Die meisten davon stellen dieselben Funktionen und dieselbe Syntax zur Festlegung und zur Anpassung der Modellparameter bereit, so dass sie zum großen Teil einheitlich und weitgehend austauschbar anwendbar sind. Wichtige Klassen von Scikit-Learn sind in Tabelle 6.1 aufgeführt. Die Modelle reichen von Regression über Klassifikation zur Clusterung, aber es gibt auch Klassen zur Dimensionsreduktion unf Funktionen zur Modellauswahl und zur Datenvorverarbeitung. Scikit-Learn ist umfangreich dokumentiert unter `http://scikit-learn.org`.

4.3.3 Statsmodels

Das Python-Modul `statsmodels` ermöglicht statistische und ökonometrische Analysen. Zur Zeitreihenanalyse eignet es sich besser als Scikit-Learn. Die API Dokumentation ist zu finden unter: `http://www.statsmodels.org/`. Zur Zeitreihenanalyse wird die Bibliothek `statsmodel.tsa.statespace` für nichtperiodische und periodische Prozesse bereitgestellt, insbesondere das Modell SARIMAX. Wir werden uns mit Statsmodels eingehend in Teil III dieses Skripts im Zusammenhang mit dem Thema Zeitreihenanalyse beschäftigen. Weitere Informationen sind unter `https://www.statsmodels.org/stable/statespace.html` zu finden.

4.4 Übungsaufgaben

Aufgabe 4.1 (**Literale in Python**)**.** Welche der folgenden Zeichenfolgen sind gültige Python-Literale?

Literal	Gültig	Ergebnis / Erläuterung
`1.000e-0.2`		
`2e+1j`		
`0x567`		
`00x567`		
`0o567`		
`0o568`		
`'Größe'`		
`''Größe''`		
`"Größe"`		
`b'Größe'`		
`Komm 'rein!`		
`00023e001`		
`(1; 2; 3)`		

Füllen Sie die Tabelle zunächst nach Ihren Überlegungen aus, bevor Sie Ihre Vermutungen mit Python testen. Die letzte Spalte soll dabei das Ergebnis der Zeichenfolge enthalten, wenn sie ein gültiges Literal ist, und stets eine kurze Erläuterung.

Aufgabe 4.2 (**Datentypen in Python**)**.** Allgemein müssen in einem Computerprogramm konkrete oder gedankliche Objekte der Wirklichkeit durch geeignete Datentypen einer Programmiersprache repräsentiert werden. Geben Sie in der folgenden Tabelle zu jedem Objekt der Wirklichkeit einen passenden Datentyp in Python an.

Objekt der Wirklichkeit	Datentyp	Beispielliteral
Radius von Atomen	`float`	`3.2e-13`
Blumenname		
Namen der Teilnehmer eines Wettlaufs		
Spielstand eines Fußballspiels (z.B. 3:1)		
Name, Vorname und Alter einer Person		
Name, Vorname und Alter der Teilnehmer eines Wettlaufs		
Tabelle, in der zu chemischen Elementensymbolen die englischen und deutschen Namen gespeichert sind (z.B. H → hydrogen, Wasserstoff)		

Aufgabe 4.3 (**Wörter eines Alphabets**)**.** Schreiben Sie ein Python-Programm, das alle zweibuchstabigen Wörter eines gegebenen Alphabets am Bildschirn ausgibt. Das Alphabet soll hier als String gegeben sein, z.B. `alphabet = '01'` mit zwei Symbolen („Buchstaben“) oder `alphabet = 'abc'` mit drei Symbolen.

Aufgabe 4.4 (**Brute-Force-Algorithmus der DNA**)**.** Die DNA, Träger der Erbinformation von Lebewesen und einigen Viren, ist eine Nukleinsäure und enthält die Basen Adenin (A), Guanin (G), Cytosin (C) und Thymin (T). Sie ist ein Riesenmolekül in Form einer Doppelhelix und besteht im Wesentlichen aus einer Sequenz der vier Basenpaare AT, TA, GC und CG.

(a) Wie viele Kombinationen gibt es theoretisch, eine Sequenz aus vier dieser Basenpaare zu bilden? (Ein Beispiel einer solchen Kombination ist AT AT GC TA.)

(b) Schreiben Sie ein Python-Programm, das alle denkbaren Kombinationen aus vier Basenpaaren der DNA ausgibt.

Aufgabe 4.5 (**Multiplikationstrainer für das kleine Einmaleins**)**.** Programmieren Sie mit Python einen Multiplikationstrainer für das kleine Einmaleins. Dem Anwender sollen nacheinander insgesamt fünf Rechenaufgaben gestellt werden, nach dem Muster `9*8 =` ☐, die mit einer Eingabe des Ergebnisses zu beantworten ist. Ist das eingegebene Ergebnis falsch, gibt es eine entsprechende Rückmeldung, und es wird eine neue Antwort zu dieser Aufgabe erwartet. Die nächste Rechenaufgabe wird erst gestellt, wenn zuvor das richtige Ergebnis eingegeben worden ist. Nach fünf richtigen Eingaben wird dem Benutzer mitgeteilt, wie viel Sekunden er insgesamt benötigt hat.

Hinweise: In Python wird die Systemzeit (in Sekunden seit dem 1.1.1970 0:00 Uhr) mit der Funktion `time()` des Moduls `time` ermittelt. Eine Zufallszahl wird mit der Funktion `randint` aus dem Modul `numpy.random` erzeugt. Für die Syntax konsultieren Sie bitte die API.

Aufgabe 4.6 (Fehlerbalken mit NumPy). (a) Die Funktion `randn` des Moduls `numpy.random` hat etwas mit standardnormalverteilten Zufallszahlen $\varepsilon \sim N(0,1)$ zu tun. Was geben `randn(3)` und `randn(2,3)` aber genau aus?

(b) Welche Anweisung muss man programmieren, um mit `randn` normalverteilte Zufallszahlen $\varepsilon \sim N(\mu, \sigma)$ mit einem Mittelwert μ und der Standardverteilung σ zu erhalten?

(c) Die kosmische Hintergrundstrahlung ist eine das gesamte Weltall durchströmende Radiostrahlung im Mikrowellenbereich, mit einer Frequenz um $f = 160$ GHz bzw. einer Wellenlänge um $\lambda = c/f = 1{,}7$ mm. Sie kann nur erklärt werden als „Nachglühen" des Photonenblitzes des Urknalls, der sich inzwischen sehr abgeschwächt und durch die Ausdehnung des Weltalls in den langwelligen Radiobereich verschoben hat. Anfang der

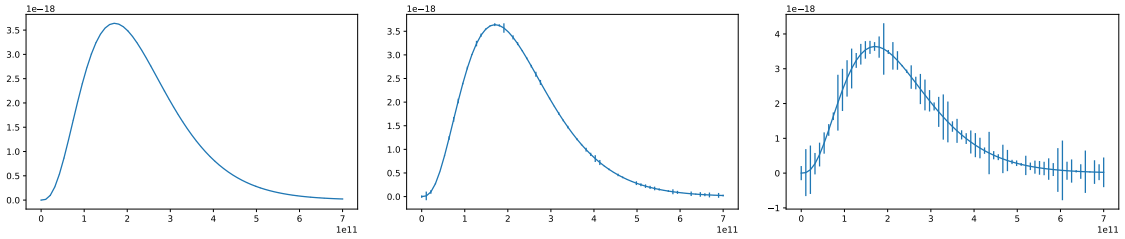


Abbildung 4.5. Das Spektrum der kosmischen Hintergrundstrahlung für $T = 2{,}728$ K mit Fehlerbalken der Breite $\sigma = 0{,}1\%$ von $y_{\max}$, $\sigma = 1\%$ von $y_{\max}$ und $\sigma = 10\%$ von $y_{\max}$, wobei $y_{\max}$ der Maximalwert der Strahldichte ist.

1990er Jahre vermaß der COBE-Satellit in einer Erdumlaufbahn die Intensität der Hintergrundstrahlung. Die Messergebnisse bestätigten die vorhergesagte Planck'sche Strahlungsverteilung bis auf 0,01% Genauigkeit für den Temperaturwert $T = 2{,}728$ K, vgl. Abbildung 4.5 links. Diese Verteilungsfunktion ist gegeben durch

$$B(f,T) = \frac{2hf^3}{c^2\left(\exp\left(\frac{hf}{kT}\right) - 1\right)} \tag{4.1}$$

mit den Konstanten c für die Lichtgeschwindigkeit, h für das Planck'sche Wirkungsquantum und k für die Boltzmann-Konstante[3]. Um zu verdeutlichen, wie verblüffend gering dieser Ungenauigkeit ist, erstellen Sie ein Python-Programm, in dem die Planck'sche Verteilung als Python-Funktion $B(f,T)$ implementiert und für $T = 2{,}728$ und einem Wertebereich für f von 0 bis $7 \cdot 10^{11}$ mit 67 Datenpunkten drei Fehlerbalkendiagramme für einen $N(0,\sigma)$-normalverteilten Fehlervektor ε mit jeweils 66 Messwerten für die Fehlerbreiten $\sigma = 0{,}1\%$ von $y_{\max}$, $\sigma = 1\%$ von $y_{\max}$ und $\sigma = 10\%$ von $y_{\max}$ zum Maximalwert $y_{\max}$ der Strahldichte wie in Abbildung 4.5 ausgibt.

Hinweise: Verwenden Sie die physikalischen Konstanten aus dem Modul `scipy.constants` und die Matplotlib-Funktion `errorbar`.

[3] Unsöld und Baschek (1999):S. 110, 493.

Teil II

Datenanalyse

5 Regression

Kapitelübersicht

5.1 Residuen und Bewertung von Regressionsmodellen 60
5.2 Lineare Regression in einer Dimension 62
5.3 Mehrdimensionale lineare Regression 64
5.4 Nichtlineare Regression 68
5.4.1 Auflösung der Nichtlinearität mit Basisfunktionen 69
5.4.2 Ausgleichsrechnung 71
5.4.3 Anfangswerte des Schätzverfahrens 73
5.4.4 Fallbeispiel: Keplers Modell der Planetenbahnen 74
5.4.5 Konfidenzintervalle angepasster Modelle 77
5.5 Übungsaufgaben 78

In Definition 3.3 haben wir den Begriff eines statistischen Modells $f : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}$ eingeführt,

$$y = f(\boldsymbol{x}, \boldsymbol{\theta}) + \varepsilon.$$

Hier beschreibt die Funktion f eine Beziehung zwischen beobachteten Daten $\boldsymbol{x} \in \mathbb{R}^n$ und $y \in \mathbb{R}$, abhängig von gewissen Parametern $\boldsymbol{\theta} \in \mathbb{R}^k$. Der Fehlerterm $\varepsilon \in \mathbb{R}$ ist eine Zufallsvariable und anhand der Daten nicht beobachtbar. Die Beobachtungsdaten können durch Vektoren und Matrizen dargestellt werden:

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} \qquad \boldsymbol{X} = \begin{pmatrix} \boldsymbol{X}_1 \\ \vdots \\ \boldsymbol{X}_m \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}, \tag{5.1}$$

vgl. Bemerkung 3.4. Mit anderen Worten stellt die Funktion f das Modell dar und die Parameter $\boldsymbol{\theta}$ seine „Freiheitsgrade“. Das Problem des maschinellen Lernens und der statistischen Datenanalyse besteht dann darin, die Werte der Parameter $\boldsymbol{\theta}_*$ aus den beobachteten Daten $\boldsymbol{X}$ und $\boldsymbol{y}$ abzuleiten. Insbesondere beim maschinellen Lernen wird dieser Ableitungsprozess als „Modellanpassung“ oder „Modell-Fitting“ bezeichnet.

Sind nun alle Beobachtungsdaten, d.h. sowohl die unabhängigen Variablen $\boldsymbol{x}$ als auch die abhängige Variable y, metrische Größen, so nennt man das Modell ein *Regressionsmodell*, und den Prozess der Ableitung der Parameter aus den Beobachtungen *Regressionsanalyse*. Die häufigste Form der Regression ist die lineare Regression. Bevor wir uns

mit ihr befassen, betrachten wir zunächst die Frage, wie ein angepasstes Modell bewertet werden kann.

5.1 Residuen und Bewertung von Regressionsmodellen

Wenn ein Regressionsmodell $f(\boldsymbol{x}, \boldsymbol{\theta})$ gefittet worden ist, sind die Parameter $\boldsymbol{\theta}$ mit den Werten $\boldsymbol{\theta}_*$ so bestimmt, dass die *Residuen*[1]

$$e_i = y_i - f(\boldsymbol{X}_i, \boldsymbol{\theta}_*) \tag{5.2}$$

mit den m Datenpunkten $\boldsymbol{y}$ und $\boldsymbol{X}$ minimiert sind, wobei

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} \qquad \text{and} \qquad \boldsymbol{X} = \begin{pmatrix} \boldsymbol{X}_1 \\ \vdots \\ \boldsymbol{X}_m \end{pmatrix}, \tag{5.3}$$

wie in Bemerkung 3.4 gezeigt. Mit m Datenpunkten gibt es also auch m Residuen, wie in

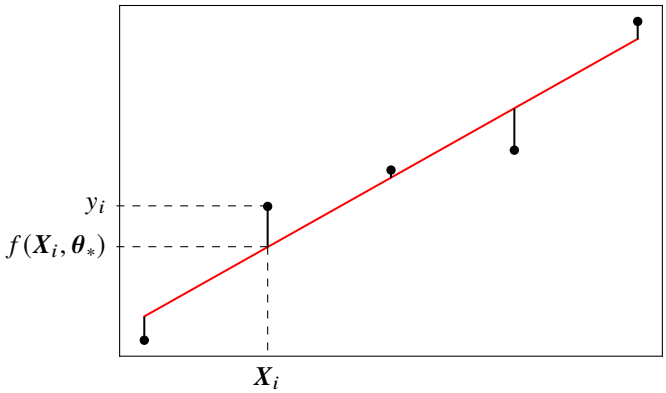


Abbildung 5.1. Die Residuen $e_i = y_i - f(\boldsymbol{X}_i, \boldsymbol{\theta}_*)$ eines gefitteten Regressionsmodells $f(\boldsymbol{x}, \boldsymbol{\theta}_*)$.

Abbildung 5.1 skizziert. Aber was heißt „die Residuen minimieren" überhaupt?

Die Berechnung der minimalen Residuen hängt von dem zugrundeliegenden Abstandsbegriff ab. In der Regel wird die Methode der kleinsten Quadrate verwendet, bei der die Summe der Quadrate der Residuen gebildet wird. Historisch gesehen wurde sie unabhängig und fast gleichzeitig von Gauß und Legendre eingeführt. Es sind jedoch auch andere Abstandsbegriffe möglich, die für die jeweilige Anwendung besser geeignet sein können. Einige davon werden weiter unten im Abschnitt 6.1.1 erwähnt.

Die Methode der kleinsten Quadrate stützt sich auf der *Summe der Residuenquadrate (residual sum of squares)* RSS. Dies ist die Abweichung des gemessenen Wertes y_i vom entsprechenden Modellwert $f(\boldsymbol{X}_i, \boldsymbol{\theta}_*)$:

$$\mathrm{RSS} = \sum_{i=1}^{m} \big(y_i - f(\boldsymbol{X}_i, \boldsymbol{\theta}_*)\big)^2. \tag{5.4}$$

Es ist die RSS, die minimiert wird, um die Koeffizienten $\boldsymbol{\theta}$ zu bestimmen, wenn die Methode der kleinsten Quadrate angewendet wird. Umgekehrt kann die Summe der quadrierten

[1] Einzahl: *Residuum*

Residuen daher für ein angepasstes Modell zur Bestimmung des *Bestimmtheitsmaßes* R^2 (sprich: „R Quadrat“) verwendet werden:

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}} \qquad \text{with} \qquad \text{TSS} = \sum_{i=1}^{m} (y_i - \bar{y})^2. \tag{5.5}$$

Hier steht TSS für die Gesamtsumme der Quadrate der beobachteten Zielwerte *(total sum of squares)*. Das Bestimmtheitsmaß gibt also den Anteil der Streuung der Daten an, den das angenommene Modell erklären kann. Je näher es bei 1 liegt, desto besser ist es. Es wird oft in Prozent ausgedrückt. Das Bestimmtheitsmaß ist somit ein Maß für die Anpassungsgüte des Modells und kann zum Vergleich mit anderen geschätzten Modellen herangezogen werden.

R^2 misst die Güte eines Regressionsmodells

Bemerkung 5.1. In Python ist eine bequeme Methode zur Berechnung der Anpassungsgüte eines Regressionsmodells die Verwendung von `r2_score` aus dem Paket `sklearn.metrics`. Als Eingabe benötigt es die wahren Zielwerte und die vom angepassten Modell vorhergesagten Werte:

```
from sklearn.metrics import r2_score
R2 = r2_score(y, y_pred)
```

□

Definition 5.2. Ein Regressionsmodell $f(\boldsymbol{x}, \boldsymbol{\theta})$, das linear in ihren Parametern $\boldsymbol{\theta}$ ist, d.h. das der Beziehung

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = g(\boldsymbol{x}) \cdot \boldsymbol{\theta} \tag{5.6}$$

für eine Funktion $g : \mathbb{R}^n \to \mathbb{R}^k$ genügt, heißt *parameter-lineares Modell*. □

Bemerkung 5.3 (Vorsicht bei der Verwendung von R^2 bei nichtlinearen Regressionsmodellen). Obwohl R^2 allgemein zur Messung der Anpassungsgüte eines Modells verwendet wird, ist es nicht generell das geeignete Maß für die Anpassungsgüte eines nichtlinearen Regressionsmodells. R^2 funktioniert perfekt für verallgemeinerte lineare Regressionsmodelle und ist ein gültiges Maß für die Anpassungsgüte, aber dies ist im Allgemeinen nicht der Fall für nichtlineare Modelle.[2] Der Grund dafür ist, dass die Gesamtsumme der Quadrate (TSS) nicht gleich der Summe der Regressionsquadrate (RegSS) plus der Summe der Residuenquadrate (RSS) ist, wie es bei der linearen Regression der Fall ist. Hier ist die Summe der Regressionsquadrate gegeben durch

$$\text{RegSS} = \sum_{i=1}^{m} \left(f(\boldsymbol{X}_i, \boldsymbol{\theta}_*) - \overline{f(\boldsymbol{X}_i, \boldsymbol{\theta}_*)} \right)^2, \tag{5.7}$$

und stellt die Summe der Quadrate der Differenzen der Modellvorhersagen ar, gegeben die beobachteten $\boldsymbol{x}$-Werte und deren Gesamtmittelwert.[3] Der Punkt ist nun, dass für ein verallgemeinertes Modell f, das (5.6) erfüllt, die Beziehung $\sum y_i = \sum_i f(\boldsymbol{X}_i, \boldsymbol{\theta}_*)$ gilt, d.h. der Mittelwert der beobachteten Zielwerte ist gleich dem Mittelwert der vorhergesagten Werte.[4] Für diese Modelle gilt also die Gleichung

$$\text{RegSS} = \sum_{i=1}^{m} \left(f(\boldsymbol{X}_i, \boldsymbol{\theta}_*) - \bar{y} \right)^2, \qquad \text{(für parameter-lineare Modelle)} \tag{5.8}$$

[2] Spiess und Neumeyer (2010).

[3] Spiess und Neumeyer (2010):Additional File 1.

[4] Zum Beweis siehe Sen und Srivastava (1990):Theorem 2.1, wobei X durch $g(\boldsymbol{x})$ ersetzt wird, und Korollar 2.2.

und damit

$$\text{TSS} = \text{RegSS} + \text{RSS} \qquad \text{(für parameter-lineare Modelle).} \tag{5.9}$$

Insbesondere folgt daraus, dass $R^2 \geqq 0$. Für nichtlineare Modelle ist dies jedoch möglicherweise nicht mehr gültig. Darüber hinaus liefert R^2 für den Vergleich nichtlinearer Modelle nicht immer ein klares Kriterium für die Auswahl des besten Modells, gegeben die beobachteten Daten. Eines der Probleme ist, dass R^2 sowie RSS von Modellen mit einer unterschiedlichen Anzahl von Parametern nicht miteinander verglichen werden können.[5] Für die Auswahl mehrerer Modelle wird häufig das BIC verwendet, das wir nun einführen. □

Definition 5.4. Gegeben sei ein statistisches Modell M mit k Parametern $\boldsymbol{\theta}$ und eine Datenstichprobe $(\boldsymbol{X}, \boldsymbol{y})$ der Größe n. Dann ist das *Bayes'sche Informationskriterium* BIC definiert durch[6]

$$\text{BIC} = k \ln n - 2 \ln L_* \tag{5.10}$$

wobei $L_* = P((\boldsymbol{X}, \boldsymbol{y}) \,|\, \boldsymbol{\theta}_*, M)$ die maximale Likelihood des Modells ist, vgl. (1.15) auf S. 12. D.h. $\boldsymbol{\theta}_*$ sind die Parameterwerte, die die Likelihood-Funktion maximieren. Das BIC kann von dem Wahrscheinlichkeitsverhältnis in Definition 2.10, S. 29, hergeleitet werden[7]. Wie ist das BIC zu interpretieren? Für eine gegeben Stichprobe gilt: je kleiner der Wert des BIC eines Modells ist, desto besser ist das Modell. □

Bemerkung 5.5. Speziell für Regressionmodelle, die auf der Methode der kleinsten Quadrate basiert und deren Fehlerterme identisch normalverteilt mit der Varianz σ_ε sind ("Gauß'sche Modelle") gilt $\sigma_\varepsilon^2 = \text{RSS}/m$, und daher $-2 \ln L_* = \ln \sigma_\varepsilon^2 = \ln(\text{RSS}/m)$. Bis auf eine nur von der Stichprobengröße m abhängige Konstante ergibt dies[8]

$$\text{BIC} = \ln(\text{RSS}/m) + k \ln m. \tag{5.11}$$

In einigen Internetquellen wird dies als Definition des BIC angegeben, ohne aber zu erwähnen, dass es nur für Gauß'sche Modelle gilt.[9] □

Bemerkung 5.6. In Python ist der einfachste Weg, das BIC eines Gauß'schen Modells zu berechnen, also eines Modells, in dem die Residuen normalverteilt um Null sind, eine Funktion `bic` zu definieren, die als Eingabe die Residuen e und die Anzahl k der Parameter des Modells erwartet und den Wert des BIC zurückgibt:

```
import numpy as np

def bic(e, k):
    return np.log(np.var(e)) + k*np.log(len(e))
```

cf. also `https://pypi.org/project/RegscorePy/`. □

5.2 Lineare Regression in einer Dimension

Beispiel 5.7. Betrachten wir als erstes Beispiel eine lineare Regression auf die Daten der Datei `kinder.csv`, deren Spaltentrenner ein Semikolon ist. Dazu importieren wir sie in

[5] cf. James et al. (2013):S. 210.

[6] cf. Durbin und Koopman (2012):S. 188; Hastie et al. (2009):S. 233; James et al. (2013):S. 212; Pourret et al. (2008):S. 62; Shumway und Stoffer (2017):S. 50.

[7] Hastie et al. (2009):S. 234.

[8] Shumway und Stoffer (2017):S. 50.

[9] e.g., `https://pypi.org/project/RegscorePy/`, `https://stackoverflow.com/questions/60823638`

Zeile 6 zunächst als Pandas Dataframe `df` und filtern anschließend in den Zeilen 8 bis 11 die Spalten `"Größe"` und `"Alter"` als Spalten-Arrays, mit den Anweisungen aus Abschnitt 4.3.1:

```
1  %matplotlib inline
2  import matplotlib.pyplot as plt
3  import numpy as np
4  import pandas as pd
5
6  df = pd.read_csv("./datasets/kinder.csv", sep=';') # Lädt Datei in Dataframe
7
8  merkmale = ["Größe"]                # Liste der unabhängigen Variable(n)
9  ziel = "Alter"                      # abhängige Variable
10 X = np.c_[df[merkmale]]             # extrahiert die Merkmalswerte als Matrix
11 y = np.c_[df[ziel]]                 # extrahiert die Zielwerte
12
13 from sklearn.linear_model import LinearRegression
14 model = LinearRegression()          # erzeugt die lineare Regression als Modell
15 model.fit(X, y)                     # berechnet die Modellparameter
16 print("Parameter: θ_0 =", model.intercept_, ", θ_1 =", model.coef_)
17
18 X_test = [[X.min()], [X.max()]]       # <- Anfangs- und Endpunkt als Liste
19 y_pred = model.predict(X_test)        # Prognose der "Testwerte"
20
21 plt.scatter(X, y, color="black")      # Streudiagramm der Trainingsdaten
22 plt.plot(X_test, y_pred, color="red") # Regressionsgerade der Prognose
23 plt.show()                           # Zeigt Plot am Bildschirm
```

In Zeile 13 importieren wir das Modell der Linearen Regression aus Scikit-Learn. (Das hätten wir natürlich genauso gut auch im Programm ganz oben tun können, aber an dieser Stelle springt es sofort ins Auge.) In Zeile 14 erstellen wir das Modell, bevor in Zeile 15 die Berechnung der relevanten Modellparameter mit der Anweisung `model.fit(X,y)` durchgeführt wird. Die beiden Parameter der linearen Regression sind der Aufpunkt (engl.

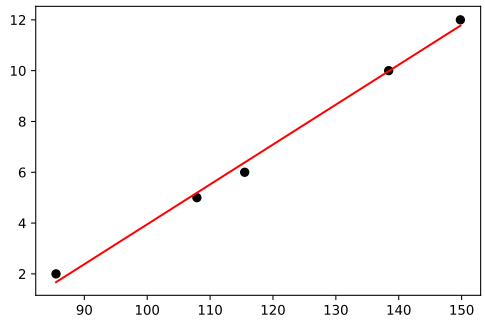


Abbildung 5.2. Ausgabe des Programms zur linearen Regression

intercept) als erster Parameter θ_0 und die Steigung θ_1 als zweiten, vgl Beispiel 3.5. Um nun die Regressionsgerade zu berechnen, gibt es verschiedene Möglichkeiten, wir verwenden hier in Zeile 19 die Modellvorhersage mit der „Testmatrix“ `X_test`, die den kleinsten und den größten Wert als Testdaten erhält. Das Ergebnis speichern wir als „Vorhersage“ in `y_pred`. Die Ausgabe des Programms ist die Grafik in Abbildung 5.2 und geschieht ab Zeile 20.

Wie gut ist das Modell? Um die Anpassungsgüte zu bestimmen, wenden wir das Bestimmtheitsmaß R^2 und das BIC an:

```
# Model evaluation:
```

```
from sklearn.metrics import r2_score
R2 = r2_score(y, y_pred)
print("R^2 =", f"{R2:.3%}")

def bic(e, k):
    return np.log(np.var(e)) + k*np.log(len(e))
BIC = bic(y - y_pred, 2)
print("BIC =", f"{BIC:.3f}")
```

Für unsere kleine Stichprobe erhalten wir also

$$R^2 = 99.463\%, \qquad \text{BIC} = 0.542.$$

Damit ist R^2 nahe bei 1 und sieht ganz gut aus, wie wir uns intuitiv mit dem Plot überzeugen können. Das BIC ist ein Maß für den Vergleich von Modellen bei gegebenen Beobachtungsdaten, d. h. der Wert allein hat keine direkte Bedeutung. □

5.3 Mehrdimensionale lineare Regression

Bisher haben wir Regressionen mit einer einzigen unabhängigen Variablen x betrachtet, d.h. Regressionsmodelle $y = f(x, \theta)$ mit einer Modellfunktion $f : \mathbb{R} \times \mathbb{R}^k \to \mathbb{R}$. Regressionen mit mehreren Merkmalen $\boldsymbol{x} = (x_1, \ldots, x_n)$ für $n \geqq 2$, werden als *mehrdimensional* oder *multivariat* bezeichnet. Im Prinzip können mehrdimensionale Regressionsmodelle linear oder nichtlinear sein. Beachten Sie jedoch die Probleme nichtlinearer Modelle, die wir zu Beginn des Abschnitts 5.4 erwähnt haben und die in mehreren Dimensionen sicher nicht einfacher werden. Eine Möglichkeit, mehrdimensionale lineare Regressionen in Python durchzuführen, ist die Verwendung von `PolynomialFeautures` aus `scikit.learn`, wie in Abschnitt 5.4.1 beschrieben.

Für mehrdimensionale Regressionen gilt weiterhin der in Definition 3.3 und Bemerkung 3.4 beschriebene Rahmen. Darüber hinaus bleiben alle Berechnungen zur Anpassung des Modells formal gleich, mit dem einzigen Unterschied, dass die algebraischen Operanden der Addition und Multiplikation nicht mehr reine Zahlen x und y sind, sondern Vektoren und Matrizen. Folglich ist die Python-Bibliothek `scikit-learn` auch für den mehrdimensionalen Fall anwendbar. (Allerdings funktioniert die Kurvenanpassung mit `scipy.optimize` nicht für den mehrdimensionalen Fall.) Betrachten wir dies anhand des folgenden zweidimensionalen Beispiels etwas genauer.

Beispiel 5.8. [10] Der SAT *(Scholastic Assessment Test)* ist ein Studierfähigkeitstest, der hauptsächlich von Studienplatzbewerbern an US-amerikanischen Universitäten gefordert wird. Er besteht aus zwei Teilen, nämlich dem *Evidence-Based Reading and Writing* (EBRW, der „Englisch-Teil“) und dem Mathematik-Teil. Die folgende Tabelle zeigt die erlangten Punkte der SAT's von neun Studierenden und ihren erreichten Notendurchschnitt (GPA, Maximum ist 4):

SAT-Math	79	71	75	74	70	67	73	79	76
SAT-English	74	76	66	76	76	66	71	71	57
GPA	3.95	3.84	3.68	3.59	3.57	3.49	3.47	3.40	3.08

[10] Sen und Srivastava (1990):S. 29, 34.

Wie beeinflussen die SAT-Ergebnisse den GPA? Diese Frage lässt sich mit einer zweidimensionalen linearen Regression beantworten, d.h. mit dem Modell

$$y = f(\boldsymbol{x}, \boldsymbol{\theta}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \tag{5.12}$$

und den Daten

$$\boldsymbol{y} = \begin{pmatrix} 3.95 \\ 3.84 \\ 3.68 \\ 3.59 \\ 3.57 \\ 3.49 \\ 3.47 \\ 3.40 \\ 3.08 \end{pmatrix}, \qquad \boldsymbol{X} = \begin{pmatrix} 79 & 74 \\ 71 & 76 \\ 75 & 66 \\ 74 & 76 \\ 70 & 76 \\ 67 & 66 \\ 73 & 71 \\ 79 & 71 \\ 76 & 57 \end{pmatrix} \tag{5.13}$$

($m = 9$) Die lineare Regression löst nun die Gleichung $\boldsymbol{y} = f(\boldsymbol{X}, \boldsymbol{\theta}) + \boldsymbol{\varepsilon}$ bezüglich der drei Parameter $\boldsymbol{\theta} = (\theta_0, \theta_1, \theta_2)^T$ unter Minimierung der Residuenquadrate. Diese Gleichung ist explizit durch das System der $m = 9$ Gleichungen

$$\begin{aligned} 3.95 &= \theta_0 + 79 \cdot \theta_1 + 74 \cdot \theta_2 + \varepsilon_1 \\ 3.84 &= \theta_0 + 71 \cdot \theta_1 + 76 \cdot \theta_2 + \varepsilon_2 \\ &\dots \\ 3.08 &= \theta_0 + 76 \cdot \theta_1 + 57 \cdot \theta_2 + \varepsilon_9 \end{aligned}$$

gegeben. Wir können die Regression mit dem folgenden Python-Programm implementieren:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

def bic(e, k):
    return np.log(np.var(e)) + k*np.log(len(e))

df = pd.read_csv("./datasets/GPA-vs-SAT-scores.csv", sep="\t") # loads file

features = ["SAT-Math","SAT-English"] # list of independent variables
target   = "GPA"                      # dependent variable
X = np.c_[df[features]]               # extract feature values as a matrix
y = np.c_[df[target]]                 # extraxt target values

model = LinearRegression()            # generate linear regression as model
model.fit(X, y)                       # adjust the model parameters
print("Parameter: θ₀ =",model.intercept_,", θ₁ =",model.coef_[0,0],", θ₂ =",model.coef_[0,1])

y_pred = model.predict(X)             # model predictions

# Model evaluation:
R2 = r2_score(y, y_pred)
print("R² =", f"{R2:.3f}")
print("BIC =", f"{bic(y - y_pred, len(features)+1):.3f}")
```

Das ergibt als Lösung $\theta_0 = 1.22$, $\theta_1 = 0.0044$, $\theta_2 = 0.029$, d.h.,

$$\text{GPA} = 1.22 + 0.0044 \cdot \text{SAT-Math} + 0.029 \cdot \text{SAT-English}. \tag{5.14}$$

Die Modellbewertungen ergeben

$$R^2 = 0.516, \qquad \text{BIC} = 3.005. \tag{5.15}$$

Gleichung (5.14) scheint darauf hinzudeuten, dass die Ergebnisse von SAT-Math nur einen sehr geringen Einfluss auf den GPA haben. Zu beachten ist jedoch, dass die Wirkung einer Variablen von den anderen Variablen im Modell abhängt. Mit x_1 = SAT-Math allein erhalten wir

$$\text{GPA} = 3.5538 + 0.0001 \cdot \text{SAT-Math}$$

mit den Modellbewertungen

$$R^2_{\text{Math}} = 4 \cdot 10^{-6}, \qquad \text{BIC} = 1.533. \tag{5.16}$$

Somit ist SAT-Math allein ein noch schlechterer Prädiktor für den GPA, zumindest für die neun Studierenden dieser Stichprobe. Für x_2 = SAT-English als einzige unabhängige Variable erhalten wir

$$\text{GPA} = 1.566 + 0.02840 \cdot \text{SAT-English}$$

mit den Modellbewertungen

$$R^2_{\text{Engl}} = 0.511, \qquad \text{BIC} = 0.818. \tag{5.17}$$

Vergleichen wir die drei Modelle anhand ihrer Bewertungen in den Gleichungen (5.15), (5.16) und (5.17), ergibt sich ein widersprüchliches Bild. Gemäß der Bestimmtheitsmaße R^2 ist das zweidimensionale Modell besser als die beiden eindimensionalen Modelle. Allerdings liefert das BIC für die eindimensionalen Modelle ein besseres Ranking. □

Beispiel 5.9. Um die Regression zu visualisieren, müssen wir zunächst verstehen, dass wir ein dreidimensionales Streudiagramm erstellen müssen, da wir zusammen mit den Zielwerten drei Variablen x_1, x_2, y haben. Ein 3D-Streudiagramm kann in Python mit dem Modul `mpl_toolkits.mplot3d` implementiert werden. Ausgehend von den Beobachtungsdaten `X` und `y` aus dem Quelltext in Beispiel 5.8 oben erhalten wir das 3D-Streudiagramm wie folgt:

```
%matplotlib notebook
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d

ax = plt.axes(projection="3d")

ax.scatter(X[:,0], X[:,1], y.ravel(), color="black") # scatter plot of training data
ax.set_yticks(range(55,76,5))                        # sets y-ticks
ax.set_xlabel("SAT-Math"); ax.set_ylabel("SAT-English"); ax.set_zlabel(target)
ax.view_init(azim=-130, elev=24)                     # camera position
plt.show()
```

Es ist in Abbildung 5.3 a) dargestellt. Für den zweidimensionalen Fall haben wir offensichtlich keine Regressions*gerade*, sondern eine Regressions*ebene*. Ausgehend von den Beobachtungsdaten `X` und `y` kann diese Ebene durch ein Numpy Meshgrid dargestellt werden, das von den Minimal- und Maximalwerten der beiden Merkmale x_1 = `X[:,0]` und x_2 = `X[:,1]` aufgespannt wird und als Testdaten zur Vorhersage der Zielwerte durch das Modell dient. Die Ebene wird dann mit der Funktion `plot_trisurf` gezeichnet:

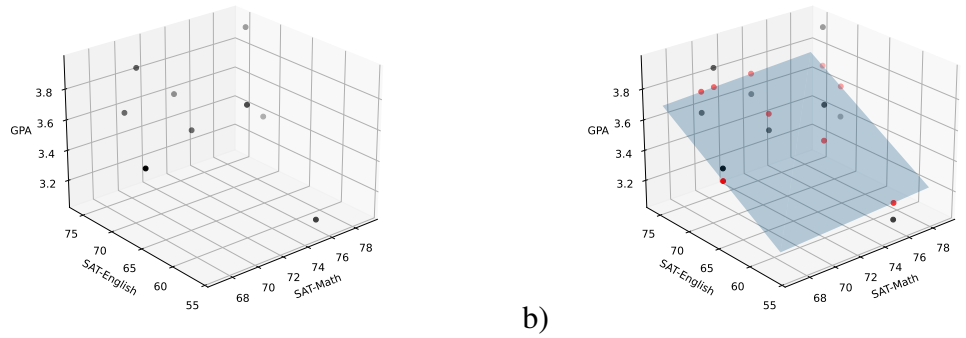


Abbildung 5.3. a) Streudiagramm der SAT-Punkte in Beispiel 5.8. b) Streudiagramm mit der Regressionsebene. Die roten Punkte sind die in der Ebene liegenden Modellwerte, die Residuen ergeben sich aus den Abständen zu den schwarz gepunkteten Datenwerten derselben SAT-Werte.

```
%matplotlib notebook
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d

# Regression plane with min/max of training data:
mg = np.meshgrid(np.r_[X[:,0].min(), X[:,0].max()], np.r_[X[:,1].min(), X[:,1].max()])
zz = model.predict(np.c_[mg[0].ravel(),mg[1].ravel()]).ravel() # predict the training data

# -- 3D scatter plot:
ax = plt.axes(projection="3d")

ax.scatter(X[:,0], X[:,1], y.ravel(), color="black") # scatter plot of training data
ax.scatter(X[:,0], X[:,1], y_pred, color="red")      # scatter plot of predictions
ax.plot_trisurf(mg[0].ravel(), mg[1].ravel(), zz, linewidth=0, alpha=0.3) # regression plane
ax.set_yticks(range(55,76,5))                        # set y-ticks
ax.set_xlabel("SAT-Math"); ax.set_ylabel("SAT-English"); ax.set_zlabel(target)
ax.view_init(azim=-130, elev=24)                     # camera position
plt.show()
```

Das Ergebnis dieses Programms ist in Abbildung 5.3 b) dargestellt. Hier sind die roten Punkte die in der Ebene liegenden Modellwerte, die Residuen ergeben sich aus den Abständen zu den schwarz gepunkteten Datenwerten derselben SAT-Werte. Vergleichen wir

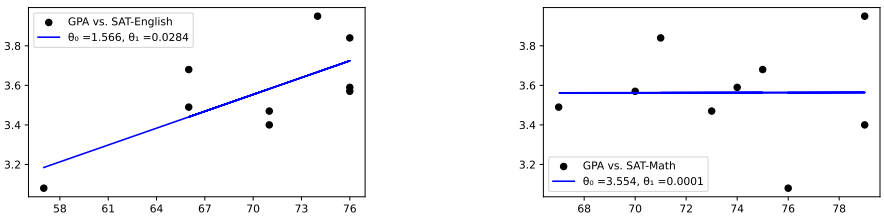


Abbildung 5.4. Streudiagramme GPA versus die individuellen SAT-Punkte und die jeweiligen Regressionsgeraden.

diese Darstellung mit den Darstellungen der Regressionsgeraden der beiden eindimensionalen Modelle in Abbildung5.4. Die SAT-Math-Regressionslinie hat eine sehr kleine Steigung $\theta_1 = 10^{-4}$, so dass der Achsenabschnitt $\theta_0 = 3.544$ sehr nahe bei dem Mittelwert des GPA liegt, $\bar{y} = 3.56\bar{3}$. Wie wir in Beispiel 5.8 oben gesehen haben, geben die Bestimmtheitsmaße der Modelle dem zweidimensionalen Modell den Vorrang. Im linken

Diagramm in Abbildung 5.4 sehen wir ein weiteres Argument für das zweidimensionale Modell: Mehrere SAT-English-Bewertungen kommen mehrfach vor, so $x_2 = 76$ dreimal und $x_2 = 66$ sowie $x_2 = 71$ jeweils zweimal. Sie haben jedoch alle unterschiedliche GPA-Werte. Aus Sicht des SAT-English-Modells bleibt dieses Phänomen völlig unklar. Erst im zweidimensionalen Modell wird der Einfluss der SAT-Math-Punkte offenbar. □

5.4 Nichtlineare Regression

Oft gibt es Problemstellungen, für die komplexere Modelle angewendet werden müssen als diejenigen, die wir bisher kennengelernt haben. Das ist beispielsweise der Fall, wenn beobachtete Datenwerte zwar einen kausalen Zusammenhang erkennen lassen, dieser aber eben nicht linear ist. Eigentlich ist dies sogar die Regel, da die Realität meist nichtlinear ist und sich bestenfalls durch ein lineares Modell approximieren lässt. Zwar ist in der Wissenschaft stets die Genauigkeit eines Modells gegen seine Fähigkeit abzuwägen, die wesentlichen Einflüsse der betrachteten Phänomene zu erklären. Das Ockham'sche Rasiermesser, nach dem ein Modell so einfach wie möglich und so komplex wie nötig sein sollte, drückt diese Abwägung wissenschaftstheoretisch aus.

Ockhams Rasiermesser

Ist nun jedoch ein nichtlinearer funktionaler Zusammenhang zwischen beobachteten Daten gesucht, so bietet sich die nichtlineare Regression an.[11] Hier ist die angesetzte Modellfunktion $f : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}$,

$$y = f(\boldsymbol{x}, \boldsymbol{\theta}) \tag{5.18}$$

nichtlinear in der unabhängigen Variablen $\boldsymbol{x}$. Leider sind wir bei einer nichtlinearen Regressionsanalyse mit einer ganzen Reihe von Problemen konfrontiert. Es ist oft allein schon schwierig, den geeigneten Typ der Funktion zu erkennen. Hier sollten Erkenntnisse aus dem sachlogischen Zusammenhang der Daten einfließen, ebenso ist ein gewisses Maß an Erfahrung und Intuition hilfreich. So liegen nichtlinearen Regressionsmodellen oft ganze wissenschaftliche Theorien zugrunde, manchmal jedoch reicht schon ein kurzer Blick auf den Plot der Daten. Zum zweiten sind nichtlineare Regressionen oft sehr rechenintensiv. In ungünstigen Fällen führen die Berechnungen dann nicht einmal zur besten Lösung, oder vielleicht sogar zu überhaupt keiner Lösung – sei es, da die Recheniterationen nicht konvergieren, sei es, da eine Lösung gar nicht existiert! Abstrakt betrachtet bergen nichtlineare Ansätze im Gegensatz zu linearen die Gefahr, dass sie mehrere lokale Optima haben, dass das globale Optimum nicht gefunden wird, oder dass ein Optimum gar nicht erst existiert.

Nichtlineare Probleme haben nicht immer eine eindeutige Lösung, sondern manchmal gar keine, manchmal gleich mehrere

Sehr wichtig beim Fitten nichtlinearer Regressionsmodelle ist oft die Vorgabe geeigneter Anfangswerte für die Approximation der Parameter, so dass die Rechenverfahren eine Lösung liefern. Dieses Problem hängt jedoch individuell von der Modellfunktion f ab, d.h. es gibt kein allgemein funktionierendes Vorgehen. Da hilft oft nur „Fummeln", also *trial-and-error*, in der Regel unterstützt durch Funktionsplots.

Anfangswerte kritisch!

Kurzum, das Ganze ist so kompliziert, dass eine nichtlineare Regression nur für eingeschränkte Fälle möglich ist. Im Wesentlichen gibt es zwei praktikable Ansätze: (1) Auflösung der Nichtlinearität durch Projektion in höhere Dimensionen oder (2) Ausgleichsrechnung. Mit Scikit-Learn kann der erste Ansatz implementiert werden. Für den zweiten Ansatz wird zwar in Scikit-Learn keine Möglichkeit angeboten, dafür jedoch in der Python-Bibliothek `scipy`. Beide Zugänge werden wir in den folgenden Abschnitten betrachten.

2 Ansätze für nichtlineare Regression in Python

[11] Backhaus et al. (2015):§1.

5.4.1 Auflösung der Nichtlinearität mit Basisfunktionen

In Scikit-Learn können wir eine nichtlineare Regression implementieren, wenn die Funktion f in (5.18) als Linearkombination endlich vieler Basisfunktionen $g_1, \ldots, g_k : \mathbb{R}^n \to \mathbb{R}$ dargestellt werden kann:

$$f(\boldsymbol{x}, \boldsymbol{\theta}) \;=\; \theta_0 + \theta_1 g_1(\boldsymbol{x}) + \ldots + \theta_k g_k(\boldsymbol{x}) \tag{5.19}$$

Wir können dann die Nichtlinearität von f in ein höherdimensionales, aber zumindest bezüglich der Parameter $\boldsymbol{\theta}$ *lineares* Modell projizieren[12]:

Nichtlinear in n Dimensionen		Linear in $n \times k$ Dimensionen
$\downarrow$		$\uparrow$
$f(\boldsymbol{x}, \boldsymbol{\theta})$	$\longrightarrow$	$\theta_1 g_1(\boldsymbol{x}) + \ldots + \theta_k g_k(\boldsymbol{x})$

Vergleiche dazu auch Gleichung (5.6) Wie kann so eine solche Linearisierung aussehen? Ein Beispiel dafür ist die polynomielle Regression, also eine nichtlineare Regression mit der Modellfunktion

$$f(x, \boldsymbol{\theta}) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_k x^k, \tag{5.20}$$

einem Polynom k-ten Grades. Da dieses Modell bereits linear bezüglich der $(k+1)$ Modellparameter θ_i ist (sie also niemals miteinander multipliziert oder durch einander geteilt werden), können wir mit den Basisfunktionen

$$g_0(x) = 1, \quad g_1(x) = x, \quad g_2(x) = x^2, \quad \ldots, \quad g_k(x) = x^k \tag{5.21}$$

die Funktion f wie in (5.19) darstellen – allerdings ergänzt um den konstanten Term $\theta_0 g_0(x)$. In Scikit-Learn ist diese Projektion mit der Transformation `PolynomialFeatures` aus dem Paket `sklearn.preprocessing` implementierbar. Sie bildet jeden Wert x eines Spaltenvektors in eine Zeile mit seinen einzelnen Potenzen $x^0, x^1, ..., x^k$ ab, also

Projektion mit `PolynomialFeatures`

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad \longrightarrow \quad \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ \vdots & & & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^k \end{pmatrix} \tag{5.22}$$

Betrachten wir dazu die Anweisungen:

```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
X = np.c_[[2,3]]
poly = PolynomialFeatures(3)
X_trans = poly.fit_transform(X)
print(X_trans)
```

Sie ergeben die Ausgabe:

```
[[ 1.  2.  4.  8.]
 [ 1.  3.  9. 27.]]
```

Mit der Option `include_bias=False` kann man die konstanten Terme $x^0 = 1$ ausschließen:

[12] VanderPlas (2018):§5.6.2.

```
X = np.c_[[2,3,4]]
poly = PolynomialFeatures(3, include_bias=False)
X_trans = poly.fit_transform(X)
print(X_trans)
```

Damit erhalten wir:

```
[[ 2.   4.   8.]
 [ 3.   9.  27.]
 [ 4.  16.  64.]]
```

Für den Erfolg eines solchen Ansatzes entscheidend ist die geeignete Wahl der Basisfunktionen. Betrachten wir für Möglichkeiten und Grenzen dazu das folgende Beispiel eines nichtlinearen Polynoms vom Grad 7 als Modellfunktion, das mit Hilfe einer Pipeline in Scikit-Learn als lineares 7-dimensionales Regressionsmodell analysiert wird.

```
%matplotlib inline
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model  import LinearRegression
import matplotlib.pyplot as plt

# -- 0. Simulate observational data: --
num = 50  # number of observations
X = np.c_[np.linspace(0, 10, num)]
y = np.sin(X.ravel()) + 0.1*np.random.RandomState(1).randn(num)

#plt.scatter(X,y)

# -- 1. Nonlinear model with base polynomial of degree 7: --
poly = PolynomialFeatures(7)
X_trans = poly.fit_transform(X)
model = LinearRegression()
model = model.fit(X_trans, y)

y_pred = model.predict(X_trans)
print("score=",model.score(X_trans, y))

#-- 2. Plot data points and regression curve: --
plt.scatter(X, y)
plt.plot(X, y_pred)  # predictions in traing data
plt.show()
```

Der Vergleich der in einer Punktewolke dargestellten simulierten Trainingsdaten mit der modellierten Regressionskurve in Abbildung 5.5 zeigt, dass ein Polynom 7. Grades die Daten gut modelliert.

Problemfall ungeeignete Basisfunktionen

Oder stimmt hier irgendetwas vielleicht doch nicht? Die Daten für y sind als Sinuskurve mit zufälligen Störwerten simuliert. Auch an der Grafik erwarten wir intuitiv, dass die Daten periodisch fortfahren. Ein Polynom aber kann niemals periodisch sein! Würden wir mit diesem Modell beispielsweise Daten vorhersagen wollen, deren x-Werte nicht im Trainingsintervall $[0, 10]$ liegen, erleben wir eine Enttäuschung:

```
X_curve = np.c_[np.linspace(0, 11.5, 1000)]
```

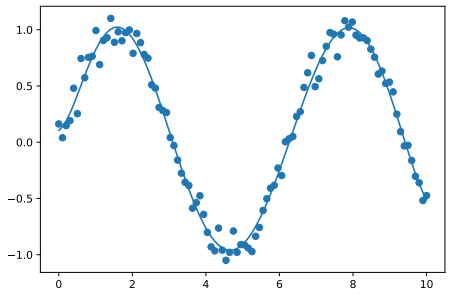


Abbildung 5.5. Linearer Fit für nichtlineare Trainingsdaten

```
X_trans = poly.fit_transform(X_curve)
y_curve = model.predict(X_trans)
plt.scatter(X, y)
plt.plot(X_curve, y_curve)
```

liefert den Plot in Abbildung 5.6. So gut das Regressionsmodell also auf dem Trainings-

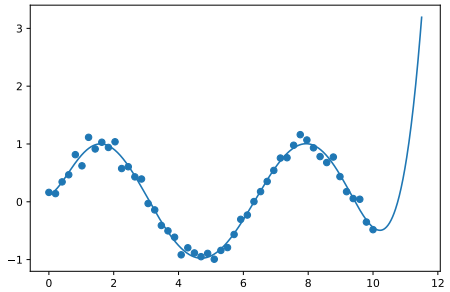


Abbildung 5.6. Linearer Fit für nichtlineare Trainingsdaten

intervall ist, so unbrauchbar ist es außerhalb davon. Wir haben also nicht die „wahre“ Gesetzmäßigkeit des funktionalen Zusammenhangs $y = f(x)$ herausgefunden, sondern einen Overfit trainiert!

Die Wahl von geeigneten Basisfunktionen erfordert also, dass wir die Daten „verstehen“. Wollen wir beispielsweise periodische Vorgänge modellieren, brauchen wir auch periodische Basisfunktionen. Scikit-Learn sieht solche Funktionen nicht vor. VanderPlas[13] gibt jedoch in Kapitel 5.6.2 ein schönes Beispiel an, das als Vorbild zur Erstellung eigener Klassen dienen kann, mit denen man beliebige Basisfunktionen in ein Scikit-Learn Modell integrieren kann.

In Scikit-Learn können eigene Basisfunktionen implementiert werden

Übrigens liefert für die polynomielle Regression auch die Klasse `SVR` des Moduls sklearn.svm Lösungsmöglichkeiten.

5.4.2 Ausgleichsrechnung

Ein für den eindimensionalen Fall oft einfacheres Vorgehen ist die *Ausgleichungsrechnung* (engl. *curve fitting*) für eine beliebige Funktion

$$f(x, \theta_1, \theta_2, \ldots) \tag{5.23}$$

anhand beobachteter Daten x und y. Sie wird in Python mit der Methode `curve_fit(f,x,y)` des Moduls `scipy.optimize` zur Verfügung gestellt. Dabei muss die Funktion f wie in

`curve_fit` erwartet eine bestimmte Struktur der Funktion f und der Daten x, y

[13] VanderPlas (2018).

(5.23) als ersten Parameter stets die unabhängige Variable x enthalten und kann um die anzupassenden Modellparameter $\theta_1, \theta_2, \ldots$ als weitere Parameter ergänzt werden. Daneben erwartet die Methode `curve_fit` die beobachteten Daten der unabhängigen Variablen x und der abhängigen Variablen y als Numpy-Arrays (hier ist scipy leider nicht so flexibel wie andere Module wie Scikit-Learn oder Statsmodels). Für eine klassische lineare Regression z.B. können wir programmieren:

```
import numpy as np
from scipy.optimize import curve_fit
def f(x,a,b): return a + b*x  % das Regressionsmodell
x = np.array([x_1, ... x_m])
y = np.array([y_1, ... y_m])
coefs, res = curve_fit(f, x, y)
```

Hier werden die Modellparameter mit a und b bezeichnet und stellen den Aufpunkt und die Steigung der Regressionsgeraden dar. Die Methode `curve_fit` wird nun mit der Funktion als ersten Parameter und den Beobachtungsdaten x und y aufgerufen. Ihr Ergebnis ist ein Paar von Arrays, wobei das erste die Werte der Modellparameter enthält und das zweite deren Kovarianzmatrix. Die Berechnung dieser Werte geschieht mit Hilfe der Methode der kleinsten Quadrate. Sie minimiert die Quadratsumme der Residuen SSR (*sum of squared residuals*), wie in Gleichung (5.4) oben definiert wurde, und wird zur Berechnung von R^2 in Gleichung (5.5) verwendet.

Beispiel 5.10. *(Polynom 3. Grades)* Betrachten wir als ersten Testfall die durch die folgenden Anweisungen simulierten Daten für x und y:

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

x = np.array([-2,-1,0,1,2,3,4,5,6])
y = x**3 - 6 * x**2 + 3*x + 20
```

Die Werte von y sind hier also per Konstruktion in einem funktionalen Zusammenhang mit den Werten von x, nämlich

$$y = x^3 - 6x^2 + 3x + 20, \tag{5.24}$$

also ein Polynom dritten Grades. Tun wir nun aber so, also wüssten wir das nicht und sähen lediglich die Datenpunkte als Streudiagramm wie in Abbildung 5.7. Da ist die Funktion

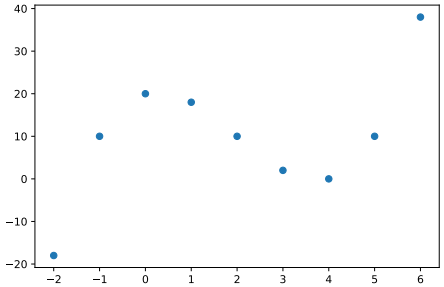


Abbildung 5.7. Daten eines kubischen Polynoms

nicht direkt zu erkennen. Wenden wir auf diese Daten nun das Modell

$$f(x) = a + bx + cx^2 + dx^3 \tag{5.25}$$

eines Polynoms dritten Grades mit den vier Modellparametern a, b, c und d an, so können wir das in Python mit den folgenden Anweisungen erreichen:

```
from scipy.optimize import curve_fit
def f(x,a,b,c,d): return a + b*x + c * x**2 + d * x**3
coefs, cov = curve_fit(f, x, y)
```

Geben wir die Werte der Modellparameter, also die Werte $y_{\text{pred}} = f(X, \boldsymbol{\theta}_*)$ des angepassten Modells, mit Hilfe der Anweisung

```
print("Koeffizienten:", coefs)
```

aus, so ergibt sich

```
Koeffizienten: [20.  3. -6.  1.]
```

d.h. $a = 20$, $b = 3$, $c = -6$ und $d = 1$. Tor! Dass dieses geschätzte Modell perfekt die Daten repräsentiert, wird einerseits visuell dadurch plausibilisiert, dass wir die Regressionskurve

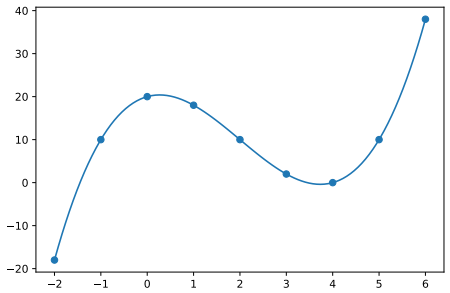


Abbildung 5.8. Die Daten und die geschätzte Regressionskurve

plotten (Abb. 5.8). Andererseits wird es mit Hilfe der Summe der Residuenquadrate anhand der Anweisungen

```
y_pred = f(x, *coefs)
from sklearn.metrics import r2_score
R2 = r2_score(y, y_pred)

def bic(e, k):
    return np.log(np.var(e)) + k*np.log(len(e))

print("R² =", f"{R2:.3%},", "BIC =", f"{bic(y - y_pred, len(coefs)):.0f}")
```

mit dem Bestimmtheitsmaß von $R^2 = 100\%$ bestätigt. Der BIC beträgt -56. □

5.4.3 Anfangswerte des Schätzverfahrens

Bei der nichtlinearen Regression kann es bei der Berechnung der Modellparameter zu dem ernsten Problem kommen, das man von der linearen Regression nicht kennt, dass die Schätzwerte ein unsinniges Ergebnis produzieren oder manchmal gar keine Lösung gefunden und die Berechnung abgebrochen wird. (`curve_fit` stoppt seine Berechnung nach 800 Iterationen.) Das Auffinden einer unsinnigen Lösung ist dabei vielleicht noch schlimmer als wenn das Verfahren abbricht, da oft keine Warnmeldung erfolgt und man ohne Weiteres den Fehler nicht bemerkt. Eine unsinnige Lösung erkennt man oft visuell, wenn man die beobachteten Werte und das geschätzte Modell in einem Diagramm plottet, oder daran, dass das Bestimmtheitsmaß größer 1 oder kleiner 0 ist.

Ursache dafür ist, dass das iterative Schätzverfahren sehr empfindlich von den Anfangswerten der Modellparameter abhängt. Die Methode `curve_fit` startet standardmäßig mit dem Wert 1 für alle Modellparameter. In manchen Fällen führt dies dazu, dass das Schätzverfahren nicht konvergiert. Mit dem optionalen Parameter `p0` kann man `curve_fit` beim Aufruf eine Liste von Anfangswerten übergeben, wobei sie in derselben Reihenfolge und Anzahl auftauchen müssen wie in der Regressionsfunktion, also z.B.:

```
def f(x,a,b,c): return a*x + b/x + c
curve_fit(f, x, y, p0=(0.5,2,100))
```

für die Anfangswerte $a = 0.5$, $b = 2$ und $c = 100$.

Das Finden geeigneter Anfangswerte kann ein schwieriges Unterfangen werden und lässt sich nicht automatisieren. *„Unfortunately, there are no good rules for starting values.“*[14] Oft muss man sich mit Versuch und Irrtum herantasten. Und manchmal hilft auch das und viel Geduld nicht: Vielleicht ist das Regressionsmodell ungeeignet oder sogar falsch!?

5.4.4 Fallbeispiel: Keplers Modell der Planetenbahnen

Betrachten wir als Beispiel ein Problem aus der Astronomie, das Kepler am 15. Mai 1608 löste und ein Jahr später in seinem Werk *Harmonice Mundi* veröffentlichte. Wie wir es

Planet	Merkur	Venus	Erde	Mars	Jupiter	Saturn	Uranus	Neptun
Umlaufzeit [a]	0,2408	0,6152	1	1,8810	11,8626	29,4475	84,0168	164,7913
Halbachse [AE]	0,3871	0,7233	1	1,5237	5,2033	9,5371	19,1912	30,0690

Tabelle 5.1. Bahndaten der Planeten des Sonnensystems: Die große Halbachse in Astronomischen Einheiten (AE) und die Umlaufzeiten in Jahren [a]. Quelle: `https://de.wikipedia.org/wiki/Liste_der_Planeten_des_Sonnensystems`

heute ausdrücken würden, gelang es ihm, einen funktionalen Zusammenhang zwischen der Umlaufzeit T und der großen Halbachse r der Planeten des Sonnensytems zu ermitteln. Aus Sicht der Datenanalyse kann man sein Problem auffassen als die Entwicklung eines nichtlinearen Regressionsmodells aus gegebenen Beobachtungsdaten. Die Daten sind in Tabelle 5.1 aufgelistet und mit den folgenden Anweisungen in Abbildung 5.9 in einem

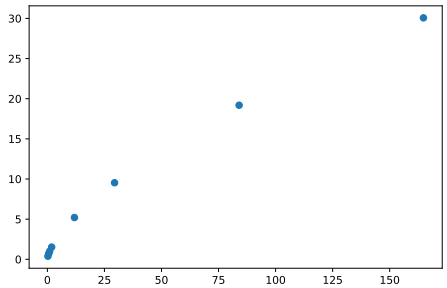


Abbildung 5.9. Die Halbachsen der Planeten aufgetragen gegen ihre Umlaufzeiten

Streudiagramm dargestellt:

[14] Greene 2003, zitiert nach (Backhaus et al. (2015):S. 31)

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

T = np.array([0.2408, 0.6152, 1, 1.8808, 11.8626, 29.4475, 84.0168, 164.7913])
r = np.array([0.3871, 0.7233, 1, 1.5237, 5.2034, 9.5371, 19.1913, 30.0690])
plt.scatter(T, r)
plt.show()
```

Diese Visualisierung kann einen Zusammenhang durch eine stetige Funktion vermuten lassen. Welche Funktion könnte das sein? Wenden wir zur Erforschung drei verschiedene Regressionsmodelle an, ein lineares Modell f_1, ein quadratisches Modell f_2 und ein Potenzmodell f_{pot}:

$$f_1(t,a,b) = a + bt, \quad f_2(t,a,b,c) = a + bt + ct^2, \quad f_{\text{pot}}(t,a) = t^a. \tag{5.26}$$

In Python können wir dann die Modellparameter a, b, ... wie folgt mit den Bahndaten der Planeten schätzen:

```
from scipy.optimize import curve_fit
def f1(t,a,b)  : return a + b*t
def f2(t,a,b,c): return a + b*t + c*t**2
def f_pot(t,a) : return t**a
coefs1, cov1       = curve_fit(f1, T, r)
coefs2, cov2       = curve_fit(f2, T, r)
coefs_pot, cov_pot = curve_fit(f_pot, T, r)
```

Die berechneten Parameter lassen sich dann mit den Anweisungen

```
print("Koeffizienten des linearen Modells", coefs1)
print("Koeffizienten des quadratischen Modells", coefs2)
print("Koeffizient des Potenzmodells", coefs_pot)
```

ausgeben. Sie lauten

$$\begin{aligned} \text{Lineares Modell:} \quad & a = 1{,}79, \quad b = 0{,}18 \\ \text{Quadratisches Modell:} \quad & a = 0{,}97, \quad b = 0{,}28 \quad c = -6{,}26 \cdot 10^{-4} \\ \text{Potenzmodell:} \quad & a = 0{,}6667 \approx \tfrac{2}{3}. \end{aligned}$$

Die Anweisungen

```
plt.scatter(T, r)
plt.plot(T, f1(T, *coefs1), label="lineares Modell")
plt.plot(T, f2(T, *coefs2), label="quadratisch Modell")
plt.plot(T, f_pot(T, *coefs_pot), label="Potenzmodell")
plt.legend()
plt.show()
```

liefern das Diagramm in Abbildung 5.10. Man erkennt mit einem Blick, dass die beiden nichtlinearen Modelle den konkaven Kurvenverlauf deutlich besser wiedergeben als das lineare. Welches davon aber besser ist, sieht man nicht sofort. Hier schafft erst ein Vergleich der entsprechenden Bestimmtheitsmaße Klarheit:

```
# 3. Model evaluation
# 3.1 Coefficients of determination:
from sklearn.metrics import r2_score
R2_1   = r2_score(r, f1(T, *coefs1))
R2_2   = r2_score(r, f2(T, *coefs2))
R2_pot = r2_score(r, f_pot(T, *coefs_pot))
```

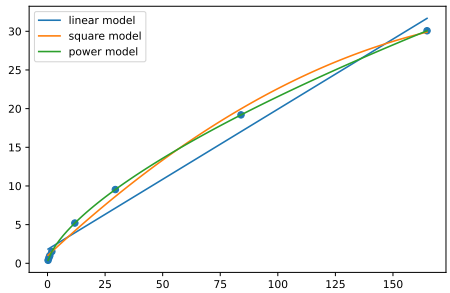


Abbildung 5.10. Bahndaten der Planeten und die drei Regressionsmodelle

```
print("Coefficients of determination")
print(" linear:", f"{R2_1:.3%},", "\tsquare:", f"{R2_2:.3%},", "\tpower: ", f"{R2_3:.3%}")

# 3.2 BICs:
def bic(e, k):
    return np.log(np.var(e)) + k*np.log(len(e))

print(
    " BIC₁ =", f"{bic(y - f1(X, *coefs1), len(coefs1)):.1f},",
    "\t\tBIC₂ =", f"{bic(y - f2(X, *coefs2), len(coefs2)):.1f},",
    "\t\tBIC₃ =", f"{bic(y - f3(X, *coefs3), len(coefs3)):.1f}"
)
```

Es gilt für sie, in Prozent und gerundet auf drei Nachkommastellen:

$$R_1^2 = 97{,}636\%, \qquad R_2^2 = 99{,}618\%, \qquad R_{\text{pot}}^2 = 100\%, \tag{5.27}$$

während sich für das BIC jeweils ergibt:

$$\text{BIC}_1 = 5.0, \qquad \text{BIC}_2 = 5.3, \qquad \text{BIC}_3 = -11.9, \tag{5.28}$$

Das Potenzmodell ist für beide Scorings am besten geeignet! Wir können dies auch visuell bestätigen, indem wir den Plot der drei Modelle mit einer log-log-Skala erstellen:

```
plt.scatter(T, r)
plt.plot(T, f1(T, *coefs1), label="lineares Modell")
plt.plot(T, f2(T, *coefs2), label="quadratisch Modell")
plt.plot(T, f_pot(T, *coefs_pot), label="Potenzmodell")
plt.legend()
plt.xscale("log")
plt.yscale("log")
plt.show()
```

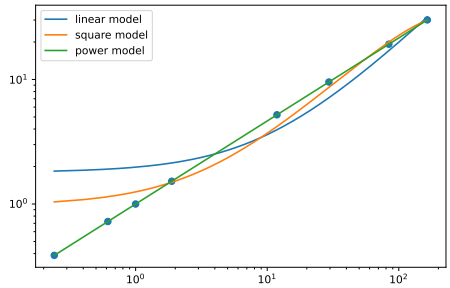


Abbildung 5.11. Die drei Regressionsmodelle in log-log-Skala

In dieser Skala ist deutlich zu erkennen, dass das Potenzmodell einen linearen Verlauf hat und insbesondere die Halbachsenwerte der drei inneren Planeten Merkur, Venus und Erde

durch die beiden anderen Modelle völlig falsch modelliert werden. Das Potenzmodell ist damit anhand der Beobachtungsdaten sehr genau betätigt und liefert die Beziehungen

$$r = T^{2/3} \quad \Longleftrightarrow \quad r^3 = T^2 \quad \Longleftrightarrow \quad \boxed{\frac{r^3}{T^2} = 1.} \tag{5.29}$$

Die letzte Gleichung ist Keplers drittes Gesetz in den Einheiten der Erdbahn ($r_{\earth} = T_{\earth} = 1$). Kepler hatte es bei der Herleitung seines Gesetzes ungleich schwieriger als wir es heute haben: Er brauchte für diese Herleitung ein ganzes Jahrzehnt. Damals gab es allerdings noch gar nicht den Begriff der Funktion, und Gauß erfand seine Methode der kleinsten Quadrate erst fast 200 Jahre später. Nun ja, und Python hatte er auch nicht zur Verfügung. Was hier also als relativ kurz abhandelbares Anwendungsbeispiel für ein Regressionsmodell erscheint, war in Wirklichkeit eine wissenschaftliche Meisterleistung und hat Geschichte geschrieben. Etwa 80 Jahre später entwarf Newton auf Grundlage des Kepler'schen Modells seine bahnbrechende Theorie der Gravitation, und Einstein erklärte etwas über 300 Jahre später einen winzigen Fehler des Modells für die Bahn des inneren Planeten Merkur, die „Perihelbewegung", mit seiner Allgemeinen Relativitätstheorie. Zu Keplers Lebzeiten waren übrigens die beiden äußeren Planeten Uranus und Neptun noch nicht bekannt.

5.4.5 Konfidenzintervalle angepasster Modelle

Die Anpassung der Parameter θ_i eines Regressionsmodells kann man auch als Schätzung für die unbekannten „wahren" Werte der Parameter betrachten. Denn das Ergebnis liefert die beste Schätzung für die unbekannten Parameter anhand der beobachteten Daten. In der Statistik spricht man von einer Parameterschätzung anhand einer Stichprobe (engl.*sample*).

Ein *Konfidenzintervall* gibt den Bereich an, in dem mit einer gewissen Wahrscheinlichkeit der „wahre" Wert einer Zufallsgröße (genauer: einer Zufallsvariablen) um einen Stichprobenwert liegt. Diese Wahrscheinlichkeit heißt *Konfidenzniveau*[15] und wird oft mit $1 - \alpha$ bezeichnet, wobei α die Fehlerwahrscheinlichkeit ist. Konfidenzintervalle sind

Konfidenzintervall und Konfidenzniveau von Zufallswerten

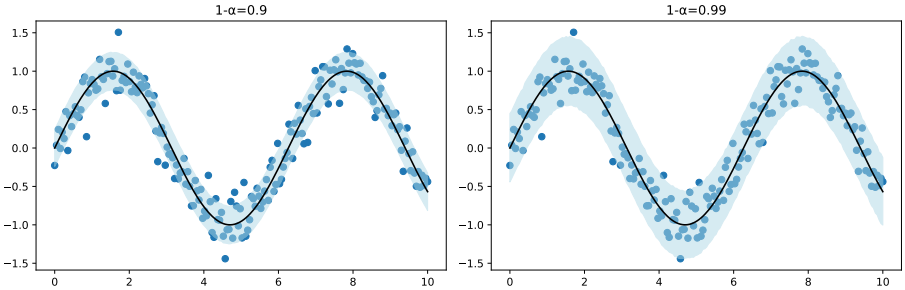


Abbildung 5.12. Konfidenzintervalle um eine Regressionskurve zu den Konfidenzniveaus $1 - \alpha = 0,9$ und $0,99$. Je größer das Konfidenzniveau $1 - \alpha$ ist, desto breiter ist das Konfidenzintervall.

also eine weitere Möglichkeit, die Anpassungsgüte einer Regression für gegebene Datenpunkte zu messen. Wir können sie gut visualisieren, indem wir sie als ein Band um eine Regressionskurve plotten, wie in Abbildung 5.12 für verschiedene Konfidenzniveaus dargestellt.

In Python eignen sich dazu die Numpy-Funktion `quantile` und die Plotfunktion `fill_between`. Übergibt man `quantile` eine Matrix `sims` und eine Liste `[alpha, 1-alpha]`,

[15] Backhaus et al. (2016):§1.2.4.2.

so werden die beiden Quantile $q = \alpha$ und $q = 1 - \alpha$ zurückgegeben. Mit dem optionalen Parameter `axis=0` werden die Zeilen der Matrix `preds` als zusammengehörige Datensätze betrachtet:

```
x = np.linspace(0,10,100)
y_pred = np.sin(x)                # angenommene Regressionskurve
y = np.random.normal(y_pred,0.2) # simulierte Daten
sims = np.array([np.random.normal(y_pred,0.2) for j in range(1000)])

alpha = 0.01                      # Fehlerwahrscheinlichkeit
u, l = np.quantile(sims, [alpha, 1 - alpha], axis=0)

plt.scatter(x,y)
plt.plot(x,y_pred, color="black")
plt.fill_between(x, l, u, color="lightblue", alpha=0.5)
```

Hier enthält `y_pred` die Werte der angenommenen oder ermittelten Regressionskurve und `y` simulierte „beobachtete“ Daten, die mit einer Standardabweichung $\sigma = 0,2$ um die y-Werte gestreut sind. Die Matrix `sims` enthält 100 solcher Zufallssimulationen. Mit `np.quantile` werden die Quantile $q = \alpha$ und $q = 1-\alpha$ in den Variablen `u` und `l` gespeichert und in der Plotfunktion `plt.fill_between` als Ober und Untergrenze eingesetzt. (Beachten Sie, dass der Parameter `alpha` in Plotfunktionen die Opazität des zu zeichnenden Objekts darstellt!) Bei einer Regressionsanalyse mit `curve_fit` muss für die Werte von `y_pred` statt der hier statisch definierten Funktion `np.sin` natürlich die gefittete Modellfunktion verwendet werden, und für `sigma` die ermittelte Standardabweichung der Datenreihen `y` und `y_pred`, statt der Konstanten 0,2:

```
y_pred = f(x,*params)
sigma = np.std(y - y_pred)
```

5.5 Übungsaufgaben

Aufgabe 5.1. Under `https://www.itl.nist.gov/div898/strd/nls/nls_main.shtml`, the NIST provides some challenge datasets for testing robustness and reliability of statistical software with varying levels of difficulty. The “Thurber problem” is classified to be of higher level of difficulty. The data involve semiconductor electron mobility, where the independent variable x is the natural logarithm of the density and the target variable y a measure of electron mobility. The data are available under `https://www.itl.nist.gov/`

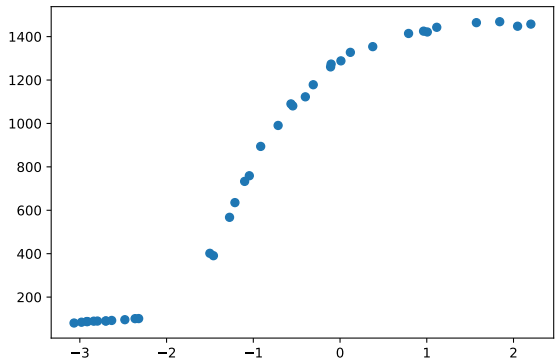


Abbildung 5.13. Semiconductor electron mobility y versus the log electron density x.

`div898/strd/nls/data/thurber.shtml`, or in Moodle. Test whether Python's `curve_fit` function fits the model

$$f(x, \boldsymbol{\theta}) = \frac{\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3}{1 + \theta_4 x + \theta_5 x^2 + \theta_6 x^3}$$

correctly, with initial values $(1000, 1000, 400, 40, 1, 0.5, 0.05)$. Plot the regression curve along with a scatter plot of the data. Evaluate goodness of fit of the model.

Aufgabe 5.2. [16] Die Marketingleitung eines Unternehmens möchte herausfinden, welcher Zusammenhang zwischen den Werbeausgaben und den Absatzmengen eines bestimmten Produkts besteht. Dazu teilt es per Zufall 15 verschiedenen Verkaufsgebieten unterschiedliche Werbebudgets zu und erfasst am Quartalsende die jeweiligen Absatzmengen, siehe Tabelle 5.2. Schätzen Sie für diese Daten die Parameter der folgenden vier Regressi-

Verkaufsgebiet	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Werbeausgaben	22	8	14	26	12	2	10	24	6	30	16	20	18	4	28
Absatzmenge	264,9	176,1	222,7	269,3	194,5	101,7	187,9	275,3	148,4	308,3	241,0	265,9	243,4	129,3	288,6

Tabelle 5.2. Daten zur Untersuchung der Wirkung von Werbung. Die Werbeausgaben sind in 1000 € angegeben, die Absatzmengen in 1000 kg.

onsmodelle. Hierbei sollen die Werbeausgaben die unabhängige Variable repräsentieren. Verwenden Sie außerdem die die Anfangswerte der Parameter, sofern angegeben.

Modell	**Funktion**	**Anfangswerte**
lineares Modell:	$f(x) = a + bx$	
Quadratwurzelmodell:	$f(x) = a + b\sqrt{x}$	
Potenzmodell:	$f(x) = a + bt^c$	$a = 1$, $b = 1$, $c = 0{,}5$
Exponentialmodell:	$f(x) = a + b\,\mathrm{e}^{cx}$	$a = 1$, $b = -1$, $c = 0{,}05$

Bewerten Sie die vier geschätzten Modelle anhand ihrer Anpassungsgüte. Welches Modell erklärt also am besten den Zusammenhang zwischen Werbeausgaben und Verkaufserfolg?

[16] Backhaus et al. (2015):S. 27ff.

6 Datenanalyse mit Python

Kapitelübersicht

6.1 Parametrische statistische Modelle in Python 80
6.1.1 Generalisierte lineare Modelle 81
6.1.2 Wahl der geeigneten Modellklasse 83
6.2 Hauptkomponentenanalyse 83
6.2.1 Fallbeispiel für Hauptkomponentenanalyse 85
6.2.2 Evaluating of predictions 90
6.3 Die Pipeline: Automatisierung der Datenanalyse 90

6.1 Parametrische statistische Modelle in Python

Einen Überblick über gängige Modelle und zugehörige Python-Klassen in Python gibt Tabelle 6.1.

Modell	**Klasse („Estimator“)**	**Modul**
Lineare Regression	`LinearRegression()`	`sklearn.linear_model`
Nichtlineare Regression	`SVR(kernel="rbf")`, `curve_fit`	`sklearn.svm`, `scipy.optimize`
Regression von Zeitreihen	`SARIMAX`$((p,d,q)\times(P,D,Q)_s)$	`statsmodels.tsa.statespace.sarimax`
Klassifikation		
Logistische Regression	`LogisticRegression()`	`sklearn.linear_model`
Support Vector Machines	`LinearSVC()`, `SVC()`	`sklearn.svm`
Naiver Bayes-Klassifikator	`GaussianNB()`	`sklearn.naive_bayes`
Clusterung	`MeanShift()`	`sklearn.cluster`
Louvain-Methode	`best_partition()`	`community`
Neuronale Netze	`Sequential()`, `Model()`	`keras.layers`, `keras.models`
Dimensionsreduktion		
Hauptkomponentenanalyse	`PCA()`	`sklearn.decomposition`
Faktorenanalyse	`FactorAnalysis()`	`sklearn.decomposition`
Lineare Diskriminanzanalyse	`LinearDiscriminantAnalysis()`	`sklearn.discriminant_analysis`
Quadratische Diskriminanzanalyse	`QuadraticDiscriminantAnalysis()`	`sklearn.discriminant_analysis`

Tabelle 6.1. Gängige Modelle zur Datenanalyse in Python.

6.1.1 Generalisierte lineare Modelle

Ein *generalisiertes lineares Modell (generalized linear model, GLM)* [1] ist ein Modell, in dem die Zielvariable y eine Linearkcombination der Merkmale $\boldsymbol{x} = (x_1, \ldots, x_n)$ ist, die durch eine invertierbare Funktion $h : \mathbb{R} \to \mathbb{R}$ transformiert wird:

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = h\left(\theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n\right). \tag{6.1}$$

Die Funktion h heißt *Antwortfunktion* des Modells, seine Inverse h^{-1} heißt *Linkfunktion*. Verallgemeinerte lineare Modelle erlauben Ziekvariablen mit beliebigen Verteilungen, es muss sich also nicht unbedingt um eine normalverteilte Variablen handeln. So umfassen GLM's speziell die Verteilungen in Tabelle 6.2. Unter diese Klasse fallen Regressionsmo-

Verteilung	**Wertebereich**	**Antwortfunktion**	**Einheitsdevianz $d(y, \hat{y})$**
Normal	$y \in (-\infty, \infty)$	$h(x) = x$	$(y - \hat{y})^2$
Bernoulli	$y \in \{0, 1\}$	$h(x) = \dfrac{1}{1 + \mathrm{e}^{-x}}$	$2(y \ln \frac{y}{\hat{y}} + (1 - y) \ln \frac{1-y}{1-\hat{y}})$
Poisson	$y \in \{0, 1, 2, \ldots\}$	$h(x) = \mathrm{e}^x$	$2\left(y \ln \frac{y}{\hat{y}} - y + \hat{y}\right)$

Quellen: https://en.wikipedia.org/wiki/Generalized_linear_model, https://bookdown.org/egarpor/PM-UC3M/glm-model.html, https://scikit-learn.org/stable/modules/linear_model.html#generalized-linear-models, https://scikit-learn.org/stable/modules/model_evaluation.html#log-loss

Tabelle 6.2. Einige generalisierte lineare Modelle (6.1). Hier gilt $\hat{y} = f(\boldsymbol{x}, \boldsymbol{\theta})$.

delle wie die lineare Regression, aber auch Klassifikationen wie die logistische Regression oder sogenannte Support Vector Machines (SVM). Um gefittete Modelle zu bewerten, wird die sogenannte *Devianz* als Maß verwendet. Sie ist für ein generalisiertes lineares Modell durch folgende Gleichung definiert:[2]

$$D(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{i=1}^{m} d(y_i, \hat{y}_i) \tag{6.2}$$

wobei $d(y, \hat{y})$ die *Einheitsdevianz* bezeichnet. Einige Devianzen sind in Tabelle 6.2 aufgeführt. In dem Python-Modul `scikit-learn.linear_models` ist der Vektor $(\theta_1, \ldots, \theta_n)$ mit `coef_` bezeichnet, der Aufpunkt θ_0 mit `intercept_`.

Die Modelle in `scikit-learn.linear_models` unterscheiden sich in dem zu minimierenden Abstandbegriff

$$\operatorname{dist}\left(\boldsymbol{y}, f(\boldsymbol{X}, \boldsymbol{\theta})\right) = D\left(\boldsymbol{y}, f(\boldsymbol{X}, \boldsymbol{\theta})\right) + r(\boldsymbol{\theta})$$

wobei $r : \mathbb{R}^k \to [0, \infty)$ die „Straffunktion" *(penalty function)* oder *Regularisierungsfunktion* des Modells ist. Die Abstandsfunktion misst also geometrisch gesehen den Abstand der m Modellwerte $f(\boldsymbol{X}, \boldsymbol{\theta})$ der Punktewolke zu den m Zielwerten $\boldsymbol{y}$ der Stichprobe, der im wesentlichen die Devianz korrigiert um einen Strafterm ist. Durch einen Modell-Fit wird diese Funktion minimiert. Die gängigsten Modelle sind in Tabelle 6.2 aufgeführt. Sie sind in Scikit-learn wie folgt implementiert:[3]

- `LinearRegression`: Für eine normalverteilte Zielvariable y erhalten wir aus der ersten Zeile von Tabelle 6.2 die Einheitsdevianz und damit die Deviance D mit

[1] `https://scikit-learn.org/stable/modules/linear_model.html#generalized-linear-models`

[2] `https://bookdown.org/egarpor/PM-UC3M/glm-deviance.html`

[3] cf. `https://scikit-learn.org/stable/modules/linear_model.html`

Gleichung (6.2). Mit der Regularisierungsfunktion $r(\boldsymbol{\theta}) = 0$ ergibt sich dann die Abstandsfunktion

$$\text{dist}\big(\boldsymbol{y}, f(\boldsymbol{X},\boldsymbol{\theta})\big) = |\boldsymbol{y} - f(\boldsymbol{X},\boldsymbol{\theta})|^2, \tag{6.3}$$

also einfach die mittlere Quadratsumme der Abstande. Dies ist das RSS der üblichen linearen Regression, vgl. (5.4).

- `Ridge(alpha=`α`)`, oft „L2“ genannt: Abstand

$$\text{dist}\big(\boldsymbol{y}, f(\boldsymbol{X},\boldsymbol{\theta})\big) = |\boldsymbol{y} - f(\boldsymbol{X},\boldsymbol{\theta})|^2 + \alpha\,|\boldsymbol{\theta}|^2 \tag{6.4}$$

durch mittlere Quadrate mit Regularisierungsfaktor α auf die Größe der Parameter $\boldsymbol{\theta}$. Der Regularisierungsfaktor verhindert ein pathologisches Aufschaukeln der Parameterwerte.

- `Lasso(alpha=`α`)`, oft „L1“ genannt. Hier ist die Abstandsfunktion durch

$$\text{dist}\big(\boldsymbol{y}, f(\boldsymbol{X},\boldsymbol{\theta})\big) = \frac{1}{2m}|\boldsymbol{y} - f(\boldsymbol{X},\boldsymbol{\theta})|^2 + \alpha\,|\boldsymbol{\theta}|, \tag{6.5}$$

definiert, ähnlich wie Ridge. Sie eignet sich jedoch besser für Merkmale mit vielen Nullwerten.[4]

- `LogisticRegression`: Obwohl ein lineares Klassifikationsmodell, zählt die logistische Regression auch zu den generalisierten linearen Modellen. Seine Zielvariable hat eine Bernoulli-Verteilung (binäre Werte für y). Für den Defaultwert des Parameters `penalty='l2'` folgt $r(\boldsymbol{\theta}) = \frac{1}{2}|\boldsymbol{\theta}|^2$, und der Abstand

$$\text{dist}\big(\boldsymbol{y}, f(\boldsymbol{X},\boldsymbol{\theta})\big) \;=\; -y \ln h(\boldsymbol{X\theta}) - (1-y)\ln h(\boldsymbol{X\theta}) \;+\; \tfrac{1}{2}|\boldsymbol{\theta}|^2$$

zwischenden Datenwerten und den Modellwerten wird minimiert.[5] Die Devianz wird aus der zweiten Zeile von Tabelle 6.2 hergeleitet. (Hier wird der binäre Character von y ausgenutzt, da für $y = 0$ beide Summanden der Einheitsdevianz verschwinden.)

Daneben gibt es weitere generalisierte lineare Modelle in Scikit-Learn, Es gibt mehrere SVM's in Scikit-Learn, die gängigsten sind die folgenden:[6]

- `LinearSVC`: Lineare Support Vector Klassifikation.
- `SVC`: Nichtlineare Support Vector Klassifikation.

Die folgenden naiven Bayes-Klassifikatoren sind implementiert:[7]

- `GaussianNB`: Gauß-Verteilung (kontinuierliche Zielwerte)
- `BernoulliNB`: multivariate Bernoulli-Verteilung (nur binäre Zielwerte)
- `MultinomialNB`: multinomiale Modelle (mehrere diskrete Zielwerte)
- `ComplementNB`: Complement Naive Bayes Klassifizierer (ähnlich wie `MultinomialNB`, geeignet für sehr ungleichmäßig verteilte Daten)

[4] vgl. `https://scikit-learn.org/stable/auto_examples/applications/plot_tomography_l1_reconstruction.html`

[5] `https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression`

[6] Vgl. `https://scikit-learn.org/stable/modules/svm.html`

[7] Vgl. `https://scikit-learn.org/stable/modules/classes.html#module-sklearn.naive_bayes`

6.1.2 Wahl der geeigneten Modellklasse

Oft ist der schwierigste Schritt bei der Lösung eines Problems mit maschinellem Lernen, das geeignete Modell zu finden. Das liegt nicht nur an der großen Auswahl aus dem „Zoo“ an Modellen in Scikit-Learn, sondern eher daran, die genaue Fragestellung dahinter zu erkennen. Ist es ein Klassifikationsproblem, oder eher ein Clusterungsproblem? Kann man eine Dimensionsreduktion anwenden?

Im Tutorial von Scikit-Learn wird der in Abbildung 6.1 dargestellte Entscheidungsbaum als grobe Handlungsanweisung gegeben. Für uns ist er in diesem Skript zwar nicht bis ins letzte Detail relevant, da wir hier insgesamt nur eine Einführung geben und nicht so weit in die Tiefe gehen können. Als erster Orientierungsrahmen – und als Referenz für weitere Spezialisierungen – kann er uns dennoch gut dienen. Vor allem ist eine gute Hand-

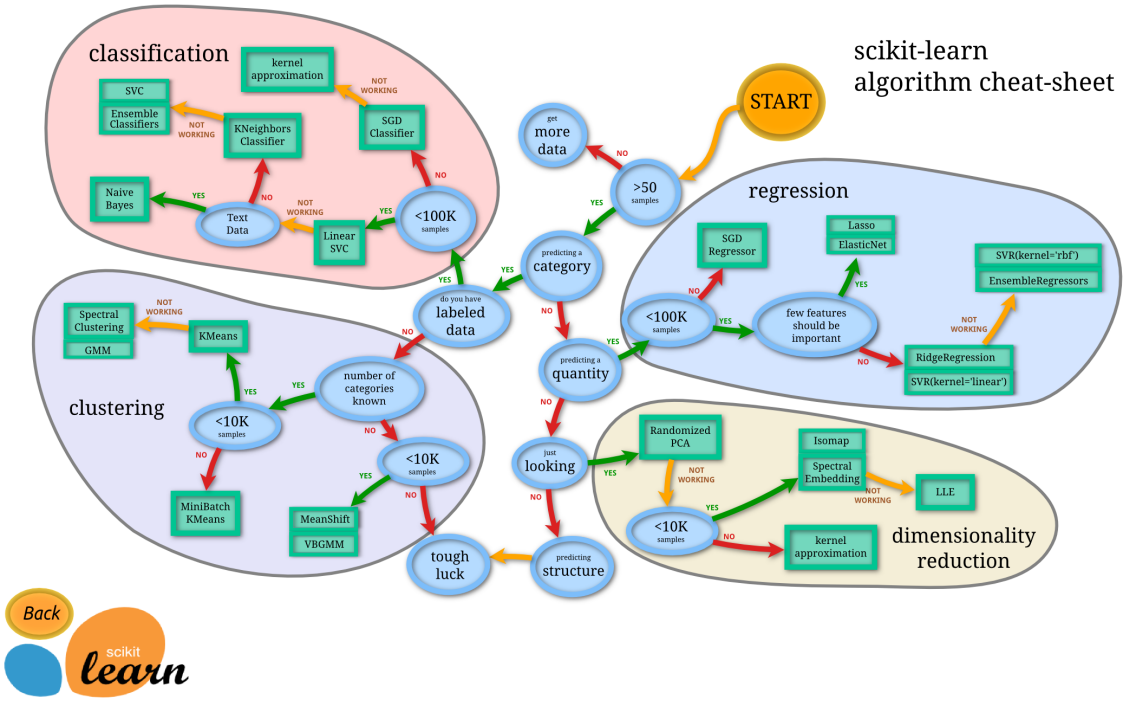


Abbildung 6.1. Entscheidungsbaum zur Ermittlung des geeigneten Modells. Quelle: `http://scikit-learn.org/stable/tutorial/machine_learning_map/`

lungsempfehlung für das Auffinden der eigentlichen Problemklasse, die zu bearbeiten ist: Ist es eine Regression, eine Klassifikation, eine Clusterung oder eine Dimensionsreduktion?

6.2 Hauptkomponentenanalyse

Die Idee der Hauptkomponentenanalyse (PCA, *principal component analysis*) besteht darin, die Informationen eines Datensatzes in seinen Hauptkomponenten zusammenzufassen. Diese Komponenten sind Linearkombinationen der Variablen in ihrer „interessantesten“ Richtung. „Interessant“ meint hier die Richtung der größten Varianz. Das ähnelt einer linearen Regression, aber anstatt die Ergebnisse auf eine Linie zu projizieren, die die x-Achse verwendet, um so viele Informationen wie möglich über y zu erfassen, verwenden wir beide Variablen und versuchen, so viele Informationen wie möglich in der x-Richtung

und der y-Richtung zu erfassen, die die größte Varianz aufweist. Dieser Unterschied zwi-

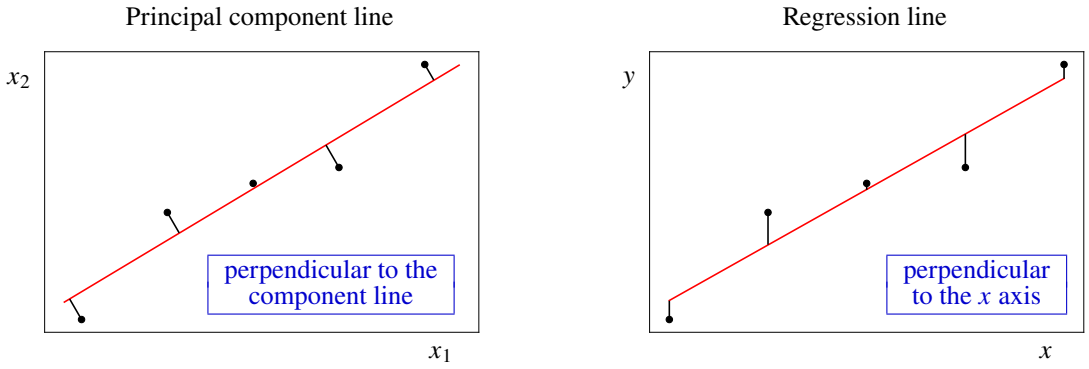


Abbildung 6.2. The subtle difference between the notions of distance for PCA and regression.

schen den Variablenladungen einer Hauptkomponentenlinie und den Variablengewichten einer Regressionslinie, der auf die unterschiedlichen Abstandsbegriffe zurückzuführen ist, wird in Abbildung 6.2 skizziert.

Now, whereas the differences between the x-values and the regression line, the residuals, contribute to the mean error, the differences between the x-values and the principal component line determines the remaining values that contributes to the *second* principal component. The then remaining value contribute to the *third* one, and so on. In principle, there are as many principal components as features (as long as we have more observations than features, i.e., $m > n$). With the notations of Equations (3.2) and (5.2) each single observation of the n features is given by a row vector $\boldsymbol{X}_i = (x_{i1}, \ldots, x_{in})$, i.e., the j-th principal component is given by[8]

$$z_{ij} = \phi_{1j}\, x_{i1} + \phi_{2j}\, x_{i2} + \ldots + \phi_{nj}\, x_{in} \qquad (i = 1, \ldots, m,\ j = 1, \ldots, n) \tag{6.6}$$

where the numbers $\phi_{ij} \in \mathbb{R}$, $i = 1, \ldots, n$, are called the *loadings* of the n features with respect to the principal component and solve the optimization

$$\max_{\phi_{1k},\ldots,\phi_{nk}} \left\{ \sum_{i=1}^{m} \left(\sum_{j=1}^{n} \phi_{jk}\, x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{n} \phi_{jk}^2 = 1 \tag{6.7}$$

for each observation $i = 1, \ldots, m$. The values z_{ik} in Equation (6.6) are called *scores* of the m observations, and the vectors $\boldsymbol{Z}_j = (z_{1j}, \ldots, z_{mj})^{\mathrm{T}}$ are called *score vectors*. The loading vector $\boldsymbol{\phi}_j = (\phi_{1j}, \ldots, \phi_{nj})$ defines the direction of the j-th principal in feature space along which the data vary the most, up to the principal components smaller than j, all direction being orthogonal to each other.[9] Once we successively have computed the principal components, starting from $k = 1$, then going to $k = 2$, and so on, until we reach $k = K$ for some $K \leqq n$, we can rewrite the loading vectors as a matrix and obtain the following relation to the score vectors $\boldsymbol{Z}_k$:

$$\begin{pmatrix} | & & | \\ \boldsymbol{Z}_1 & \cdots & \boldsymbol{Z}_K \\ | & & | \end{pmatrix} = \begin{pmatrix} — & \boldsymbol{X}_1 & — \\ & \vdots & \\ — & \boldsymbol{X}_m & — \end{pmatrix} \cdot \Phi^{\mathrm{T}} \qquad \text{with} \qquad \Phi^{\mathrm{T}} = \begin{pmatrix} | & & | \\ \boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_K \\ | & & | \end{pmatrix} \tag{6.8}$$

[8] James et al. (2013):S. 376.

[9] From a mathematical point of view, the n loading vectors $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n$ are the ordered sequence of eigenvectors of the $n \times n$ matrix $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$, given the notation of (3.2), and the variances of the components are its eigenvalues. (Footnote on James et al. (2013):S. 377)

where $\boldsymbol{X}$ is the matrix in Equation (3.2). In Scikit-learn the matrix Φ is called "components",[10] so we will refer to it as the component matrix, or load matrix. We can plot the principal components pairwise against each other to view the data with respect to them. Such a projection of the data on a pair of principal components, together with the positions of the feature vectors X_i, is called a *biplot*.[11] Besides the observations themselves, the feature variables are drawn as arrows from the origin. The coordinates of each feature with respect to the principal components are given by the respective row of the load matrix Φ, or in Scikit-learn by one of the columns of the component matrix Φ. A biplot is given below in the right-hand panel of Figure 6.3.

biplot

The property of the first principal component being the line in k-dimensional feature space that is *closest* to the m observations, as indicated in Figure 6.2, can be generalized. For instance, the first two components of a data set span the plane that is closest to the m observations. Moreover, the first three principal components of a data set span the three-dimensional space (as a hyperplane of a higher-dimensional space for $k > 3$) closest to the m observations, and so forth.[12] Therefore, the main goal of PCA usually is not to map the n feature axes to n principal component axes, but to find a *lower dimensional* representations of the observations that explain a good fraction of the variance. Therefore, only the first few principal components should be regarded. The crucial question then is: How much of the information in a given data set gets lost by projecting the observations in the first few principal components?

There is no definite method or criterion to determine, how many principal components are necessary such that "enough" information is projected onto them. However, a common step to decide this question is to visualize the proportions of explained variances by a *scree plot*.[13]

scree plot

Eine schöne und humorvolle Einführung in die PCA finden Sie im Beitrag von Harriet Mason `https://numbat.space/posts/pca/`.

Before PCA can be performed, the feature variables should be centered to have mean zero. Moreover, since the variables usually are individually scaled – one feature may be in units of currency, another one may be in tons or kg – they should be scaled to standard deviation.[14] In Scikit-learn, scaling the data is conveniently done with the Transformer `StandardScaler` of the module `sklearn.preprocessing`.

scaling data

6.2.1 Fallbeispiel für Hauptkomponentenanalyse

Der World Happiness Report wird jährlich von den Vereinten Nationen veröffentlicht, für 2018 zum Beispiel unter `http://worldhappiness.report/ed/2018/`, Chapter 2, Figure 2.2. In dem Bericht wird für fast sämtliche Nationen der Erde der Happiness-Index als Kennzahl aus verschiedenen ökonomischen und gesellschaftlichen Merkmalsdaten wie soziale Leistungen, Lebenserwartung, Grad an Freiheit oder Korruption ermittelt. Mit dem folgenden Beispielprogramm wollen wir der Frage nachgehen, welche dieser Merkmale anhand der Daten den wesentlichen Einfluss auf die Zufriedenheit einer Bevölkerung haben.

[10] Géron (2017):S. 213–214.
[11] James et al. (2013):S. 377.
[12] James et al. (2013):S. 380–381.
[13] James et al. (2013):S. 382–383.
[14] Géron (2017):S. 213; James et al. (2013):S. 381–382.

Erste Visualisierung der Hauptkomponenten

As a first step in data analysis it is common to visualize the underlying data. For this purpose we first import the data as usual, before we perform the principal component analysis with Scikit-learn. According to the discussion above, this will yield us the calculation of the

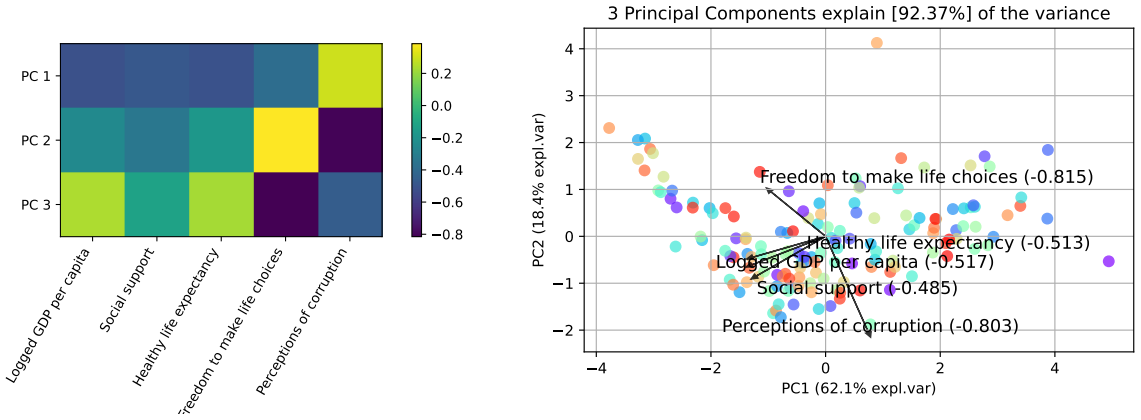


Abbildung 6.3. Left: Heatmap of the loads of the first three principal components with respect to the features. Right: Biplot of the five features projected into the plane spanned by the first two principal components.

principal components as a matrix Φ, showing how the features contribute to the first few principal components.

```
%matplotlib notebook
import matplotlib.pyplot as plt, numpy as np, pandas as pd, mpl_toolkits.mplot3d
from sklearn.preprocessing   import StandardScaler   # scales data to standard deviation
from sklearn.decomposition   import PCA              # principal component analysis
from sklearn.linear_model    import LinearRegression # linear regression

# 1. Import data as a Pandas DataFrame and preprocess them for scikit-learn:
df = pd.read_csv("./datasets/happiness-report-2021.csv", sep='\t')  # loads CSV to Pandas
features = ["Logged GDP per capita", "Social support", "Healthy life expectancy",
  "Freedom to make life choices", "Perceptions of corruption"]
target = "Happiness"                # dependent variable
X = np.c_[df[features]]             # extracts feature values as a matrix
y = np.c_[df[target]]               # extracts target values as a one-column matrix

# 2.1 Scaling the data
model1 = StandardScaler()
model1 = model1.fit(X)
X_scaled = model1.transform(X) # compute and store the scaled data

# 2.2 Principal component analysis of the feature data
model2 = PCA(3) # for example, three principal components ...
model2 = model2.fit(X_scaled, y)
X_trans = model2.transform(X_scaled) # compute and store the projected data

# 2.3 Print and plot the principal components as a heat map
print(model2.components_)
plt.figure(figsize=(7,4))
plt.imshow(model2.components_)
plt.colorbar()
plt.xticks(range(len(features)), features, rotation=60, ha="right")
```

```
yticks = [""] * len(model2.components_[:,0])
for i in range(len(yticks)):
    yticks[i] = "PC " + str(i+1)
plt.yticks(range(len(yticks)), yticks)
plt.show()

# -------------------------------------
# 3 Biplot:
from pca import pca

model = pca(n_components=3, verbose=0)
results = model.fit_transform(X_scaled, col_labels=features, row_labels=df['Country name'],
    verbose=0)

fig, ax = model.biplot(
    figsize=(10,5), PC=[0,1], alpha_transparency=0.7, SPE=True,
    cmap="rainbow", color_arrow='black', verbose=0, label=None,
)
fig.show()
```

The variances – or in other words the correlations[15] – of the individual characteristics on the principal components are given by the matrix <

$$\Phi = \begin{pmatrix} -0.517 & -0.485 & -0.513 & -0.386 & 0.293 \\ -0.244 & -0.340 & -0.180 & 0.385 & -0.803 \\ 0.238 & -0.123 & 0.225 & -0.815 & -0.462 \end{pmatrix} \tag{6.9}$$

Each column yields the loading vector of the corresponding feature. The loadings can be represented by a heat map as shown in Figure 6.3 on the left. The larger the absolute amount of variance, the greater is the influence of the particular characteristic. Negative amounts are negatively correlated with the principal component, but this has no deeper meaning in principle since only the direction of the principal component is relevant. Figure 6.3 on the right depicts the happiness index of each country plotted against the first two principal components. The biplot is generated by the library pca. Here the coordinates of the five feature vectors with respect to the three principal components are given by the columns of the component matrix Φ.

To get an intuition of the relationship of the principal components to the data and to the feature vectors, as well as and to understand the underlying geometry, we could plot a three-dimensional biplot (strictly speaking, a "triplot" instead) with the following code snippet.

```
# 3.2 Three-dimensional biplot
fig, ax = model.biplot(
    figsize=(14,7), PC=[0,1,2], d3=True, alpha_transparency=0.7,
    cmap="rainbow", color_arrow='black', verbose=0, label=None, legend=False
)
ax.set_xlim(-4, 4)
ax.set_ylim(-2, 2)
ax.view_init(azim=-40, elev=36) # position of camera
fig.show()
```

This gives the plot in Figure 6.4.

[15] Backhaus et al. (2016):p. 394.

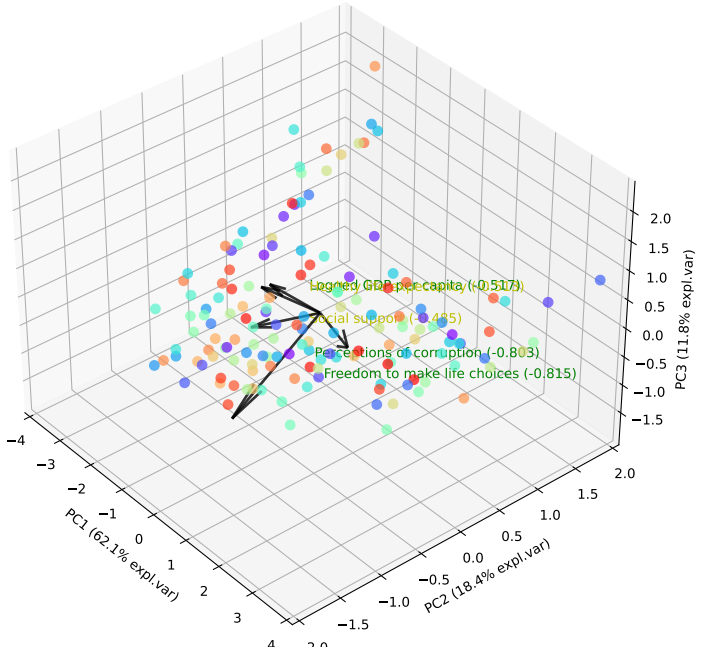


Abbildung 6.4. Biplot with the first three principal components.

Determining the number of components

Now we can tackle problem to determine the number of principal components to characterize the data good enough. A first and not unusual step is to use some hints by regarding the scree plot. For this purpose the following code snippet can be implemented:

```
model = PCA()  # enable all possible principal components ...
model = model.fit(X_scaled,y)
print("Explained variance ratio:", model.explained_variance_ratio_)
plt.figure(figsize=(6,4))
pc_values = np.arange(model.n_components_) + 1
plt.plot(pc_values, model.explained_variance_ratio_, 'o-')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Explained Variance')
plt.xticks(pc_values)
plt.show()
```

This gives the plot in Figure 6.5. Since there are five features (and mopre than 140 data points), the maximum number of principal components is five. The scree plot simply plots the explained variance of each principal component, given by the model attribute `explained_variance_ratio`. The scree plot does not show a definite "elbow", but we may located it at the second component. Therefore, two principal components may be considered to be "enough" to explain the relevant feature combinations. Considering the output of the program

```
Explained variance ratio: [0.6212864  0.18409387 0.11832235 0.0507004  0.02559698]
```

we conclude that the first two components explain $62.13+18.41 = 80.54$ % of the variances.

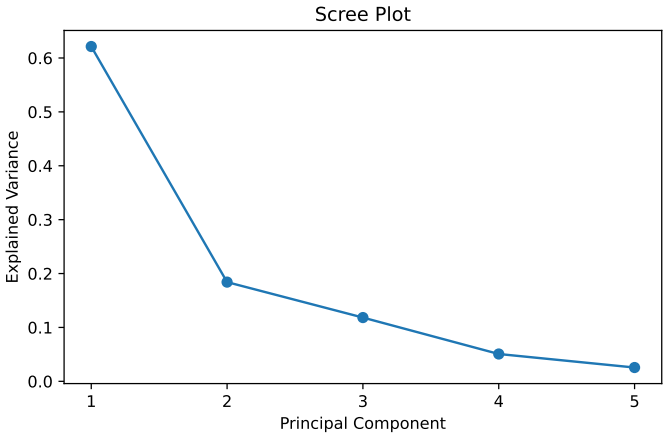


Abbildung 6.5. Scree plot of the possible five principal components. Its "elbow" may be located at the second component.

General overview with scatterplot matrices

A convenient way to give a visualized overview over multidimensional data is to use a *scatterplot matrix*. Here every numerical variable is plotted against every other numerical variable, arranged in a tabular form.[16] In Python, scatterplot matrices can be generated from a Pandas DataFrame by the Pandas function `scatter_matrix` in the module `pandas.plotting`. Another possibility is the module `plotly.express`, which is more

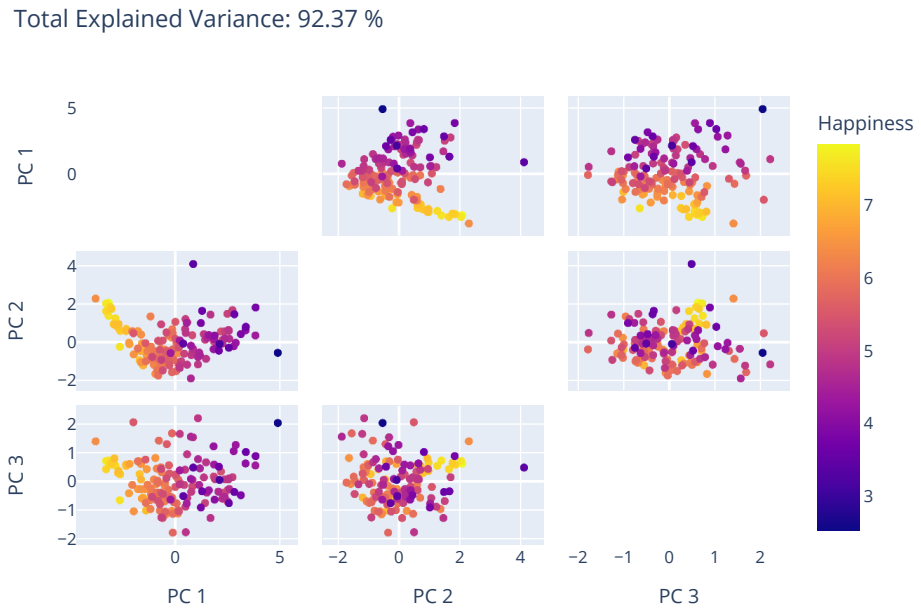


Abbildung 6.6. Scatterplot matrix of the happiness indices of each country, plotted against each pair of the first three principal components PC1, PC2, and PC3.

convenient for Numpy arrays as in Scikit-learn. If `X_trans` represents the observations projected onto the space of the first three principal components, as done in step 2.2 above by `model2`, we could depict the $3 \cdot (3-1) = 6$ plots by the following snippet:

```
import plotly.express as px
```

[16] cf. Géron (2017):S. 57–58; James et al. (2013):S. 50.

```
n_components = model2.n_components
total_var = model2.explained_variance_ratio_.sum() * 100

labels = {str(i): f"PC {i+1}" for i in range(n_components)}
labels['color'] = target

fig = px.scatter_matrix(
    X_trans,
    color=y.ravel(),
    dimensions=range(n_components),
    labels=labels,
    title=f'Total Explained Variance: {total_var:.2f} %',
)
fig.update_traces(diagonal_visible=False)
fig.show()
```

Since the diagonal would give two-dimensional plots showing all observations on a straight line, we suppress them with the option `diagonal_visible=False`. The result of this program gives the scatterplot matrix in Figure 6.6.

6.2.2 Evaluating of predictions

Prognosemodelle können Zielwerte vorhersagen und haben in Scikit-Learn eine Methode `predict`, die für Merkmalswerte als Eingabedaten `X_test` die Zielwerte nach dem Modell berechnet:

```
y_pred = model.predict(X_test)
```

Um die Vorhersagen zu bewerten, kann man die Methode `score` verwenden, die eine für das Modell relevante Kennzahl für die Übereinstimmung des Modells für `X_test` mit den tatsächlichen Werten `y_test` angibt.

```
print( model.score(X_test, y_test) )
```

Man sollte also nicht alle verfügbaren Daten als Trainingsdaten verwenden, sondern einen Teil als Testdaten nutzen. Dafür ist die Methode `train_test_split` geeignet:

```
from sklearn.model_selection import train_test_split
# Zufallsauswahl 30% der Daten als Testdaten (d.h. 70% Trainingsdaten):
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Weitere Möglichkeiten zur Bewertung von Prognosen sind unter `https://scikit-learn.org/stable/modules/model_evaluation.html` aufgeführt.

6.3 Die Pipeline: Automatisierung der Datenanalyse

Möchte man mehrere Modelle und Datentransformationen nacheinander ausführen, so kann man eine sogenannte Pipeline verwenden. Eine *Pipeline* ist eine Sequenz von Modellen mit Methoden `fit` und `transform`, deren letztes Modell nur die Methode `fit` benötigt. Sie dient dazu, aufwendige Modellierungsprozesse zu automatisieren und damit ganze Modellierungsszenarios durchzuspielen.

Beispielsweise können in einer Pipeline ein Skalierer, eine Hauptkomponentenanalyse und ein lineares Regressionsmodell hintereinander geschaltet werden:

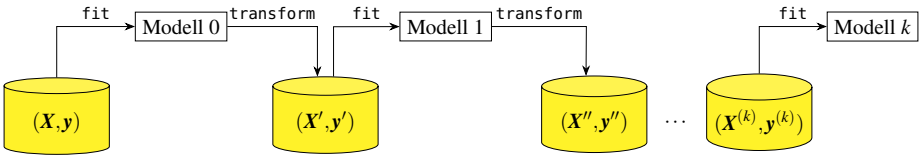


Abbildung 6.7. Prinzip einer Pipeline

```
pipe = make_pipeline(StandardScaler(), PCA(2), GaussianNB())
```

In dem Attribut `steps` der Pipeline sind die Modelle aufgelistet, und zwar als 2-Tupel `('name', modell)`. Z.B. erhält man das Hauptkomponentenmodell mit

```
pca = pipe.steps[1][1]
```

Da das letzte Modell der Pipeline eine Methode `predict` enthält, kann eine Prognose der Zielwerte für weitere Testdaten erstellt werden.

```
y_pred = pipe.predict(X_test)
```

Beispiel 6.1. *(Hauptkomponentenanalyse und lineare Regression mit Pipeline)* Wollen wir statt einer Gauss'schen naiven Bayes-Klassifikation wie in Beispiel 6.2 eine lineare Regression durchführen, so genügt im Prinzip ein Austausch des Modells in der dortigen Pipeline. Da die lineare Regression allerdings reelle Werte der Zielvariablen erwartet, sollte eine Kategorierung der Daten wie in Schritt 3 in Beispiel 6.2 weggelassen werden.

```
%matplotlib notebook
import matplotlib.pyplot as plt, numpy as np, pandas as pd, mpl_toolkits.mplot3d
from sklearn.model_selection import train_test_split # splits data into training and test data
from sklearn.pipeline        import make_pipeline    # makes a pipeline
from sklearn.preprocessing   import StandardScaler   # scales data to standard normal distribution
from sklearn.decomposition   import PCA              # principal component analysis
from sklearn.linear_model    import LinearRegression # linear regression

# 1. Import data as a Pandas DataFrame and preprocess them for scikit-learn:
df = pd.read_csv("./datasets/happiness-report-2021.csv", sep='\t')  # loads CSV file as a Pandas dataframe
features = ["Logged GDP per capita", "Social support", "Healthy life expectancy", "Freedom to make life choices",
            "Perceptions of corruption"]
target = "Happiness"                # dependent variable
X = np.c_[df[features]]             # extracts feature values as a matrix
y = np.c_[df[target]]               # extracts target values as a one-column matrix

# 2. Choose by random 30 % of data as test data, i.e., 70 % as training data:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)

# 3. Fit and predict with a pipeline of scaling, PCA, and linear regression:
pipe = make_pipeline(StandardScaler(), PCA(2), LinearRegression())
pipe.fit(X_train, y_train)

# 4. Print model score:
print("score (train values): ", f"{pipe.score(X_train, y_train):.2%}")
print("score (test values):",f"{pipe.score(X_test, y_test):.2%}")

# 5. Plot 3D scatter plot:
from mpl_toolkits.mplot3d import Axes3D

# 5.1. Choose PCA model from pipeline and project data onto the principal components:
X_scaled = pipe.steps[0][1].fit_transform(X) # scaled data
X_trans  = pipe.steps[1][1].fit_transform(X_scaled) # Dimensionsreduzierung auf die Hauptkomponenten
y_pred  = pipe.predict(X)              # Vorhersagewerte der Pipeline ...

# 5.2. Plot 3D scatter diagram:
fig = plt.figure(figsize=(7,5))
ax = fig.add_subplot(111, projection="3d")
ax.scatter(X_trans[:,0], X_trans[:,1], y, marker="o", c='red', label='actual values')
ax.set_xlabel("PC 1"), ax.set_ylabel("PC 2"), ax.set_zlabel(target)
ax.view_init(azim=-60, elev=20) # position of camera
plt.tight_layout()
```

```
plt.show()

# 5.3. Plot regression plane witht min/max of the transformed data:
x0 = np.linspace(X_trans[:,0].min(), X_trans[:,0].max(), num=2)
x1 = np.linspace(X_trans[:,1].min(), X_trans[:,1].max(), num=2)
xx0, xx1 = np.meshgrid(x0,x1) # 2x2 - Gitter
X0, X1 = xx0.ravel(), xx1.ravel()
yy = pipe.steps[2][1].predict(np.c_[X0, X1]).ravel()  # Prediction values in the regression plane
ax.plot_trisurf(X0, X1, yy, linewidth=0, alpha=0.3)
plt.tight_layout()
plt.show()
```

Das ausgegebene Streudiagramm ist in Abbildung 6.8 wiedergegeben. Rechts sieht man

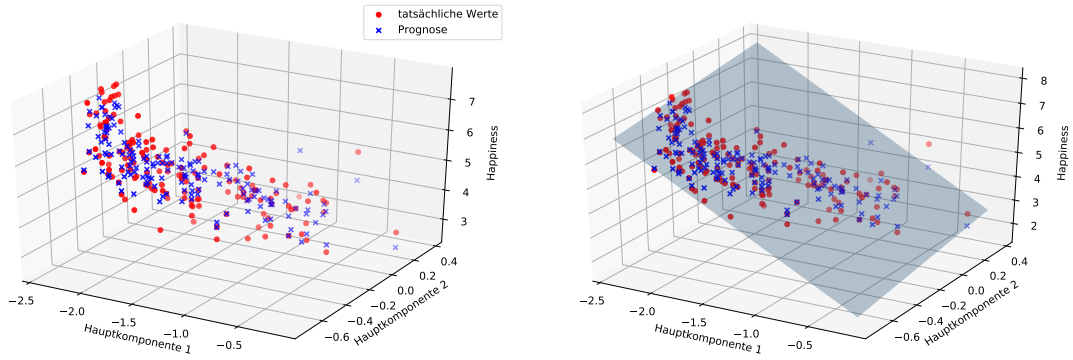


Abbildung 6.8. Streudiagramm der Projektion auf die zwei Hauptkomponenten der tatsächlichen Daten und der nach einer linearen Regression prognostizierten Werte (Beispiel 6.1). Die Prognosewerte liegen in der Regressionsebene (rechts).

das Streudiagramm mit der Regressionsebene, auf der die mit der linearen Regression prognostizierten Werte (hier als x gekennzeichnet) liegen. □

Wie eingangs bereits besprochen kann der Pipeline-Mechanismus leicht dafür verwendet werden, Verkettungen von Modellen zu ändern. Das folgende Beispiel zeigt diese Möglichkeit, indem die lineare Regression durch einen Gauß'scher naiver Bayes-Klassifikator ausgetauscht wird. Hier geschieht dies noch manuell, aber natürlich können wir uns leicht Programme vorstellen, die diesen Mechanismus für eine ganze Liste von Modellen automatisiert gegenseitig austauschen und somit gewissermaßen das Testen ganzer Modellketten ermöglichen.

Beispiel 6.2. *(Hauptkomponentenanalyse und Klassifizierung mit Pipeline)* In dem folgenden Programm wird eine Hauptkomponentenanalyse mit einer anschließenden Klassifizierung als Pipeline durchgeführt.

```
%matplotlib notebook
import matplotlib.pyplot as plt, numpy as np, pandas as pd, mpl_toolkits.mplot3d
from sklearn.model_selection import train_test_split # teilt Daten in Training und Test
from sklearn.preprocessing   import KBinsDiscretizer # teilt stetige Werte in Kategorien
from sklearn.pipeline        import make_pipeline    # erstellt eine Pipeline
from sklearn.preprocessing   import StandardScaler   # normalverteilte Skala für alle Daten
from sklearn.decomposition   import PCA              # Hauptkomponentenanalyse
from sklearn.naive_bayes     import GaussianNB       # Gauss'scher naiver Bayes-Klassifikator

df = pd.read_csv("./datasets/happiness-report-2021.csv", sep='\t')  # lädt CSV-Datei in Pandas
merkmale = ["Logged GDP per capita", "Social support", "Healthy life expectancy", "Freedom to make life choices", "
      Perceptions of corruption"]
ziel = "Happiness"                  # abhängige Variable
X = np.c_[df[merkmale]]             # extrahiert die Merkmalswerte als Matrix
y = np.c_[df[ziel]]                 # extrahiert die Zielwerte

# Wähle per Zufall 30 % der Daten als Testdaten aus, d.h. 70 % als Trainingsdaten:
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)

# Teile die Trainings- und Testdaten der abhängigen Variablen in 3 Kategorien („Bins“):
y_train = KBinsDiscretizer(n_bins=3, encode='ordinal').fit_transform(y_train).ravel()
y_test  = KBinsDiscretizer(n_bins=3, encode='ordinal').fit_transform(y_test).ravel()

# Fitting und Prognose mit Pipeline aus Skalierung, PCA und Gauss Naive-Bayes-Klassifizierer:
pipe = make_pipeline(StandardScaler(), PCA(2), GaussianNB())
pipe.fit(X_train, y_train)

# Vorhersagegenauigkeit angeben:
print("score (train values): ", f"{pipe.score(X_train, y_train):.2%}")
print("score (test values):",f"{pipe.score(X_test, y_test):.2%}")

# -- 3D-Streudiagramm:
X_trans = pipe.steps[0][1].transform(X) # Dimensionsreduzierung auf die Hauptkomponenten!
y_pred  = pipe.predict(X)               # Vorhersagewerte der Pipeline

plt.figure(figsize=(8,5)); ax = plt.axes(projection="3d")
ax.scatter(X_trans[:,0], X_trans[:,1], y, c=y_pred)
ax.set_xlabel("Hauptkomponente 1"); ax.set_ylabel("Hauptkomponente 2"); ax.set_zlabel(ziel)
plt.show()
```

Im Einzelnen werden in dem Programm die folgenden Schritte ausgeführt:

1. Die CSV-Datei wird als Pandas DataFrame importiert und die Daten in Numpy-Arrays `X` und `y` gespeichert.
2. Die Daten `X` und `y` werden in Trainings- und Testdaten separiert.
3. Die nominalen Zieldaten `y_train` und `y_test` für Training und Test werden ordinal in 3 Kategorien geteilt.
4. Eine Pipeline `pipe` wird aus einem Standardisierer, einer Hauptkomponentenanalyse und einem Gauß'schen naiven Bayes-Klassifikator gebildet und mit den Trainingsdaten gefittet.
5. Ein Vektor `y_pred` wird mit den Prognosewerten der Testdaten `X_test` bestimmt.
6. Die Prognosegenauigkeiten des naiven Bayes-Klassifikators werden ausgegeben.

Das Programm gibt damit die folgenden Ergebnisse aus:

```
score (train values): 76.95%
score (actual values): 78.59%
```

Danach werden Anweisungen zum Plotten eines Streudiagramms mit den auf die Hauptkomponenten projizierten Daten durchgeführt:

7. Die originalen Merkmalsdaten `X` mit ihren 5 Dimensionen werden geometrisch betrachtet auf die beiden Hauptkomponenten projiziert und entsprechend die Daten für die gesamten Merkmalsausprägungen anhand der Pipeline vorhergesagt und in der Arrayvariablen `y_pred` gespeichert.
8. Das dreidimensionale Streudiagramm wird geplottet. Die Farbe eines Datenpunktes wird hier mit dem Parameter `c` in `ax.scatter` durch den Wert der durch den Gauss'schen NB-Klassifizierer ermittelten Kategorie bestimmt.

Das ausgegebene Streudiagramm ist in Abbildung 6.9 wiedergegeben. In dem Streudiagramm sind die einzelnen Länder als Datenpunkte dargestellt, die sich farblich aufgrund ihrer jeweils durch den NB-Klassifikator ermittelten Klasse unterscheiden. So erkennt man die Verteilung der in Schritt 3 in drei Kategorien in Abhängigkeit zu den beiden Hauptkomponenten. □

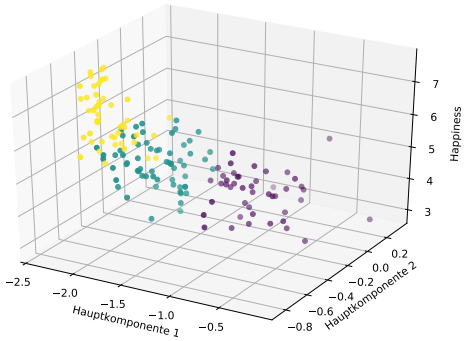


Abbildung 6.9. Streudiagramm der Projektion auf die zwei Hauptkomponenten nach einer Naiven Bayes-Klassifikation (Beispiel 6.2). Die Klassen sind in unterschiedlichen Farben dargestellt.

Teil III

Zeitreihen

7 Zeitreihenanalyse

Kapitelübersicht

7.1 Einführung 96
7.1.1 Zeitreihen und maschinelles Lernen 97
7.1.2 Model forms of time series 97
7.2 Stochastische Prozesse als Grundlage von Zeitreihen 98
7.3 Kausale lineare Prozesse 99
7.4 Die Random-Walk-Hypothese in der Ökonomie 101
7.5 Übungsaufgaben 102

7.1 Einführung

Eine Zeitreihe ist eine zeitabhängige endliche Folge von reellwertigen Datenpunkten y_0, y_1, y_2, ..., $y_t \in \mathbb{R}$, die mit einem den Zeitablauf wiedergebenden Index versehen ist. Ein einzelner Datenpunkt ist dabei ein beobachteter oder gemessener Wert eines bestimmten Prozesses. Typische Beispiele solcher Prozesse sind die Entwicklung von Börsenkursen, Wahlabsichtsbefragungen, Wetterbeobachtungen oder Temperaturwerten. Eine Zeitreihe wird häufig gegen die Zeit aufgetragen, indem die Datenpunkte zu einer Kurve verbunden werden (Abbildung 7.1).[1] Die Zeitpunkte, denen Datenpunkte zugeordnet werden, können äquidistant angeordnet sein, also in konstanten Abständen (beispielsweise alle 5 Sekunden), in anderer Regelmäßigkeit (beispielsweise werktäglich) oder unregelmäßig. Meist werden äquidistante Zeitreihen betrachtet, der Kehrwert der Dauer zwischen zwei Messwerten wird in diesem Fall als Abtastrate *(sampling rate)* bezeichnet.

Abtastrate

Eine Zeitreihe kann zu jedem Zeitpunkt einen einzelnen Zahlwert annehmen, sie heißt dann univariat oder skalar. Nimmt sie dagegen jeweils einen Tupel von mehreren Zahlenwerten an, so heißt sie multivariat. Typische Zeitreihen entstehen aus dem Zusammenwirken regelhafter und zufälliger Ursachen. Die regelhaften Ursachen können periodisch bzw. saisonal variieren und langfristige Trends enthalten. Zufällige Einflüsse werden als Rauschen bezeichnet.

univariat, multivariat

Rauschen

Häufig werden Zeitreihen analysiert, um ihre künftige Entwicklung vorherzusagen. Die Zeitreihenanalyse ist eine spezielle Form der Regressionsanalyse und versucht, mit-

Zeitreihenanalyse

[1] vgl. dazu auch: Burke et al. (2018); Kalvelage (2018).

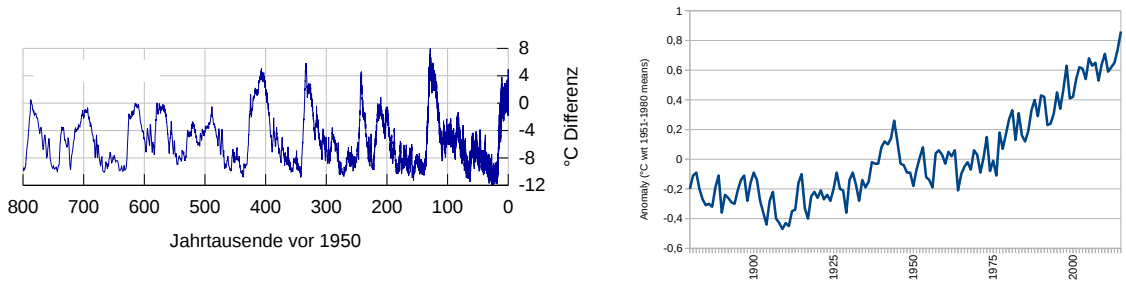


Abbildung 7.1. Links: Temperaturveränderungen in der Antarktis der letzten 800 000 Jahre im Vergleich zum Durchschnitt der letzten 1000 Jahre, durchgeführt durch EPICA (Jouzel et al., 2007). Rechts: Globale Temperaturanomalien 1880 – 2015 (zum Mittelwert 1951–1980), Datenquelle: `http://cdiac.ornl.gov/ftp/trends/temp/hansen/gl_land_ocean.txt`; vgl. auch `https://xkcd.com/1732/`

tels statistischer Verfahren Muster beziehungsweise Regelmäßigkeiten in der Entwicklung von Zeitreihen zu erkennen und zu modellieren. Solche Regelmäßigkeiten sind von ganz prinzipiellem Interesse, da gegebene wissenschaftliche Theorien ganz bestimmte Gesetzmäßigkeiten erwarten lassen.

Die folgenden Fachbücher über Zeitreihen werden für weitere Informationen empfohlen und später an geeigneten Stellen in diesem Skript speziell referenziert:[2].

7.1.1 Zeitreihen und maschinelles Lernen

Zeitreihen spielen eine recht spezielle Rolle im maschinellen Lernen. Bei vielen Problemen hängt die abhängige Variable y, d.h. das, was wir vorhersagen möchten, von sehr klaren Eingaben ab, wie z. B. Pixeln eines Bildes, Wörtern in einem Satz, den Eigenschaften des Kaufverhaltens einer Person usw. In Zeitreihen sind diese unabhängigen Variablen oft nicht bekannt. Beispielsweise haben wir an den Aktienmärkten keine eindeutigen unabhängigen Variablen, auf die wir ein Modell anwenden können. Sind die Aktienmärkte von den Eigenschaften eines Unternehmens oder den Eigenschaften eines Landes abhängig? Oder von der Stimmung in den Nachrichten? Sicherlich können wir versuchen, eine Beziehung zwischen diesen unabhängigen Variablen und den Börsenergebnissen zu finden, und vielleicht können wir einige gute Modelle finden, die diese Beziehungen abbilden. Der Punkt ist, dass diese Beziehungen nicht sehr klar sind und die unabhängigen Daten auch nicht leicht erhältlich sind.

Bei der klassischen Analyse von Zeitreihen geht man den radikal anderen Weg, keine unabhängigen Variablen zu suchen oder anzunehmen, sondern aus der bisherigen Entwicklung der Zeitreihe ihre Eigenschaften abzuleiten. D.h. die einzige unabhängige Variable einer Zeitreihe ist die Zeit.

7.1.2 Model forms of time series

Usually time series analysis is considered in combination with forecasting. However, strictly speaking forecasting and the analysis of time series are two distinct activities. A forecast is a view in an uncertain future. Time series is a description of the past. The main precursors to the forecasting activity are the construction of a suitable model based upon

[2]Deistler und Scherrer (2018); Kreiß und Neuhaus (2006); Neusser (2011); Palma (2016); Vogel (2015).

analysis of the historical development of the series and utilisation of information relevant to the series' likely future development.

That is not to say that the modelling and analysis process is concluded once a forecast has been produced. Eventually the quantity being forecast will become known, this value providing new information to incorporate into the analysis. Thereafter, in a typical live situation, a forecast for a subsequent time will be produced and the forecast-observation cycle repeated.

Three basic model forms encompass the great majority of time series and forecasting situations. They are:

- trend models,
- seasonal models, i.e., models for systematic cyclical variation, and
- regressions, i.e., models with influential or causal variables.

Trend models are the simplest component models. They represent a system with a straightforward linear progression: growing, decreasing, or staying roughly constant. Seasonal models provide the mechanism to model systematic cyclical variation. This kind of variation is often present in commercial series, and typically related to the passage of the seasons as the earth orbits the sun during the course of a year. Regressions on explanatory variables are potentially the most valuable models because they may incorporate much external information.[3]

A forecast is a statement about an uncertain future. It is a belief, not a statement in fact. Representing uncertainty in scientific analysis is the province of probability, its practical application the domain of statistics. Whenever we make a forecast we actually make a statement of probability, or more generally, state a probability distribution that quantifies the nature of our uncertainty. Any and every forecast is predicted upon a fount of knowledge; forecasts are therefore conditional probability statements, the conditioning being in the existing state of the knowledge. Knowledge is available in several forms, a useful classification being into historical information and professional wisdom or expertise.[4]

7.2 Stochastische Prozesse als Grundlage von Zeitreihen

In der Stochastik fasst man die Datenpunkte $y_0, y_1, \ldots, y_t$ einer Zeitreihe als Stichprobe oder konkrete Realisierung eines stochastischen Prozesses auf, also als eine Folge von Zufallsvariablen $(Y_t)_{t\in\mathbb{T}}$ mit einer unendlichen Indexmenge $\mathbb{T}$. Den Begriff der Zufallsvariable haben wir bereits in Abschnitt 1.1 auf Seite 7 kennen gelernt. Genaueres zu Zufallsvariablen im Zusammenhang mit Zeitreihen siehe z.B. Vogel (2015:S. 19ff).

stochastischer Prozess

Definition 7.1. Es sei $\mathbb{T}$ eine unendliche Teilmenge der Menge der reellen Zahlen, z.B. $\mathbb{N}_0$, $\mathbb{Z}$ oder $\mathbb{R}_0^+ = [0, \infty)$. Dann ist ein *stochastischer Prozess* eine Familie

$$(Y_t)_{t\in\mathbb{T}}$$

von unendlich vielen Zufallsvariablen auf einem gemeinsamen Wahrscheinlichkeitsraum. □

Wir können uns einen stochastischen Prozess so vorstellen, dass aus dem gegebenen Ereignisraum Ω per Zufall ein Ereignis ω ausgewählt wurde, und was wir dann als

Trajektorie

Beobachtungen sehen, ist ein Zeitfenster $\mathbb{T}$ aus der Folge der Realisierungen (y_t) mit $y_t = Y_t(\omega)$. Die Zahlenfolge $(y_t)_{\mathbb{T}}$ nennt man eine *Realisierung* oder auch eine *Trajektorie* des stochastischen Prozesses.

Definition 7.2. Ein stochastischer Prozess $(Y_t)_{t\in\mathbb{Z}}$ heißt *stationär*, wenn die folgenden statistischen Funktionen allesamt unabhängig von der Zeit t sind:

stationärer Prozess

Mittelwert	$\mu(t) = \mathbb{E}(Y_t)$		(7.1)
Varianz	$\sigma^2(t) = \mathrm{Var}\,(Y_t) = \mathbb{E}\big((Y_t - \mu(t))^2\big)$		(7.2)
Autokovarianz	$\gamma(t,s) = \mathrm{Cov}\,(Y_t, Y_s) = \mathbb{E}\big((Y_t - \mu(t)) \cdot (Y_s - \mu(s))\big)$	für $s \in \mathbb{R}$	(7.3)

Für einen stationären Prozess sind also insbesondere $\mu(t) = \mu$ und $\sigma(t) = \sigma$ konstant. □

Beispiel 7.3 (*Weißes Rauschen*)**.** Ein *weißes Rauschen* ist definiert als ein Prozess

$$Y_t = \varepsilon_t \tag{7.4}$$

mit einer Folge (ε_t) unkorrelierter und identisch verteilter Zufallsvariablen, deren Varianz existiert. Ist speziell die Varianz σ_ε^2 für alle ε_t gleich, so schreiben wir $\varepsilon_t \sim \mathrm{WN}(0, \sigma_\varepsilon^2)$. Sind alle Zufallsvariablen zudem normalverteilt, so schreiben wir $\varepsilon_t \sim \mathrm{N}\,(0, \sigma_\varepsilon^2)$. Der Begriff „weißes Rauschen“ kommt aus den Ingenieurwissenschaften und beschreibt Störsignale, die wegen ihrer konstanten Spektraldichte keine nützliche Information enthalten, so wie das weiße Licht. Das weiße Rauschen ist der einfachste stochastische Prozess, spielt jedoch in der Zeitreihenanalyse eine herausragende Rolle. Ein weißes Rauschen mit konstantem Mittelwert und konstanter Varianz, d.h. $\varepsilon_t \sim \mathrm{WN}(\mu_\varepsilon, \sigma_\varepsilon^2)$, ist ein stationärer Prozess. □

weißes Rauschen

Beispiel 7.4 (*Random Walk*)**.** Ein *Random Walk* ist definiert als ein Prozess

$$Y_t = Y_{t-1} + \varepsilon_t \tag{7.5}$$

mit einer Zufallsvariablen $\varepsilon_t \sim \mathrm{WN}(\mu_\varepsilon, \sigma_\varepsilon^2)$. Das ist ein Prozess mit unabhängigen Zuwächsen. Wenn der Erwartungswert μ_ε der Zuwächse verschieden von null ist, hat der Prozess eine Drift. Es ist übrigens gar nicht so einfach, am Zeitreihenplot zu erkennen, ob der Random Walk eine Drift hat. Die Varianz jedes Random Walks beträgt nämlich $\mathrm{Var}\,(Y_t) = t\sigma_\varepsilon^2$ und ist daher linear mit der Zeit[5]. Auch ein Random Walk ohne Drift kann streckenweise also so aussehen, als ob er fort treibt. Vor allem aber ist ein Random Walk *nicht* stationär. □

Random Walk

7.3 Kausale lineare Prozesse

Definition 7.5. Ein stochastischer Prozess $(Y_t)_{t\in\mathbb{Z}}$ heißt *linearer Prozess*, wenn er für alle $t \in \mathbb{Z}$ als

linearer Prozess

$$Y_t = \sum_{j=-\infty}^{\infty} \psi_j\, \varepsilon_{t-j} \tag{7.6}$$

[3] Pole et al. (1994):pp. 4.
[4] Pole et al. (1994):p. 9.
[5] Vogel (2015):S. 27.

dargestellt werden kann, wobei $\varepsilon_t \sim \mathrm{WN}(0, \sigma_\varepsilon^2)$ ein zentriertes weißes Rauschen und (ψ_j) eine Folge reeller Zahlen mit $\sum_{j=-\infty}^{\infty} |\psi_j| < \infty$ ist. Gleichung (7.6) wird auch *Wold-Entwicklung (Wold expansion)* genannt.[6] □

Die zuletzt genannte Bedingung garantiert, dass die Reihe in (7.6) mit Wahrscheinlichkeit 1 und im quadratischen Mittel konvergiert[7]. Ein linearer Prozess ist stets stationär und hat die folgenden Werte für Erwartungswert, Varianz und Kovarianzfunktion:

$$\mu = 0, \qquad \sigma^2(t) = \sigma_\varepsilon^2 \sum_{j=-\infty}^{\infty} \psi_j^2, \qquad \gamma(t) = \sigma_\varepsilon^2 \sum_{j=-\infty}^{\infty} \psi_j \psi_{j+t}. \tag{7.7}$$

(Natürlich ist $\sigma^2(t) = \gamma(0)$.) Die Störterme des weißen Rauschens $(\varepsilon_t) \sim \mathrm{WN}(0, \sigma_\varepsilon^2)$ werden auch als *Schocks* oder *Innovationen* bezeichnet, weil sie als Impuls gebende Ursache den linearen Prozess antreiben.

Schocks, Innovationen

In der Signaltheorie nennt man die Folge (ψ_j) einen „linearen Filter“ (eigentlich: die „Impulsantwort“ eines „linearen Filters“). Ein linearer Prozess entspricht damit der Faltung eines linearen Filters (ψ_j) mit einem weißen Rauschen (ε_j).[8].

Bemerkung 7.6. Interpretiert man in (7.6) den Zeitpunkt t als Gegenwart, so hängt der Zustand Y_t eines linearen Prozesses zum jetzigen Zeitpunkt offenbar auch von ε_{t+1}, ε_{t+2}, ... und damit von der Zukunft ab. Fassen wir den Prozess als reale Zeitreihe auf, so ist diese Eigenschaft natürlich unbrauchbar. Für deterministische Prozesse, die – wie in der klassischen Mechanik – durch exakte Bewegungsgesetze bestimmt sind und die wir exakt messen können, trifft das zwar in gewisser Weise zu: Für solche Prozesse aber braucht ein Laplace'scher Dämon nur *einen einzigen* Zustand zu einem beliebigen Zeitpunkt exakt zu kennen, und er kennt damit *alle* Zustände des Prozesses!

von der *Zukunft* abhängig?

Sobald wir es aber mit Zufallsprozessen oder mit Messungenauigkeiten zu tun haben, können wir die Zustände der Zeitreihe nur in der Vergangenheit und der Gegenwart kennen, Zustände der Zukunft können wir allerhöchstens prognostizieren. Daher beschränken wir uns auf von der Zukunft unabhängige Prozesse, die gegeben sind, wenn die Koeffizienten ψ_j für alle negativen i verschwinden. □

Definition 7.7. Ein stochastischer Prozess $(Y_t)_{t\in\mathbb{Z}}$ heißt *kausaler linearer Prozess*, wenn er für alle $t \in \mathbb{Z}$ als

kausaler linearer Prozess

$$Y_t = \sum_{j=0}^{\infty} \psi_j \, \varepsilon_{t-j} \tag{7.8}$$

dargestellt werden kann, wobei $\varepsilon_t \sim \mathrm{WN}(0, \sigma^2)$ ein zentriertes weißes Rauschen und (ψ_j) eine Folge reeller Zahlen mit $\sum_{j=0}^{\infty} |\psi_j| < \infty$ ist. □

Durch die Bezeichnung „kausal“ wird sicher gestellt, dass Vergangenheit und Gegenwart ursächlich für den weiteren Prozessverlauf sind, nicht aber die Zukunft. Eine besondere Bedeutung haben kausale lineare Prozesse bei Analyse und Prognose stationärer Prozesse. Dies beruht auf dem Zerlegungssatz von Wold, der besagt, dass fast jeder stationäre Prozess sich durch einen kausalen linearen Prozess und einen deterministischen Prozess bilden lässt.[9] Eine wichtige Klasse linearer Prozesse bilden die ARMA (p, q)-

ARMA(p, q)

[6] Palma (2016):S. 93.
[7] Vogel (2015):S. 77.
[8] vgl. Palma (2016):p. 122.
[9] Palma (2016):§2.5.1; Vogel (2015):§5.3.

Modelle mit den Parametern $p, q \in \mathbb{N}_0$. Sie sind durch die Gleichung

$$Y_t = \varphi_1 Y_{t-1} + \ldots + \varphi_p Y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \ldots - \theta_q \varepsilon_{t-q} \tag{7.9}$$

gegeben, wobei die Koeffizienten $\varphi_1, \ldots, \varphi_p$ und $\theta_1, \ldots, \theta_q$ reelle Zahlen sind und (ε_t) ein zentriertes weißes Rauschen mit Varianz σ^2 ist. Damit ein solcher Prozess kausal ist, müssen die Koeffizienten gewisse Bedingungen erfüllen. Betrachten wir zunächst die autoregressiven Prozesse AR(p).

7.4 Die Random-Walk-Hypothese in der Ökonomie

In den Wirtschaftswissenschaften wird für bestimmte ökonomische Prozesse die Random-Walk-Hypothese angenommen, nach der die weitere Entwicklung eines Prozesses allein von seinem aktuellen Zustand abhängt (Y_{t-1}), nicht aber von weiteren historischen Daten oder zeitlich verzögerten Variablen. So wird nach dieser Hypothese beispielsweise die Wachstumsrate des realen privaten Konsums weder von vergangenen Wachstumsraten noch vom vergangenen disponiblen Einkommen bestimmt; auch Aktienkurse lassen sich danach nicht aus der Vergangenheit prognostizieren[10]. Konkret bedeutet diese Annahme, dass die Trajektorien der betreffenden Zeitreihen einen Random Walk darstellen, möglicherweise mit einem nichtverschwindenden Mittelwert μ. Als Beispiel betrachten wir den realen Kursverlauf des Dow-Jones Index im Jahr 2019 im Vergleich zu einem mit Python simulierten Random Walk, siehe Abbildung 7.2. Hier sind die Indexwerte dividiert durch

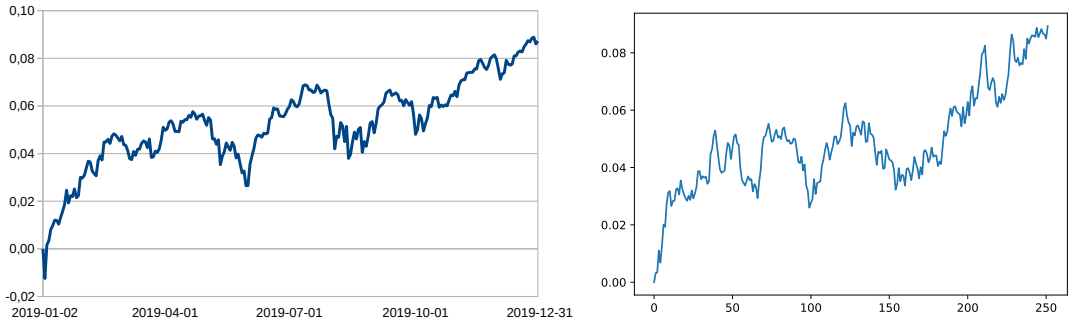


Abbildung 7.2. Zeitreihe des Dow-Jones-Index im gesamten Jahr 2019 (links, Logarithmus der Quotienten der Indexwerte durch den Startwert) und ein simulierter Random Walk (rechts). Datenquelle: finance.yahoo.com

den Startwert am 2. Januar 2019 logarithmisch gegen die Zeit aufgetragen. Im Vergleich dazu ist ein simulierter Random Walk mit verschwindendem Mittelwert und Innovationstermen $\varepsilon_t \sim \mathrm{WN}(0, \sigma_\varepsilon^2)$ mit Mittelwert $\mu_\varepsilon = 0$ und Standardabweichung $\sigma_\varepsilon = 0{,}0035$ wie in Gleichung (7.5) dargestellt.

Die Random-Walk-Hypothese ist eine in der Ökonomie weitgehend anerkannte Grundannahme. Im Grenzwert für unendlich kleine Zeitschritte ist der Random Walk für normalverteilte Innovationsterme ein eindimensionaler Wienerprozess, der wiederum der Theorie der Finanzderivate wie Optionen, Futures und Swaps zugrunde liegt[11]. Die Random-Walk-Hypothese ist im Einklang mit der Hypothese der Markteffizienz der Finanzmärkte, nach

Markteffizienz

[10] Neusser (2011):S. 3.
[11] Hull (2000):§10.

der die jeweils aktuellen Preise stets alle verfügbaren Informationen beinhalten. Folglich ist mit Annahme dieser Hypothes kein Marktteilnehmer in der Lage, auf Finanzmärkten durch technische Analyse, Fundamentalanalyse, Insiderhandel oder anderweitig zu dauerhaft überdurchschnittlichen Gewinnen zu kommen[12]. In[13] werden drei Varianten der Random-Walk-Hypothese diskutiert, die sich in der Verteilung der Innovationsterme ε_t unterscheiden.[14]

Hypothese umstritten

Ist die Random-Walk-Hypothese realistisch? Einige Wirtschaftswissenschaftler bezweifeln dies. Insbesondere Vertreter der *Technischen Analyse* sehen nicht Zufallsterme mit identischer Verteilung als bestimmend für die Kursentwicklung von Wertpapieren an, sondern gewisse geometrische Muster der Kurvenverläufe.[15].

7.5 Übungsaufgaben

Aufgabe 7.1 (Stationäre Prozesse). (a) Gegeben seien die beiden stochastischen Prozesse

$$Y_1(t) = \sin t + \varepsilon_t, \qquad Y_2(t) = \sqrt{|t|} + \varepsilon_t \tag{7.10}$$

mit den Innovationstermen $\varepsilon_t \sim \mathrm{WN}(0,1)$. Programmieren Sie diese zwei Funktionen in Python und plotten Sie die Funktionsgraphen für 100 Zeitpunkte $t = 1, 2, 3, \ldots, 100$.

(b) Gehen wir davon aus, dass Mittelwert und Standardabweichung von Y_1 genau $\mu_1(t) = 0$ und $\sigma_1 = 1/\sqrt{2}$ betragen, und entsprechend $\mu_2(t) = \frac{2}{3}\sqrt{t}$ und $\sigma_2(t) = \mu_2(t)/\sqrt{8}$. Sind Y_1 und Y_2, aufgefasst als stochastische Prozesse, jeweils stationär? Können Sie das anhand der Funktionsgraphen erkennen?

(c*) Leiten Sie eine allgemeine Formel her, die den Mittelwert $\mu(t)$ des stochastischen Prozesses $Y(t) = f(t)+\varepsilon_t$ ergibt, wobei $f : \mathbb{R} \to \mathbb{R}$ und $\varepsilon_t \sim \mathrm{WN}(0, \sigma_\varepsilon^2)$ ist. Was sind dann also speziell die Formeln für $\mu_1(t)$ und $\mu_2(t)$ aus (b)? Was folgt daraus näherungsweise für den Grenzfall $t \to \infty$ großer Zeiträume?

Hinweise: Geometrisch kann man den Mittelwert einer Funktion f in einem Intervall $[0, t]$ als den Quotienten $\mu = A/t$ auffassen, wobei A den Flächeninhalt des Funktionsgraphen zwischen 0 und t mit der t-Achse bezeichnet. Alternativ kann man die Definition $\mu = \frac{1}{t}\sum_{\tau=0}^{t} f(\tau)\Delta\tau$ für den Grenzwert $\Delta\tau \to 0$ verwenden.

[12] Hull (2000):§10.

[13] Campbell et al. (1997):§2.1.

[14] `https://books.google.de/books?id=7Gkri6HWWkgC&pg=PA28`

[15] Campbell et al. (1997):§2.3.2; J. J. Murphy (2016).

8 Autoregressive Modelle

Kapitelübersicht

8.1 Stochastische Prozesse in der Ökonomie 103
8.2 Definition und Eigenschaften autoregressiver Prozesse 104
8.3 Kausalität autoregressiver Prozesse 105
8.4 Übungsaufgaben 108

8.1 Stochastische Prozesse in der Ökonomie

Was haben kausale lineare Prozesse mit der Realität zu tun? Das folgende Beispiel ist ein Klassiker aus den Wirtschaftswissenschaften, der bei genauerem Hinsehen den Einfluss von Investitionen auf eine Volkswirtschaft als genau einen solchen stochastischen Prozess beschreibt. Es handelt sich dabei um das Multiplikator-Akzelerator-Modell, das Paul Samualson kurz nach den wegweisenden Arbeiten von John Maynard Keynes Mitte der 1930er Jahre veröffentlichte. Keynes bewies mit seinem Multiplikatoransatz, dass der Staat durch Erhöhung seiner Ausgaben eine weit höhere Vermehrung des Volkseinkommens bewirkt als er tatsächlich ausgibt. Das stand im scharfen Widerspruch zu der bis dahin üblichen Sparpolitik der Staaten, der „Austerität“, nach der der Staat Schulden strikt vermeiden sollte. Nach Meinung vieler Wirtschaftswissenschaftler und Historiker verschärfte diese restriktive Sparpolitik die Weltwirtschaftskrise von 1929 und führte insbesondere zum Ende der Weimarer Republik[1].

Multiplikatoransatz von Keynes

Austerität

Beispiel 8.1 (*Das Samuelson'sche Multiplikator-Akzelerator-Modell)*. Das volkswirtschaftliche Multiplikator-Akzelerator-Modell von Paul Samuelson[2] aus dem Jahre 1939 setzt eine geschlossene Volkswirtschaft voraus, in der die Nachfrage nach Gütern stets bedient werden kann. Nach diesem Modell setzt sich die Gesamtnachfrage Y_t zum Zeitpunkt t additiv zusammen aus der Nachfrage I_t der Unternehmen nach Investitionsgütern und der

Samuelsons Multiplikator-Akzelerator-Modell

[1] Büttner (2008):S. 423ff.
[2] Samuelson (1939).

Nachfrage C_t der Haushalte nach Konsumgütern,[3]

$$Y_t = I_t + C_t. \tag{8.1}$$

Außerdem sollen die einzelnen Nachfragearten über die Beziehungen

$$C_t = \alpha Y_{t-1} \tag{8.2}$$
$$I_t = \beta\,(C_t - C_{t-1}) + I_{\text{Unplanned}}, \tag{8.3}$$

$0 < \alpha, \beta < 1$, miteinander verbunden sein[4]. Dahinter steckt die Überlegung, dass einerseits die allgemeine Nachfrage Y_t zeitverzögert über die durchschnittliche Konsumneigung α[5] proportional auf den Konsum C_t wirkt und andererseits die notwendigen Investitionen mit dem *Beschleuniger* β und der ungeplanten Investitionsnachfrage $I_{\text{Unplanned}}$ linear von der Veränderung des Konsums abhängen. Die Bezeichnung Multiplikator bezieht sich hier übrigens auf den Grenzwert der Reihe $1 + \alpha + \alpha^2 + \ldots \to (1-\alpha)^{-1}$. Gleichungen (8.2) und (8.3) in (8.1) eingesetzt ergibt die lineare Differenzengleichung 2. Ordnung

$$Y_t = \beta\,(\alpha Y_{t-1} - \alpha Y_{t-2}) + I_{\text{Unplanned}} + \alpha Y_{t-1} = \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + I_{\text{Unplanned}}$$

mit den Koeffizienten $\varphi_1 = \alpha(1+\beta)$ und $\varphi_2 = -\alpha\beta$. Die Lösung der Differenzengleichung kann, in Abhängigkeit von den Parametern α und β, mannigfaltige Formen annehmen, vgl.[6]. Das Modell wird stochastisch, wenn man für jeden Zeitpunkt t die Investitionen $I_{\text{Unplanned}}$ durch Innovationen $\varepsilon_t \sim WN(0, V_t)$ ersetzt, die ein zentriertes weißen Rauschen bilden. Die Nachfrage nach Gütern, aufgefasst als stochastischer Prozess (Y_t), genügt dann der Modellgleichung

$$Y_t = \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \varepsilon_t. \tag{8.4}$$

Siehe dazu auch Abschnitt A.2 im Anhang. Weitere Informationen:[7]. Blanchard (1981) bestimmte für das Bruttoinlandsprodukt (GNP) der USA abzüglich eines linearen Trends im Zeitraum März 1947 bis April 1978 die Werte $\varphi_1 = 1.34$, $\varphi_2 = -0.42$. □

8.2 Definition und Eigenschaften autoregressiver Prozesse

Ein Modell wie das von Samuelson in Beispiel 8.1 nennt man „autoregressiv“, weil der Prozesswert zum Zeitpunkt t, abgesehen von der Zufallsinnovation, eine Linearkombination von Werten desselben Prozesses zu vorhergehenden Zeitpunkten ist. Formal:

AR(p) **Definition 8.2.** Für $p \in \mathbb{N}_0$ heißt ein stochastischer Prozess (Y_t) *autoregressiv von der Ordnung* p, oder kurz AR(p)-Prozess, wenn er der Modellgleichung

$$Y_t = \varphi_1 Y_{t-1} + \ldots + \varphi_p Y_{t-p} + \varepsilon_t \tag{8.5}$$

mit reellen Konstanten $\varphi_1, \ldots, \varphi_p$ und einem zentrierten weißen Rauschen $\varepsilon_t \sim \text{WN}(0, \sigma_\varepsilon^2)$ genügt. Die Koeffizienten φ_i können auch null sein, allerdings nicht $\varphi_p \neq 0$. □

[3] Krugman und Wells (2006):S. 283; Samuelson und Nordhaus (1995):S. 450; Bofinger (2007):S. 352; Felderer und Homburg (1989):S. 112.

[4] Krugman und Wells (2006):S. 272, 278.

[5] Felderer und Homburg (1989):S. 106.

[6] Rinne und Specht (2002):S. 152ff.

[7] weitere Informationen Samuelson und Nordhaus (1995):S. 448, 557; Vogel (2015):S. 79.

Wir können allgemeiner einen autoregressiven Prozess der Ordnung p mit nichtverschwindendem Erwartungswert auch durch

$$Y_t = \varphi_0 + \varphi_1 Y_{t-1} + \ldots + \varphi_p Y_{t-p} + \varepsilon_t \tag{8.6}$$

mit einem weiteren reellen Koeffizienten φ_0 definieren. Durch den Übergang auf den Prozess $Y'_t = Y_t - \mu$ werden wir jedoch sofort wieder auf die Modellgleichung (8.5) zurückgeführt. Aus Gründen, die in der Theorie der Differenzengleichungen behandelt werden, ist der Erwartungswert für (8.6) allerdings nicht φ_0, sondern[8]

$$\mu = \mathbb{E}(Y_t) = \frac{\varphi_0}{1 - \varphi_1 - \ldots - \varphi_p}. \tag{8.7}$$

8.3 Kausalität autoregressiver Prozesse

Auf den ersten Blick ist nun jedoch nicht unbedingt ersichtlich, dass es sich bei (8.5) überhaupt um einen linearen Prozess handelt. Streng genommen ist das auch nicht immer so, allerdings sind die allermeisten autoregressiven Prozesse linear. Der folgende Satz präzisiert diese Aussage.

Satz 8.3. *Ein* AR(p)*-Prozess (8.5) ist genau dann ein kausaler linearer Prozess, wenn die Koeffizienten φ_i so beschaffen sind, dass die Wurzeln des* charakteristischen Polynoms

$$\Phi_p(z) := 1 - \varphi_1 z - \varphi_2 z^2 - \ldots - \varphi_p z^p \tag{8.8}$$

alle außerhalb des Einheitskreises in der komplexen Ebene $\mathbb{C}$ liegen, d.h. einen Betrag > 1 haben. Der Erwartungswert des Prozesses ist dabei null.

Beweis. Die Rolle des charakteristischen Polynoms ergibt sich aus der Theorie der linearen Differenzengleichungen mit konstanten Koeffizienten,[9] wenn wir (8.5) mit der Variablensubstitution $y_t = Y_{t-p}$, d.h. $Y_t = y_{t+p}$, umschreiben zu

Dieser Beweis ist nicht prüfungsrelevant, erklärt aber die Herkunft von Φ_p

$$y_{t+p} - \varphi_1 y_{t+p-1} - \ldots - \varphi_p y_t = \varepsilon_t. \tag{8.9}$$

Dessen Lösungsverhalten hängt von den Wurzeln des charakteristischen Polynoms

$$f_p(x) = x^p - \varphi_1 x^{p-1} - \varphi_2 x^{p-2} - \ldots - \varphi_p \tag{8.10}$$

ab. Insbesondere konvergiert jede Lösung y_t für $\varepsilon_t = 0$ (der „homogenen Differenzengleichung“) genau dannn stabil nach 0, wenn die möglicherweise komplexen Wurzeln x innerhalb des Einheitskreises liegen.[10] Da $\Phi_p(1/x) = f_p(x)$, liegen die Wurzeln von $\Phi_p(z)$ mit $z = 1/x$ dann genau außerhalb des Einheitskreises. In diesem Fall existieren für die Differenzengleichung (8.9) eine stabile konstante Lösung,[11] d.h. der Prozess (Y_t) wird stets ein stabiles Gleichgewicht erreichen, nämlich seinen Erwartungswert null, vgl. (8.7).[12]. Ist aber (8.5) überhaupt ein linearer Prozess? Um das einzusehen, müssen wir (8.5) als eine unendliche Reihe (7.6) umschreiben. Damit wir aber mühsame und unüber-

Backshift-Operator gegen Indexschlachten

[8] siehe Goldberg (1958):S. 170; Vogel (2015):S. 80.
[9] Goldberg (1958):S.163f.
[10] Goldberg (1958):S. 152.
[11] Goldberg (1958):S. 164, 171.
[12] siehe auch Goldberg (1958):S. 170; Vogel (2015):S. 80.

sichtliche „Indexschlachten“ vermeiden, führen wir den Backshift-Operator B ein, der einen stochastischen Prozess zum Zeitpunkt t einfach um eine Zeiteinheit zurücksetzt:

$$BY_t = Y_{t-1} \tag{8.11}$$

Wird er j-mal angewandt, bewirkt er eine Zeitverschiebung um i Zeiteinheiten zurück, $B^j Y_t = BB \ldots BY_t = Y_{t-j}$. So können wir (8.5) nach Umstellung nach ε_t vereinfacht als Operatorgleichung

$$\Phi_p(B)\, Y_t = \varepsilon_t \tag{8.12}$$

schreiben, also ganz ohne Indizes. Definieren wir nun noch das Polynom

$$\Psi(z) = \sum_{j=0}^{\infty} \psi_j z^j, \tag{8.13}$$

so lässt sich auch (7.6) als indexfreie Operatorgleichung

$$Y_t = \Psi(B)\, \varepsilon_t \tag{8.14}$$

schreiben. Ersetzen wir darin den Term ε_t durch den Term auf der linken Seite von Gleichung (8.12), so erhalten wir nach Division durch Y_t die Gleichung

$$1 \;=\; \Psi(B)\, \Phi_p(B) \;=\; (\psi_0 + \psi_1 B + \psi_2 B^2 + \ldots)\, (1 - \varphi_1 B - \ldots - \varphi_p B^p). \tag{8.15}$$

Da nun auf der linken Seite nur eine Eins steht, ergibt ein Koeffizientenvergleich nach Ausmultiplizieren, dass $\psi_0 = 1$ sein muss und die Koeffizienten aller höheren Potenzen von B jeweils verschwinden müssen:

$$\begin{aligned}
&\text{Koeffizient von } B^0\text{:} && \psi_0 = 1\\
&\text{Koeffizient von } B\text{ :} && \psi_1 - \varphi_1 = 0\\
&\text{Koeffizient von } B^2\text{:} && \psi_2 - \psi_1\varphi_1 - \varphi_2 = 0\\
&\text{Koeffizient von } B^3\text{:} && \psi_3 - \psi_2\varphi_1 - \psi_1\varphi_2 - \varphi_3 = 0\\
& && \vdots
\end{aligned}$$

Das ergibt die rekursiven Definitionen der Koeffizienten

$$\psi_0 = 1,\;\; \psi_1 = \varphi_1,\;\; \psi_2 = \varphi_1\psi_1 + \varphi_2\psi_0,\;\; \ldots,\;\; \psi_j = \sum_{k=1}^{j} \varphi_k \psi_{j-k},\;\; \ldots \tag{8.16}$$

mit $\varphi_k = 0$ für $k > p$. Damit können wir den autoregressiven Prozess umschreiben zu

$$Y_t = \sum_{j=0}^{\infty} \psi_j\, \varepsilon_{t-j}. \tag{8.17}$$

Da wir anhand der Voraussetzungen des Satzes schon wissen, dass der Prozess (Y_t) ein stabiles Gleichgewicht erreicht, muss die Bedingung $\sum_j^\infty |\psi_j| < \infty$ notwendig erfüllt sein. □

Wenn eine der Einheitswurzeln des charakteristischen Polynoms (8.8) den Betrag 1 hat, so ist der AR(p)-Prozess (8.5) nicht stationär.[13] In diesem Fall existieren für die zugehörige Differenzengleichung (8.9) konstante oder periodische Lösungen[14], d.h. der Prozess (Y_t) wird kein stabiles Gleichgewicht erreichen.[15]

[13] siehe Vogel (2015):S. 83.
[14] Goldberg (1958):S. 164.
[15] Goldberg (1958):S. 171.

Korollar 8.4. *Für $p \leqq 2$ ist ein* AR(p)*-Prozess (8.5) genau dann ein kausaler linearer Prozess, wenn gilt:*

$$|\varphi_1| < 1 \qquad \textit{für } p = 1, \tag{8.18}$$

$$|\varphi_1| < 1 - \varphi_2 \ \textit{ und } \ |\varphi_2| < 1 \qquad \textit{für } p = 2. \tag{8.19}$$

Beweis. Für $p = 1$: Das charakteristische Polynom (8.8) lautet $\Phi_1(z) = 1 - \varphi_1 z$ und hat somit nur eine einzige Wurzel, $z = 1/\varphi_1$. Da sie nur für den Fall $|\varphi_1| < 1$ außerhalb des Einheitskreises liegt, gilt nach Satz 8.3 die Behauptung.

Auch dieser Beweis ist nicht prüfungsrelevant, kommt aber mit rein elementarmathematischen Umformungen aus

Für $p = 2$: Das charakteristische Polynom (8.8) lautet $\Phi_2(z) = 1 - \varphi_1 z - \varphi_2 z^2$ und hat die beiden (möglicherweise gleichen) Wurzeln

$$a_\pm = -\frac{\varphi_1}{2\varphi_2} \pm \sqrt{\frac{\varphi_1^2}{4\varphi_2^2} + \frac{1}{\varphi_2}} = \frac{-\varphi_1 \pm \sqrt{\varphi_1^2 + 4\varphi_2}}{2\varphi_2} \ \in \ \mathbb{C}. \tag{8.20}$$

Um ihre Lage bezüglich des Einheitskreises zu bestimmen, müssen wir ihre Beträge $|a_\pm|$ errechnen und bestimmen, wann $|a_\pm| > 1$ ist, können äquivalent aber auch berechnenen, wann $|1/a_\pm| < 1$ gilt. Die Kehrwerte sind durch

$$\frac{1}{a_\pm} = \frac{-\varphi_1 \pm \sqrt{\varphi_1^2 + 4\varphi_2}}{2} \tag{8.21}$$

gegeben, wie man mit Hilfe der dritten Binomialformel zur Probe sofort nachrechnen kann:

$$a_\pm \cdot \frac{1}{a_\pm} = \frac{-\varphi_1 \pm \sqrt{\varphi_1^2 + 4\varphi_2}}{2\varphi_2} \cdot \frac{\varphi_1 \mp \sqrt{\varphi_1^2 + 4\varphi_2}}{2} = \frac{\varphi_1^2 + 4\varphi_2 - \varphi_1^2}{4\varphi_2} = 1.$$

Nun sind zwei Fälle zu unterscheiden, je nachdem ob der Ausdruck unter der Wurzel positiv oder negativ ist.

Der Fall $\varphi_1^2 + 4\varphi_2 \geqq 0$: Mit (8.21) ist $|1/a_\pm| < 1$ dann äquivalent zu

$$-1 < \frac{1}{a_-} = \frac{-\varphi_1 - \sqrt{\varphi_1^2 + 4\varphi_2}}{2} \leqq \frac{1}{a_+} = \frac{-\varphi_1 + \sqrt{\varphi_1^2 + 4\varphi_2}}{2} < 1$$

d.h.

$$\varphi_1 - 2 < -\sqrt{\varphi_1^2 + 4\varphi_2} \leqq 0 \leqq \sqrt{\varphi_1^2 + 4\varphi_2} < \varphi_1 + 2.$$

Quadrieren der ersten und der letzten Ungleichung ergibt

$$\varphi_1^2 - 4\varphi_1 + 4 > \varphi_1^2 + 4\varphi_2 \qquad \text{und} \qquad \varphi_1^2 + 4\varphi_2 < \varphi_1^2 + 4\varphi_1 + 4$$

also $1 - \varphi_2 > \varphi_1$ und $-\varphi_1 < 1 - \varphi_2$, d.h. $|\varphi_1| < 1 - \varphi_2$.

Der Fall $\varphi_1^2 + 4\varphi_2 < 0$: In diesem Fall sind die Wurzeln nicht-reell und paarweise komplex-konjugiert, d.h. $a_\pm = -\left(\varphi_1 \pm \mathrm{i}\sqrt{|\varphi_1^2 + 4\varphi_2|}\right)$. Insbesondere gilt

$$\left|\frac{1}{a_+}\right|^2 = \left|\frac{1}{a_-}\right|^2 = \frac{\varphi_1^2 - (\varphi_1^2 + 4\varphi_2)}{4} = -\varphi_2,$$

d.h. $0 \leqq |1/a_\pm| < 1 \Leftrightarrow 0 \geqq \varphi_2 > -1$. Insgesamt liegen die Wurzeln $a_\pm$ also außerhalb des Einheitskreises, wenn die Ungleichungen (8.19) gelten.[16] □

[16] vgl. dazu auch Goldberg (1958):S. 171; Vogel (2015):S. 85.

Im Folgenden werden wir einige Beispiele zur Verdeutlichung der Aussage des Korollars betrachten.

Beispiel 8.5 (AR(1)-Prozesse)**.** [17] Ein AR(1)-Prozess ist mit (8.5) durch

$$Y_t = \varphi_1 Y_{t-1} + \varepsilon_t \tag{8.22}$$

AR(1)

gegeben. Nach Korollar 8.4 handelt es sich nur für den Fall $|\varphi_1| < 1$ um einen kausalen linearen Prozess. In der Tat können wir direkt nachrechnen,

$$\begin{aligned} Y_t &= \varphi_1 Y_{t-1} + \varepsilon_t \\ &= \varphi_1^2 Y_{t-2} + \varphi_1 Y_{t-1} + \varepsilon_t \\ &= \varphi_1^3 Y_{t-3} + \varphi_1^2 Y_{t-2} + \varphi_1 Y_{t-1} + \varepsilon_t = \dots \\ &= \sum_{j=0}^{\infty} \varphi_1^j \varepsilon_{t-j}. \end{aligned}$$

Diese Reihe konvergiert genau für $|\varphi_1| < 1$. Für $|\varphi_1| = 1$ stellt (Y_t) keinen stationären Prozess dar. Als ein Spezialfall ist der Random Walk (7.5) mit $\varphi_1 = 1$ *kein* kausaler Prozess. Für $|\varphi_1| > 1$ schließlich ist der Prozess zwar stationär und linear, aber nicht kausal[18]. □

Random Walk ... again!

AR(2)

Beispiel 8.6 (AR(2)-Prozesse)**.** Ein AR(2)-Prozess ist mit (8.5) durch

$$Y_t = \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \varepsilon_t \tag{8.23}$$

gegeben. Dann handelt es sich genau dann um einen kausalen linearen Prozess, wenn die Bedingung (8.19) erfüllt ist. □

8.4 Übungsaufgaben

Aufgabe 8.1 (**Simulierte AR(1)-Prozesse**)**.** (a) Programmieren Sie in Python eine Funktion `AR_1(phi)`, die einen AR(1)-Prozess mit dem Parameter φ und einer Zufallsvariablen $\varepsilon_t \sim \mathrm{WN}(0, 1)$ als Zeitreihe mit 1000 Datenwerten implementiert. Erstellen Sie damit vier Funktionsgraphen für $\varphi = -1$, $\varphi = 0.5$, $\varphi = 1$ und $\varphi = 2$.

(b) Bei solchen AR(1)-Prozessen gilt für $\varphi \leqq 1$:

$$\mu = 0, \qquad \sigma^2 = \begin{cases} 1, & \text{wenn } |\varphi| < 1, \\ t, & \text{wenn } \varphi = 1. \end{cases} \tag{8.24}$$

Beschreiben Sie mit diesen Angaben die Funktionsgraphen.

[17] Vogel (2015):S. 81.
[18] Vogel (2015):S. 81.

9

Autoregressive Modelle mit gleitendem Durchschnitt

Kapitelübersicht

9.1 MA-Modelle 110
9.2 ARMA 112
9.3 Schätzung der Ordnung von ARMA-Modellen 114

Autoregressive Modelle scheinen reale Zeitreihen gut zu beschreiben. Immerhin sind Klassiker der volkswirtschaftlichen Theorie wie das Samuelson'sche Multiplikator-Akzelerator-Modell als solche Prozesse darstellbar. Allerdings haben autoregressive Modelle auch Nachteile. Einer davon ist, dass sie, wenn sie kausal sind, automatisch auch stationär sind. Das klingt zunächst unspektakuär, ist aber in der Realität eine meist uninteressante

kausal $\Rightarrow$ stationär

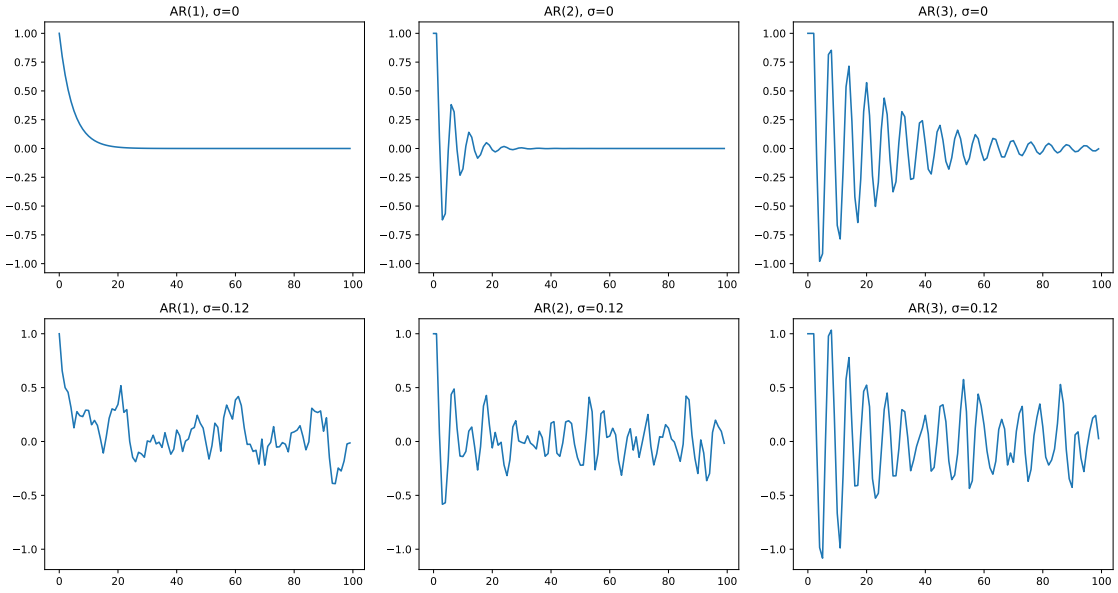


Abbildung 9.1. Simulationen verschiedener AR-Prozesse mit den Koeffizienten $\varphi_1 = 0.8$ für AR(1), $\varphi_1 = 0.8$, $\varphi_2 = -0.7$ für AR(2) und $\varphi_1 = 0.8$, $\varphi_2 = -0.7$, $\varphi_3 = -0.2$ für AR(3) und zeilenweise jeweils mit den Werten $\sigma_\epsilon = 0$ und 0,12.

Eigenschaft. Betrachten wir dazu beispielsweise die Simulationen autoregressiver Prozesse in Abbildung 9.1. In der ersten Zeile der Abbildung sind die Trajektorien der drei simulierten Prozesse AR(1), AR(2) und AR(3) ohne Schocks (d.h. $\sigma_\varepsilon = 0$) skizziert. Sie streben mehr oder weniger zügig auf den stationären Wert 0 zu. Werden dagegen dieselben AR-Prozesse nun mit einem nichtverschwindenden Wert (hier $\sigma_\varepsilon = 0{,}5$) versehen, so erkennen wir, dass es nur die Schocks sind, die verhindern, dass ein kausaler autoregressiver Prozess in seinem stationären Gleichgewicht verharrt.

Stationarität ist meist langweilig

Was sagt uns das? Selbstbezüglichkeit („Autoregression"!) ohne Zufallsmoment führt uns nur zur Null. Ziemlich langweilig eigentlich. Aber mit Schocks genügend hoher Standardabweichung nimmt der deterministische Anteil der Zeitreihe über lange Sicht ab. Warum also nicht den umgekehrten Weg gehen und Modelle verwenden, die nicht auf die bisherigen Werte Y_{t-j} der Zeitreihe zurückgreift, sondern ausschließlich auf die Schocks ε_{t-j}? Genau das geschieht bei dem MA-Modellen, also Modellen mit gleitendem Durchschnitt. Wir werden sie im nächsten Abschnitt behandeln.

Selbstbezüglichkeit ohne Zufall → 0

MA = Moving Average

9.1 MA-Modelle

Grundidee der MA-Modelle ist, dass die Innovationen ε_{t-j}, $j = 1, 2, \ldots$, der Gegenwart und der Vergangenheit als gleitenden Durchschnitt in Abhängigkeit von Gewichtungsfaktoren θ_j den aktuellen Wert Y_t der Zeitreihe bestimmen. Also ganz ähnlich den autoregressiven Modellen, bei denen die vergangenen Beobachtungswerte Y_{t-j} den aktuellen Wert als gewichtete Summe bestimmen. Formal ist die Bestimmung der Parameter eines MA-Modells anhand einer gegebenen Zeitreihe also eine lineare Regression. Da allerdings die Werte der Innovationen nicht beobachtbar sind, ist es keine Regression im üblichen Sinne. Definieren wir dazu MA-Prozesse präziser.

MA und Regression

MA(q)

Definition 9.1. Für $q \in \mathbb{N}_0$ heißt ein stochastischer linearer Prozess (Y_t) *Prozess der gleitenden Mittel (moving averages) von der Ordnung* q, oder kurz MA(q)-Prozess, wenn er der Modellgleichung

$$Y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q} \tag{9.1}$$

mit reellen Konstanten $\theta_1, \ldots, \theta_q$ und einem weißen Rauschen $\varepsilon_t \sim \mathrm{WN}(0, \sigma_\varepsilon^2)$ genügt.[1] Die Koeffizienten θ_i können auch für $i < q$ null sein, nur der höchste Koeffizient θ_q darf nicht verschwinden, $\theta_q \neq 0$. □

MA(∞)

Bemerkung 9.2. Ein kausaler linearer Prozess (7.8) ist also ein MA(∞)-Prozess. Mit anderen Worten ist ein MA-Prozess per Konstruktion immer kausal und damit stationär. Andererseits ist damit jeder autoregressive kausale Prozess ein MA(∞)-Prozess. □

So definiert hat ein allgemeiner MA-Prozess stets den Erwartungswert null. Einen MA-Prozess mit einem Erwartungswert $\mu = \mathbb{E}(Y_t) \neq 0$ könnten wir beschreiben, indem wir in (9.1) die Konstante μ hinzufügen:

$$Y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q} \tag{9.2}$$

Allerdings können wir diese Modellgleichung durch Betrachtung des Prozesses $Y'_t = Y_t - \mu$ nach (9.1) überführen. Wegen der Unkorreliertheit der Variablen ε_t des weißen Rauschens

[1] In der Literatur werden die Koeffizienten θ_j oft mit dem umgekehrten Vorzeichen definiert, z.B. Vogel (2015). Wir halten uns hier an die Konvention von Brockwell und Davis (2016), Deistler und Scherrer (2018), Kreiß und Neuhaus (2006), Neusser (2011), Palma (2016) und Shumway und Stoffer (2017), die auch in dem Python Modul `statsmodels` verwendet wird.

gilt für die Autokorrelationsfunktion $\rho(k)$ eines MA(q)-Prozesses

$$\rho(k) = \begin{cases} \dfrac{\theta_1 + \theta_1\theta_{k+1} + \theta_2\theta_{k+2} + \ldots + \theta_{q-k}\theta_q}{1 + \theta_1^2 + \theta_2^2 + \ldots + \theta_q^2} & \text{für } 1 \leqq k \leqq q, \\ 0 & \text{für } k > q. \end{cases} \tag{9.3}$$

Obwohl jeder MA-Prozess per Konstruktion kausal und stationär ist, ist es üblich, Bedingungen an die Parameter θ_i zu stellen. Ein wesentlicher Grund besteht darin, dass wir ohne solche Einschränkungen nicht eindeutig von der Autokorrelationsfunktion auf die Koeffizienten schließen können. So hat beispielsweise ein MA(1)-Prozess mit dem Parameter $1/\theta_1$ dieselbe ACF und damit auch dieselbe PACF wie der MA(1)-Prozess mit dem Parameter θ_1, was die folgende Umformung von (9.3) zeigt:

$$\rho(1) = \frac{\frac{1}{\theta_1}}{1 + \frac{1}{\theta_1^2}} = \frac{\frac{1}{\theta_1}}{\frac{\theta_1^2+1}{\theta_1^2}} = \frac{\theta_1}{1 + \theta_1^2}$$

Abhilfe schafft hier das Kriterium der Invertierbarkeit eines MA-Prozesses, in dem gefordert wird, das er sich als AR(∞)-Prozess beschreiben lässt.

Definition 9.3. Ein MA(q)-Prozess $(Y_t)_{t\in\mathbb{Z}}$ heißt *invertierbar* bezüglich (ε_t), wenn er sich als AR(∞)-Prozess, also in der Form invertierbar

$$Y_t = \sum_{i=1}^{\infty} \varphi_i Y_{t-i} + \varepsilon_t \tag{9.4}$$

beschreiben lässt, wobei (φ_i) eine Folge von reellen Zahlen mit $\sum\limits_{i=1}^{\infty} |\varphi_i| < \infty$ ist. □

Was bringt die Invertierbarkeit eines MA-Prozesses? Da ein invertierbarer MA-Prozess mit (9.4) also der Gleichung

$$\varepsilon_t = Y_t - \sum_{1}^{\infty} \varphi_i Y_{t-i} \tag{9.5}$$

genügt, lassen sich bei ihm die nichtbeobachtbaren Innovationen ε_t aus den beobachteten Prozesswerten Y_t der Vergangenheit und Gegenwart rekonstruieren. Damit ist für einen invertierbaren Prozess aber auch der äquivalente AR-Prozess rekonstruierbar. Für nichtinvertierbare MA-Prozesse ist eine solche Rekonstruktion nicht möglich. invertierbar ⇒ AR-Prozess aus MA rekonstruierbar

Zudem stellt die Beschränkung auf invertierbare MA-Prozesse praktisch keine Einschränkung dar, denn man kann zu einem MA(q)-Prozess, dessen charakteristisches Polynom keine Wurzel auf dem Einheitskreis hat, auch immer eine MA(q)-Darstellung mit einem charakteristischen Polynom finden, dessen Wurzeln weder auf noch innerhalb des Einheitskreises liegen, wie wir später mit Satz 9.8 sehen werden.

Satz 9.4. *Ein* MA(q)*-Prozess (9.1) ist genau dann invertierbar bezüglich* (ε_t)*, wenn alle Wurzeln des charakteristischen Polynoms*

$$\Theta_q(z) = 1 + \theta_1 z + \theta_2 z^2 + \ldots + \theta_q z^q \tag{9.6}$$

außerhalb des Einheitskreises liegen.

Beweis. [2]. □

Beispiel 9.5. Da ein MA(0)-Prozess der Modellgleichung $Y_t = \varepsilon_t$ genügt, ist er ein weißes Rauschen (Beispiel 7.3). Da das charakteristische Polynom $\Theta_0(z) = 1$ gar keine Nullstellen hat, hat es insbesondere keine Nullstellen für $|z| \leqq 1$. Welcher AR-Prozess entspricht ihm dann? Selbstverständlich der AR(0)-Prozess. □

MA(0) = Weißes Rauschen

MA(1)

Beispiel 9.6. Ein MA(1)-Prozess ist durch die Gleichung $Y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1}$ gegeben. Um für ihn die Koeffizienten φ_i in der Reihe (9.4) zu bestimmen, setzen wir $\varepsilon_t = Y_t - \theta_1 \varepsilon_{t-1}$ sukzessive mit abnehmenden t in sich selbst ein und erhalten so

$$\varepsilon_t = Y_t - \theta_1 \varepsilon_{t-1} = Y_t - \theta_1 \varepsilon_{t-1} - \theta_1^2 \varepsilon_{t-2} = \ldots = Y_t - \sum_{i=1}^{\infty} \theta_1^i Y_{t-i}$$

Der Prozess ist also genau dann invertierbar, wenn $|\theta_1| < 1$. Mit dem charakteristischen Polynom $\Theta_1(z) = 1 - \theta_1 z$ ist das ganz im Einklang mit Satz 9.4.[3]. □

9.2 ARMA

Der Ausgangspunkt dieses Kapitels war es, ausdrucksstärkere als die autoregressiven Modelle für Zeitreihen zu finden. Leider ist uns das trotz des neuen Ansatzes mit gleitenden Durschnitten aber bis jetzt nicht gelungen: Mit Bemerkung 9.2 ist jeder autoregressive kausaler Prozess zwar ein spezieller MA(∞)-Prozess, d.h. MA-Prozesse sind eine echte Erweiterung autoregressiver Prozesse. Allerdings ist mit Definition 9.3 ein invertierbarer MA-Prozes sofort ein AR(∞)-Prozess. Vor allem invertierbare MA-Prozesse sind aber interessant, da sich nur bei ihnen die unbeobachtbaren Innovationen ε_{t-j} aus den Prozessvariablen Y_t der Gegenwart und der Vergangenheit eindeutig rekonstruieren lassen. Ein nächster Schritt, die Ausdrucksstärke der Modelle zu erhöhen, ist es, mit den

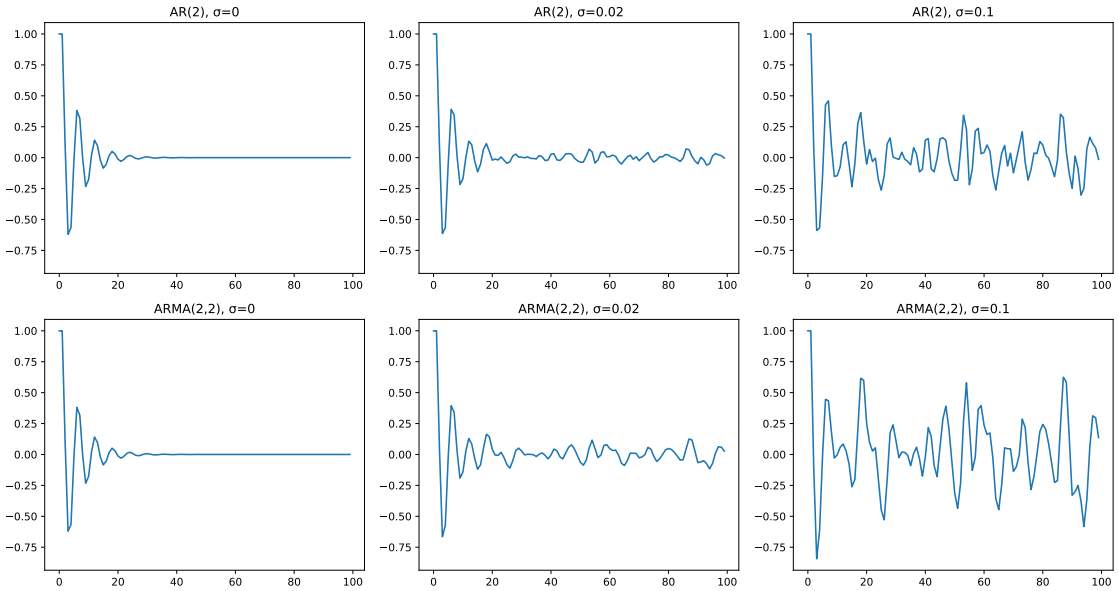


Abbildung 9.2. Vergleich simulierter AR- und ARMA-Prozesse mit den Koeffizienten $\varphi_1 = 0.8$, $\varphi_2 = -0.7$ für AR(2) und $\varphi_1 = 0.8$, $\varphi_2 = -0.7$, $\theta_1 = 0.8$ für ARMA(2,2), zeilenweise mit den Standardabweichungen $\sigma_\varepsilon = 0$, 0,02 und 0,1 dargestellt.

ARMA = AR + MA

ARMA-Modellen beide Modelle zu vereinen. Ein Vergleich von AR(2)-Prozessen mit

[2] Brockwell und Davis (2016):§3.1.

[3] vgl. dazu auch Shumway und Stoffer (2017):S. 83.

ARMA(2,2)-Prozessen entsprechender AR-Koeffizienten bei verschiedenen σ-Werten für die Innovationsterme wie in Abbildung 9.2 zeigt, dass die Grundstruktur der Autoregression durch die normalverteilten Innovatiosnterme der MA-Anteile geglättet, oder bei hohen Werten für θ_j oder σ_ε sogar durch sie dominiert wird.

Insgesamt betrachtet erhöhen also ARMA-Prozesse die Ausdrucksstärke von AR-Prozessen zwar nicht qualitativ, allerdings erweitern sie die Möglichkeiten, differenziert den Einfluss vergangener Innovationen zu modellieren.

Definition 9.7. Ein stochastischer Prozess $(Y_t)_{t\in\mathbb{Z}}$ heißt ARMA(p,q)-Prozess, wenn er stationär ist und der Gleichung

$$Y_t - \varphi_1 Y_{t-1} - \ldots - \varphi_p Y_{t-p} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q} \tag{9.7}$$

genügt, wobei $\varphi_p \neq 0$ und $\theta_q \neq 0$ gilt und die gemäß (8.8) und (9.6) definierten charakteristischen Polynome $\Phi_p(z)$ und $\Theta_q(z)$ keine gemeinsamen Wurzeln haben. □

Die Modellgleichung (9.7) ist natürlich äquivalent zu Gleichung (7.9), die wir in der Einleitung kennen gelernt haben. Ein ARMA$(p,0)$-Prozess ist ein reiner AR(p)-Prozess, während ein ARMA$(0,q)$-Prozess ein reiner MA(q)-Prozess ist. Die Bedingungen für die Stationarität (Definition 7.2), Kausalität (Definition 8.2) und Invertierbarkeit (Definition 9.3) gelten unverändert auch für das ARMA-Modell. Es gilt also insbesondere:

ARMA$(p,0)$ = AR(p)

ARMA$(0,q)$ = MA(q)

- Eine stationäre Lösung der Gleichung (9.7) gibt es genau dann, wenn das AR-Polynom Φ_p keine Wurzel auf dem Einheitskreis hat; die stationäre Lösung ist eindeutig.
- Ein ARMA-Prozess ist genau dann kausal bezüglich ε_t, wenn alle Wurzeln des AR-Polynoms Φ_p außerhalb des Einheitskreises liegen.
- Ein ARMA-Prozess ist genau dann invertierbar bezüglich ε_t, wenn alle Wurzeln des MA-Polynoms Θ_q außerhalb des Einheitskreises liegen.

Hätten die charakteristischen Polynome Φ_p und Θ_q gemeinsame Nullstellen a_i, so könnte es mehrere stationäre Lösungen der Gleichung (9.7) geben, d.h. eine Lösung der Gleichung wäre nicht eindeutig. Die Eindeutigkeit lässt sich jedoch einfach herstellen, indem man beide Polynome durch die gemeinsamen Faktoren $(z - a_i)$ dividiert. Dabei reduzieren sich die Ordnungen p und q je Faktor um eins[4]. Der folgende bemerkenswerte Satz geht über diese Eigenschaft weit hinaus, denn er besagt, dass man ARMA-Prozesse nicht nur geeignet zu kausalen und invertierbaren Prozessen *umformen* kann, sondern dass sie es in fast allen Fällen sogar kausal und invertierbar *sind*.

Satz 9.8. *Sei (Y_t) ein durch Gleichung (9.7) gegebener ARMA(p,q)-Prozess, dessen charakteristische Polynome keine Wurzeln auf dem Einheitskreis haben, d. h. dass $\Phi_p(z) \neq 0$ und $\Theta_q(z) \neq 0$ für alle $|z| = 1$ gilt. Dann gibt es stets zwei Polynome $\tilde{\Phi}_p(z)$ und $\tilde{\Theta}_q(z)$ mit Grad p bzw. q und ein weißes Rauschen $\tilde{\varepsilon}_t$, so dass*

$$Y_t - \tilde{\varphi}_1 Y_{t-1} - \ldots - \tilde{\varphi}_p Y_{t-p} = \tilde{\varepsilon}_t + \theta_1 \tilde{\varepsilon}_{t-1} + \ldots + \theta_q \tilde{\varepsilon}_{t-q} \tag{9.8}$$

gilt, so dass (Y_t) kausal und invertierbar ist.

[4] Vogel (2015):S.102.

Beweis. [5] Seien $a_r, \ldots, a_p$ bzw. $b_s, \ldots, b_q$ die Wurzeln von Φ_p und Θ_q mit $|z| < 1$ innerhalb des Einheitskreises. Die damit durch die Austauschung der Nullstellen durch ihre Kehrwerte, also durch

$$\tilde{\Phi}_p(z) = \Phi_p(z) \cdot \prod_{j=r}^{p} \frac{(1 - a_j z)}{(1 - a_j^{-1} z)}, \quad \tilde{\Theta}_q(z) = \Theta_q(z) \cdot \prod_{j=s}^{q} \frac{(1 - b_j z)}{(1 - b_j^{-1} z)} \tag{9.9}$$

definierten Polynome haben keine Nullstellen $|z| \leqq 1$. Mit Hilfe des bereits im Beweis von Satz 8.3 in Gleichung (8.11) definierten Backshift-Operators können wir dann die Innovationsterme

$$\tilde{\varepsilon}_t = \frac{\tilde{\Phi}_p(B)}{\tilde{\Theta}_q(B)} \, \varepsilon_t \tag{9.10}$$

definieren, die mit[6] wieder ein weißes Rauschen sind. Daher ist mit Gleichung (9.8) der Prozess (Y_t) kausal und invertierbar. □

Bemerkung 9.9. Satz 9.8 ist verblüffend. Er geht auf Arbeiten der US-amerikanischen Statistiker Brockwell und Davis aus den 1980er Jahren zurück und besagt nichts weniger, als dass *alle* ARMA-Prozesse kausal und linear sind, solange ihre charakteristischen Polynome keine Einheitswurzeln haben. Was für die Aussage zur Invertierbarkeit noch eher wie eine rein technische Umformulierung ausgesehen haben mag – Zufallsterme ε_t geeignet nach $\tilde{\varepsilon}_t$ umzuformen ändert ja nichts an kausalen Zusammenhängen –, bedeutet für die Philosophie der Zeit auf den ersten Blick eine geradezu dramatische Konsequenz: Ein nichtkausaler Prozess (mit charakteristischen Polynomen ohne Einheitswurzeln), der mit Bemerkung 7.6 von Zufallstermen der Zukunft abhängt, ist mit Satz 9.8 nahtlos als ein Prozess darstellbar, der genau diese Abhängigkeit in die Innovationen der Zukunft nicht hat. Auf den zweiten Blick allerdings reduziert sich die Bedeutung für die Kausalität dann doch auf einen eher technischen Aspekt, denn nichtkausale Prozesse hängen zwar von zukünftigen *Zufallsereignissen* ε_t ab, nicht aber von zukünftigen *Beobachtungen* Y_t. Strenggenommen ist daher der Begriff „kausal“ in Definition 8.2 nicht angemessen, denn ein nichtkausaler Prozess kann im üblichen Wortsinn von „Kausalität“ durchaus kausal sein! Die Fachbegriffe sind aber nun mal so, wie sie sind. Da sie in der Fachliteratur jedoch einerseits sehr gebräuchlich und andererseits sehr brauchbar sind, da mit ihnen kausale und invertierbare Prozesse mathematisch einfacher zu behandeln sind, werden sie auch in diesem Skript verwendet. □

Philosophie der Zeit?

Missverständnis von „kausal“?

9.3 Schätzung der Ordnung von ARMA-Modellen

Wir haben bisher die Theorie von ARMA-Prozesse erarbeitet, haben aber bislang nicht das praktische Problem behandelt, eine real beobachtete Zeitreihe als einen ARMA-Prozess einzuordnen. Erste Aufgabe muss es daher sein, geeignete Methoden dafür zu finden. In der Praxis als nützlich dazu haben sich die Autokorrealationsfunktion ACF und die partielle Autokorrelationsfunktion PACF erwiesen, die wir in diesem Abschnitt behandeln.

Schätzung eines ARMA-Modells für eine gegebene Zeitreihe

Definition 9.10. Die für eine Zeitreihe (y_t) *geschätzte Autokorrelationsfunktion* (ACF)

ACF = geschätzte Autokorrelationsfunktion

[5] Brockwell und Davis (1991):Proposition 3.5.1.

[6] Brockwell und Davis (1991):S.105.

$\hat{\rho} : \mathbb{N}_0 \to \mathbb{R}$, auch *empirische Autokorrelationsfunktion* genannt, ist definiert durch

$$\hat{\rho}(k) \;=\; \frac{\sum_{t=1}^{n-k} (y_t - \bar{y})\,(y_{t+k} - \bar{y})}{\sum_{t=1}^{n} (y_t - \bar{y})^2}, \tag{9.11}$$

mit dem arithmetischen Mittel $\bar{y}$ der Messwerte $\{y_t\}$, vgl.[7]. □

Die Autokorrelationsfunktion $\hat{\rho}(k)$ beschreibt die Korrelation zwischen einer Beobachtung y_t und allen anderen Beobachtungswerten $y_{t+1}, \ldots, y_{t+k}$ danach, Idee der ACF

$$\ldots, y_{t-1}, y_t, y_{t+1}, \ldots, y_{t+k-1}, y_{t+k}, y_{t+k+1}, \ldots$$

Mit Hilfe der Autokorrelationsfunktion lässt sich nun die partielle Autokorrelationsfunktion wie folgt definieren.

Definition 9.11. Die *geschätzte partielle Autokorrelationsfunktion* (PACF) $\hat{\pi} : \mathbb{N}_0 \to \mathbb{R}$, oft auch *empirische partielle Autokorrelationsfunktion* genannt, ist definiert durch PACF = geschätzte partielle Autokorrelationsfunktion

$$\hat{\pi}(k) = C_{k,k}$$

mit den Koeffizienten $C_{k,t}$, die wiederum durch die *Durbin-Levinson-Rekursion* *Hilfe, eine komplizierte rekursive Definition!? ... Aber leicht zu programmieren!*

$$C_{k,k} = \frac{\hat{\rho}(k) - \sum_{t=1}^{k-1} C_{k-1,t}\,\hat{\rho}(k-t)}{1 - \sum_{t=1}^{k-1} C_{k-1,t}\,\hat{\rho}(t)}, \qquad C_{k,i} = C_{k-1,i} - C_{k,k}\,C_{k-1,k-i} \qquad (0 < i < k)$$

gegeben sind; vgl.[8], oder auch[9],[10]. Die Startwerte $C_{0,0} = 1$ und $C_{1,1} = \hat{\rho}(1)$ ergeben sich aus der ersten der beiden Gleichungen. □

Die PACF $\hat{\pi}(k)$ gibt eine Maßzahl für den Zusammenhang von y_t und y_{t+k} an, der die Einflüsse der Werte dazwischen abzieht, Idee der PACF

$$\ldots, y_{t-1}, y_t, y_{t+1}, \ldots, y_{t+k-1}, y_{t+k}, y_{t+k+1}, \ldots$$

Bei einer gegebenen Zeitreihe wird man im Allgemeinen die Ordnungen p und q nicht kennen. Die Autokorrelationsfunktionen ACF und PACF sind wichtige Instrumente, um die Natur des zugrundeliegenden Prozesses zu bestimmen. Sie können verwendet werden, um für eine gegebene Zeitreihe die Ordnung p und q zu schätzen. Grundlage für die Schätzungen bilden die in Tabelle 9.1 zusammengefassten theoretischen Eigenschaften von ACF und PACF. Bei einem AR(p)-Prozess strebt die PACF für Lags $k > p$ exponentiell gegen null, bei einem MA(q)-Prozess ist es die ACF für Lags $k > q$, vgl.[11],[12],[13]. Für einen ACF und PACF helfen, ARMA(p, q) zu bestimmen

[7] Vogel (2015):S. 32.
[8] Vogel (2015):S. 35.
[9] Kreiß und Neuhaus (2006):S. 40f, 69ff.
[10] Neusser (2011):S. 63.
[11] Vogel (2015):S. 88f.
[12] Kreiß und Neuhaus (2006):S. 143.
[13] Neusser (2011):S. 64f.

Tabelle 9.1. Theoretische Eigenschaften der Autokorrelationsfunktionen[a,b]

Prozess	ACF	PACF
AR(p)	fällt für $k > p$ exponentiell gegen null (monoton oder oszillierend)	$\pi(k) = 0$ für $k > p$
MA(q)	$\rho(k) = 0$ für $k > q$	fällt für $k > q$ exponentiell gegen null (monoton oder oszillierend)
ARMA(p, q)	fällt für wachsendes $k > \max\{p, q+1\}$ exponentiell gegen null (monoton oder oszillierend)	fällt für große $k > \max\{p, q+1\}$ exponentiell gegen null (monoton oder oszillierend)

[a] Neusser (2011):S. 64f.
[b] Vogel (2015):S. 110.

ARMA(p, q)-Prozess mit $p, q > 0$ schließlich bricht keine der Autokorrelationsfunktionen ab, sondern beide fallen exponentiell gegen null, eventuell oszillierend[14]. Falls (Y_t) ein kausaler und invertierbarer ARMA(p, q)-Prozess ist, gilt Folgendes. Die ACF genügt für $k > \max\{p, q+1\}$ der Differenzengleichung

$$\rho(k) = \varphi_1 \rho(k-1) + \ldots + \varphi_p \rho(k-p).$$

Die Nullstellen der charakteristischen Gleichung liegen wegen der Kausalität innerhalb des Einheitskreises (vgl. die Diskussion nach Gleichung (8.10)). D.h. die Autokorrelationsfunktion $\rho(k)$ fällt für k gegen unendlich exponetiell gegen null. Ob $\rho(k)$ monoton oder oszillierend gegen null fällt, hängt von den Nullstellen der charakteristischen Gleichung ab. Die PACF $\pi(k)$ beginnt ab $k > p$ gegen null zu fallen.

Um eine Vermutung über die Größe von p und q zu bekommen, sollte also zuerst eine visuelle Auswertung mit Korrelogrammen der Zeitreihe, wie in Abbildung 9.3, der empirischen ACF und PACF vorgenommen werden. Ein Korrelogramm ist der Funktionsgraph der Autokorrelationsfunktionen $\hat{\rho}(k)$ bzw. $\hat{\pi}(k)$, oft als Balken- oder Stabdiagramm gegen den Lag k aufgetragen. Häufig wird dabei ein Konfidenzintervall für das Konfidenzniveau

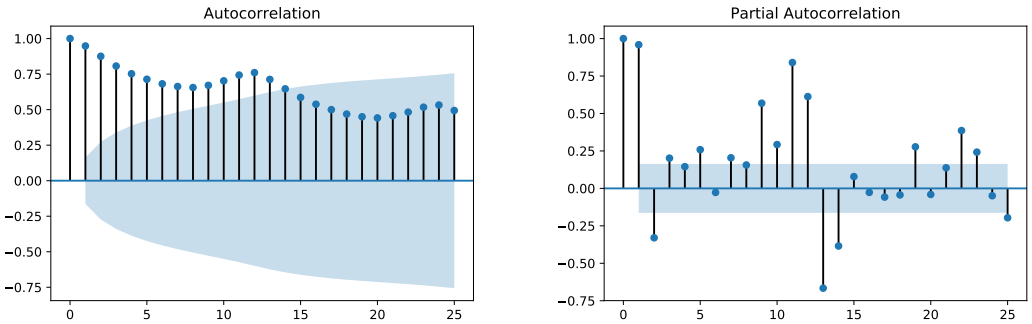


Abbildung 9.3. Korrelogramme der geschätzten Autokorrelationsfunktionen ACF und der PACF für eine reale Zeitreihe. Die saisonale Periode 12 ist hier gut zu erkennen

95% eingezeichnet. Für die Autokorrelationsfunktion $\hat{\rho}(k)$ bezeichnet dieses Konfidenzniveau die Wahrscheinlichkeit, mit der jeweils die Hypothese für einen MA(k)-Prozess mit normalverteiltem Rauschen $\varepsilon_t \sim N(0, \sigma_\varepsilon^2)$ verworfen werden muss. Die Intervallgrenzen

Konfidenzintervall zur Filterung der MA- und AR-Koeffizienten

[14] Vogel (2015):S. 110.

sind dabei durch die Bartlett-Formel

$$\rho_{\pm}(k) = \pm 1{,}96 \cdot \sqrt{\frac{1 + 2\left[\rho(1) + \rho(2) + \ldots + \rho(k)\right]}{n}} \tag{9.12}$$

gegeben[15]. Im Korrelogramm für die partielle Autokorrelationsfunktion werden dagegen entsprechend die Intervallgrenzen

$$\pm \frac{1{,}96}{\sqrt{n}} \tag{9.13}$$

eingezeichnet[16]. Mit Korrelogrammen können wir so zumindest reine AR- und MA-Prozesse gut identifizieren, wenn nämlich die ACF bzw. PACF nach wenigen Lags k plötzlich abbricht. Für ARMA(p, q)-Modelle mit $p \cdot q > 0$ sind allerdings die Ordnungen an den Korrelogrammen nicht so leicht auszumachen, denn sowohl ACF als auch PACF klingen hier exponentiell ab. Eine Auflistung verschiedener simulierter allgemeiner ARMA-Prozesse ist in Abbildung 9.4 dargestellt. Hier ist die Tendenz zu erhöhten Autokorrelationen bei steigender Ordnung der Autoregression gut zu erkennen, angefangen von dem völlig unkorreliertem weißen Rauschen über den Random Walk hin zu den ARMA-Prozessen höherer Ordnung.

[15] Vogel (2015):S. 96ff.
[16] Neusser (2011):S. 66.

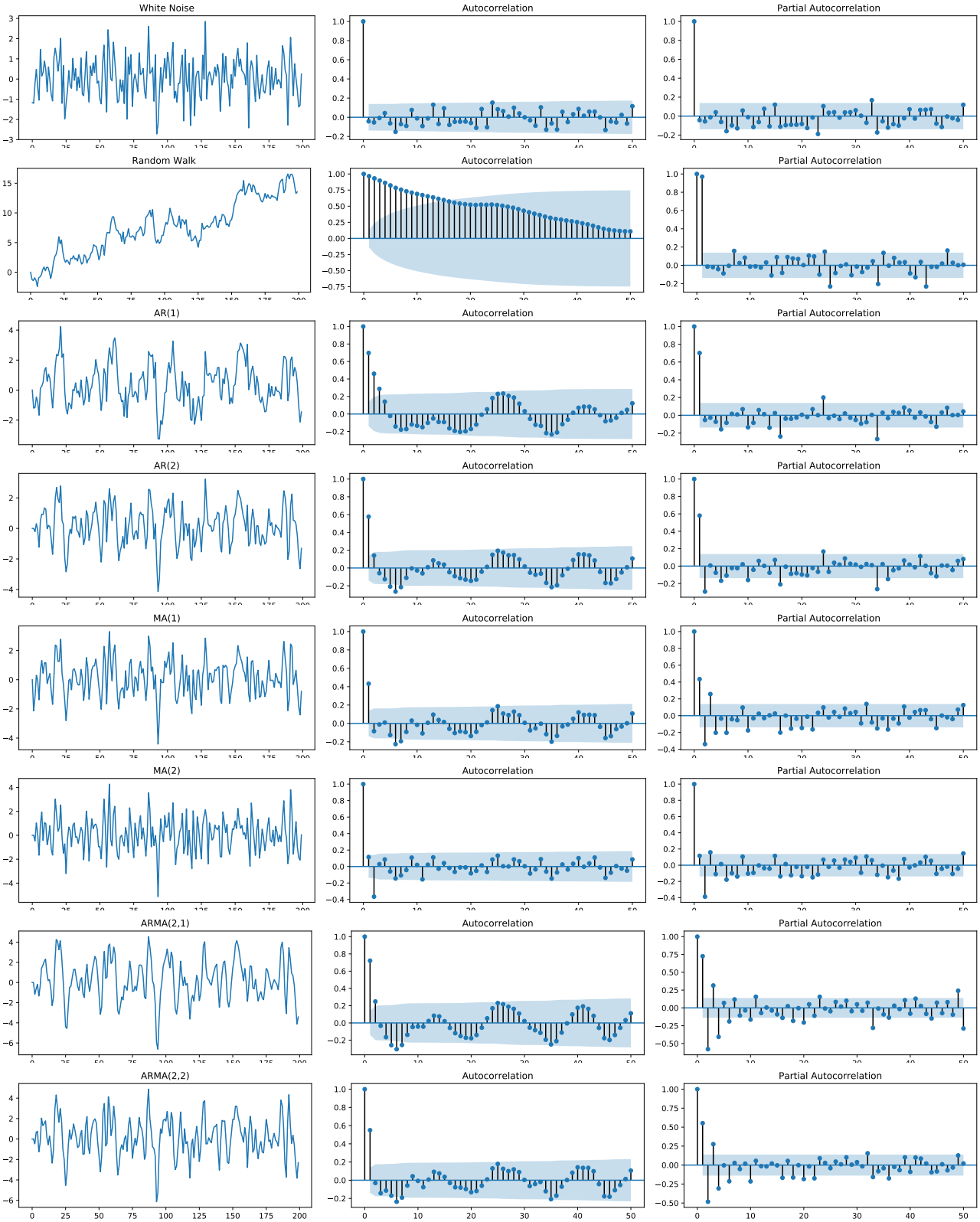


Abbildung 9.4. Simulation verschiedener ARMA-Prozesse und ihrer Autokorrelationsfunktionen.

10 Trends und Perioden: SARIMA-Modelle

Oft sind reale Zeitreihen, insbesondere aus dem Bereich der Wirtschaftswissenschaften, nicht stationär. Daher sind die bisher betrachteten ARMA-Modelle grundsätzlich zu ihrer Analyse zunächst leider nicht geeignet. Dennoch sind sie nicht völlig unbrauchbar. Denn erstens versteht man sie und ihre Eigenschaften recht gut, insbesondere kann man ihre Ordnung durch die in Abschnitt 9.3 beschriebenen Methoden gut schätzen. Zweitens können sie als elementare stochastische Prozesse allgemeinerer Zeitreihen verstanden werden, die man mit diesem Verständnis analysieren kann. Eine dafür geeignete Klasse nichtstationärer Prozesse sind die „integrierten“ Prozesse, die wir in diesem Kapitel behandeln werden.

nichtstationäre Modelle

10.1 Zeitreihen mit Trends: Integrierte Prozesse

Ein ARMA-Prozess kann durch Differenzieren zu einem ARIMA-Prozess verallgemeinert werden. Damit können nun auch Prozesse mit einem Trend dargestellt werden, also nichtstationäre Zeitreihen. Um die Idee dahinter zu illustrieren, definieren wir zunächst das Differenzieren des Prozesses (Y_t), eigentlich der Differenzbildung, durch

$$\Delta Y_t := Y_t - Y_{t-1}. \tag{10.1}$$

Umgekehrt kann man mit $\Delta Y_t + Y_{t-1} = Y_t$ einen differenzierten Prozess wieder „integrieren“. Betrachten wir als Beispiel den Prozess $Y_t = a + bt + \varepsilon_t$ mit einem linearen Trend $b \neq 0$. Dann gilt

$$\Delta Y_t = Y_t - Y_{t-1} = a + bt + \varepsilon_t - (a + b(t-1) + \varepsilon_{t-1}) = b + \varepsilon_t - \varepsilon_{t-1}.$$

Dann ist der differenzierte Prozess (ΔY_t) im Allgemeinen zwar nicht zentriert, aber der Trend ist verschwunden. Differenziert man einen Prozess (Y_t) zweimal,

$$\begin{aligned}\Delta^2 Y_t &= \Delta(\Delta Y_t) = \Delta(Y_t - Y_{t-1}) = \Delta Y_t - \Delta Y_{t-1} = Y_t - Y_{t-1} - (Y_{t-1} - Y_{t-2}) \\ &= Y_t - 2Y_{t-1} + Y_{t-2},\end{aligned} \tag{10.2}$$

so wird ein quadratischer Trend aus der Zeitreihe gefiltert. Entsprechend kann man durch weitere Differenzierungen höhere nichtlineare Trends aus dem Prozess entfernen.

Definition 10.1. Sei $d \in \mathbb{N}_0$ eine nichtnegative ganze Zahl. Ein stochastischer Prozess (Y_t) heißt ARIMA(p,d,q)-Prozess, wenn $Y'_t = (\Delta^d Y_t)$ ein kausaler ARMA(p,q)-Prozess ist. Mit anderen Worten erfüllt ein ARIMA(p,d,q)-Prozess die Modellgleichung

ARIMA(p,d,q)

$$Y'_t - \sum_{k=1}^{p} \varphi_k Y'_{t-k} = \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \qquad \text{mit} \qquad Y'_t = \Delta^d Y_t, \tag{10.3}$$

wobei $\varepsilon_t \sim \mathrm{WN}(0, \sigma_\varepsilon^2)$ sowie φ_p und $\theta_q \neq 0$ gilt und die gemäß (8.8) und (9.6) definierten charakteristischen Polynome Φ_p und Θ_q des AR- und des MA-Teils keine gemeinsamen Wurzeln haben. □

Das „I“ in dem Kunstwort ARIMA taucht auf, da ein d-fach integrierter ARMA(p,q)-Prozess genau ein ARIMA(p,d,q)-Prozess ist. Ein ARIMA-Prozess (Y_t) genügt ebenfalls der Modellgleichung (9.7) eines ARMA-Prozesses, und zwar mit $\Phi_p(z)(1-z)^d$ als charakteristischem Polynom und entsprechend modifizierten Koeffizienten φ_k.

Beispiel 10.2. Das einfachste Beispiel eines ARIMA-Prozess, der kein ARMA-Prozess ist, ist der Random Walk: Er entsteht durch einmaliges Differenzieren aus dem ARMA(0,0)-

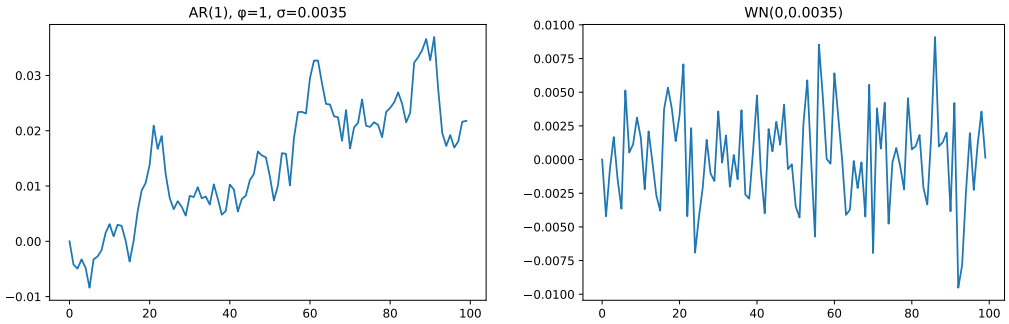


Abbildung 10.1. Ein Random Walk und sein differenzierter Prozess

Prozess $Y_t = \varepsilon_t$, also nach Beispiel 7.3 aus dem Weißen Rauschen, denn

$$\Delta Y_t = Y_t - Y_{t-1} = \varepsilon_t \qquad \Longleftrightarrow \qquad Y_t = Y_{t-1} + \varepsilon_t. \tag{10.4}$$

vgl. Abbildung 10.1. Der Random Walk ist also ein ARIMA(0,1,0)-Prozess, sein integrierter Prozess ist das Weiße Rauschen. □

10.2 Vorgehensweise bei Trends

An eine gegebene Zeitreihe, die wegen eines offensichtlich enthaltenen Trends nicht als stationär angesehen werden darf, soll ein ARIMA(p,d,q)-Modell anpasst werden. Dann haben wir, bevor wir uns um die Ordnungen p und q kümmern müssen, zunächst einmal das Problem der Wahl der Integrationsordnung d zu lösen. Dazu bietet sich an, die Zeitreihe so lange zu differenzieren, bis kein Trend mehr zu erkennen ist. Entsteht nach einmaliger Differenzenbildung bereits ein stationärer Prozess, was bei ökonomischen Zeitreihen häufig der Fall ist, so heißt der ursprüngliche Prozess *integriert* oder auch *differenzenstationär*.

integrierte Prozesse

Beim Differenzieren ist allerdings Vorsicht geboten, denn sowohl ein zu großes als auch ein zu kleines d macht die weitere Analyse mit einem ARMA-Modell unmöglich. Wählt man die Ordnung d zu klein, führt das d-fache Differenzieren zu einem Prozess, der

(noch) nicht stationär ist. Ein instationärer stochastischer Prozess, der zwar der ARMA-Modellgleichung (9.7) genügt, dessen AR-Polynom aber eine Einheitswurzel hat, ist im Zeitreihenplot mitunter schwer oder gar nicht als instationär zu erkennen. Um die Wahl von d zu objektivieren, sind sogenannte Unit-Root-Tests entwickelt worden, mit denen man der Frage nachgehen kann, ob das AR-Polynom Φ_p Einheitswurzeln besitzt. Solche Tests werden z. B. in Computerprogrammen zur automatischen Bestimmung der Ordnung d bei der Suche nach einem passenden ARIMA-Modell gebraucht. Die beiden bekanntesten Vertreter sind der ADF-Test und der KPSS-Test.

Tests auf Einheitswurzeln bei AR-Prozessen

Differenziert man einen ARMA-Prozess, der stationär ist, so entsteht wieder ein stationärer Prozess. Dieser „überdifferenzierte“ Prozess hat dann aber eine Einheitswurzel im MA-Polynom und ist somit nicht invertierbar. Das Testen auf Wurzel 1 im charakteristischen Polynom eines MA-Modells ist viel schwieriger als beim AR-Modell. Für das einfache MA(1)-Modell

Überdifferenzierung

Tests auf Einheitswurzeln bei MA-Prozessen

$$Y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} \qquad \text{mit} \qquad \varepsilon_t \sim \text{IID}(0, \sigma_\varepsilon^2)$$

jedoch gibt es einen Test von Davis, Chen und Dunsmuir: Unter der Nullhypothese $H_0 : \theta_1 = 1$ konvergieren die Zufallsvariablen $n(\hat{\theta}_1 - 1)$ der Maximum-Likelihood-Schätzer $\hat{\theta}_1$ für θ_1, und für die Grenzverteilung werden α-Quantile angegeben. Die Nullhypothese, dass es eine Einheitswurzel gibt, ist zugunsten der Alternative $H_1 : \hat{\theta}_1 < 1$ zum Signifikanzniveau α zu verwerfen, wenn $\hat{\theta}_1 < 1 - c_\alpha/n$ gilt. Die gebräuchlichsten Quantile sind $c_{0,01} = 11{,}93$, $c_{0,05} = 6{,}80$ und $c_{0,1} = 4{,}90$.[1].

10.3 SARIMA

Neben Trends gibt es noch einen weiteren wichtigen deterministischen Anteil bei realen Prozessen, nämlich periodische wiederkehrende Effekte. Solche Effekte werden im wirtschaftlichen Bereich auch saisonal genannt, da die Umsatzzahlen durch regelmäßige Ereignisse geprägt werden, so wie zum Beispiel Jahreszeiten oder bestimmte Feiertage wie Weihnachten oder Ostern. Um saisonale Effekte bei der Zeitreihenanalyse zu berücksichtigen, wird eine ARIMA(p, d, q)-Modell um die vier Parameter $(P, D, Q)_s$ ergänzt werden, die entsprechend für die Periode s einer Saison mit P saisonale AR-Effekte, mit Q saisonale MA-Effekte und mit D saisonale Inegrationseffekte berücksichtigt. In der Fachliteratur wird das gesamte Modell mit SARIMA (für „saisonales ARIMA-Modell“) mit sieben Parametern $(p, d, q) \times (P, D, Q)_s$ bezeichnet.

saisonale Effekte

$(p, d, q) \times (P, D, Q)_s$

Definition 10.3. Ein stochastischer Prozess (Y_t) heißt SARIMA$(p, d, q) \times (P, D, Q)_s$ mit den Parametern $p, d, q, P, D, Q, s \in \mathbb{N}_0$, wenn er der folgenden Modellgleichung genügt:

$$y'_t = \underbrace{\sum_{i=1}^{p} \varphi_i y'_{t-i}}_{\text{AR}} + \varepsilon_t + \underbrace{\sum_{i=1}^{q} \theta_i \varepsilon_{t-i}}_{\text{MA}} + \underbrace{\sum_{i=1}^{P} \tilde{\varphi}_i y'_{t-s-i}}_{\text{saisonale AR}} + \underbrace{\sum_{i=1}^{Q} \tilde{\theta}_i \varepsilon_{t-s-i}}_{\text{saisonale MA}} \tag{10.5}$$

mit dem differenzierten Prozess

$$\underbrace{y'_t = \Delta^d \Delta_s^D y_t}_{\text{(seasonal) I}}, \tag{10.6}$$

und den Bezeichnungen

[1] vgl. Vogel (2015):S. 124f.

y_t – Beobachtungsdaten zum Zeitpunkt t,
s – Periode einer Saison (z.B. 12 bei Monatsdaten, 4 bei Quartalsdaten)
p, P – Ordnung der Autoregression = *Lag* (Verzögerung)
= Anzahl der autoregressiven Terme,
d, D – Ordnung der Differenzenbildung Δ^d ($\Delta y_t = y_t - y_{t-1}$, $\Delta^2 y_t = \Delta\Delta y_t$, ...)
= Anzahl der Differenzen, um Stationarität zu erhalten
q, Q – Ordnung des gleitenden Durchschnitts der Rauschterme

(Großbuchstaben bezeichnen hier jeweils den saisonalen Anteil). Hierbei haben die charakteristischen Polynome

$$\Phi_p(z) = 1 - \sum_{i=1}^{p} \varphi_i z^i, \quad \Theta_p(z) = 1 + \sum_{i=1}^{q} \theta_i z^i, \quad \tilde{\Phi}_p(z) = 1 - \sum_{i=1}^{P} \tilde{\varphi}_i z^{i+s}, \quad \tilde{\Theta}_p(z) = 1 + \sum_{i=1}^{Q} \tilde{\theta}_i z^{i+s} \tag{10.7}$$

jeweils keine gemeinsamen Wurzeln. □

In der Praxis werden die Polynomgrade p, q, P,und Q nicht größer als 2 sein. Zur Vorhersage ökonomischer Zeitreihen kommen am häufigsten SARIMA$(p, d, q) \times (0, 1, 1)_s$-Modelle mit $p + d + q \leqq 4$ zur Anwendung. Das SARIMA$(0, 1, s + 1) \times (0, 1, 0)_s$-Modell ist äquivalent zu dem additiven Holt-Winters-Verfahren der saisonalen exponentiellen Glättung, das beispielsweise in der Logistik zur Bestandsoptimierung verwendet wird[2].

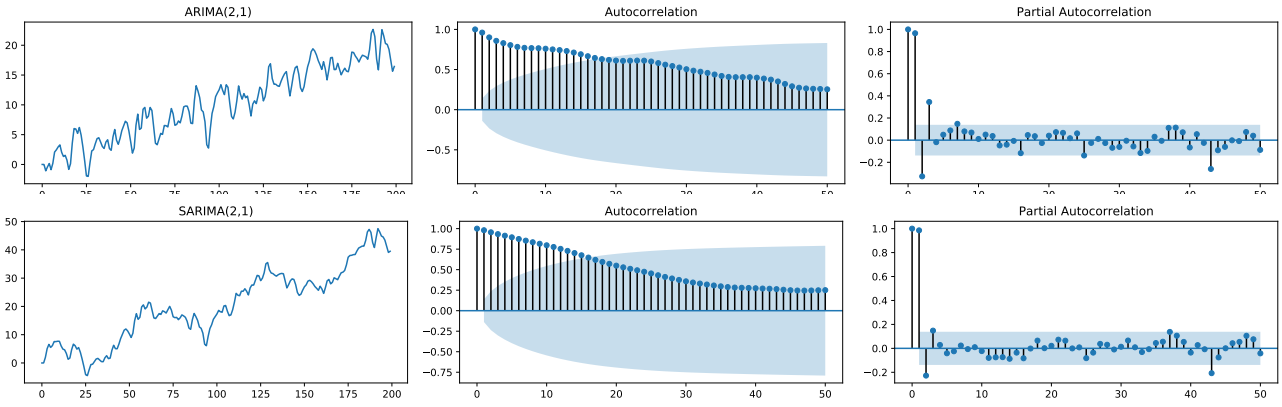


Abbildung 10.2. Simulation verschiedener SARIMA-Prozesse und deren Autokorrelationsfunktionen. Beide Prozesse sind nichtstationär.

10.4 SARIMA-Modelle in `statsmodels`

Die zentrale Klasse für allgemeine SARIMA-Modelle in der Bibliothek `statsmodels` ist `SARIMAX`

`https://www.statsmodels.org/stable/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html`

im Modul `sarimax`, die mit der Anweisung

[2] Katzenberger (2013).

```
from statsmodels.tsa.statespace.sarimax import SARIMAX
```

importiert werden kann. Notwendiger Parameter zur Erzeugung einer SARIMAX-Instanz ist hierbei `endog` für die zu modellierende Zeitreihe. Die Zeitreihe kann hierbei ein numerisches Array sein, möglich und in der Regel zu empfehlen ist jedoch eine Pandas Series. Der Vorteil einer Pandas Series ist, dass die Zeitpunkte der Zeitreihenwerte auch in Datums- oder Zeitformaten gespeichert und die Art der Messperiode (`freq`) festgelegt werden kann. Beispielsweise erzeugen die Anweisungen

Pandas `Series`

```
y = pd.Series([1000, 2500, 500, 3000])
y.index = pd.DatetimeIndex(
    ["2020-01-01", "2020-04-01", "2020-07-01", "2020-10-01"],
    freq='QS-JAN')
```

die Zeitreihe

```
2020-01-01    1000
2020-04-01    2500
2020-07-01    500
2020-10-01    3000
Freq: QS-JAN, dtype: int64
```

deren Messzeitpunkte quartalsweise zum 1. Januar (`"QS-JAN"`) auftreten. Die möglichen Strings für den Datumsoffset sind unter dem URL

`https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#dateoffset-objects`

aufgelistet. Gängig sind `"D"` für tägliche Zeitpunkte, `"M"` und `"MS"` für Monatsdaten zum

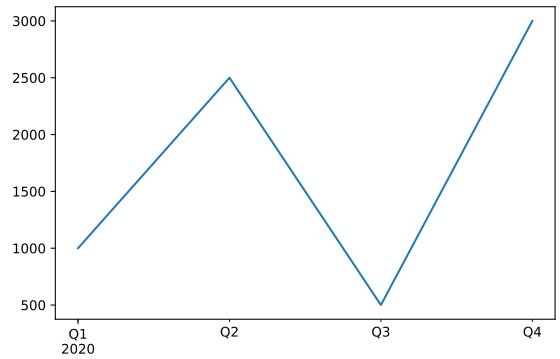


Abbildung 10.3. Plot einer Trajektorie einer als Pandas Series gepeicherten Zeitreihe

jeweils Monatsersten bzw. Monatsletzten, `"Q"` und `"QS"` für Quartalsdaten zum Monatsanfang oder -ende, sowie `"AS"` und `"A"` für Jahresdaten zum Jahresanfang bzw. -ende. Mit dem Plotbefehl `y.plot()` wird damit die Zeitreihentrajektorie in Abbildung 10.3 dargestellt.

10.5 Parameter zur Erzeugung eines SARIMAX-Modells

Die Ordnung des Modells wird mit dem Parameter `order` bestimmt, die ein Tupel (p, d, q) von drei Zahlen erwartet. Ferner wird die saisonale Komponente $(P, D, Q)_s$ durch den Parameter `seasonal_order` mit einem Tupel (P, D, Q, s) von vier Zahlen festgelegt. Die vorgegebenen Standardwerte der beiden Parameter sind `order = (1, 0, 0)` und `seasonal_order = (0,0,0,0)`, d.h. das Modell repräsentiert einen AR(1)-Prozess.

`order`

`seasonal_order`

trend

Ein weiterer in der Praxis sehr nützlicher optionaler Parameter ist `trend`, der das der Zeitreihe zugrunde gelegte deterministische Trendpolynom $A(t)$ bestimmt. Er kann entweder einen der vier Strings `'n'`, `'c'`, `'t'`, `'ct'` annehmen oder eine Liste $[e_0, e_1, \ldots, e_k]$ von Nullen und Einsen, $e_i \in \{0, 1\}$. Die Strings bewirken, dass gar kein Trend, ein konstanter Trend, ein in t linearer Trend oder ein linearer und konstanter Trend angesetzt wird. Eine Liste $[e_0, e_1, \ldots, e_k]$ von Nullen und Einsen bestimmt die nichtverschwindenden Terme des angesetzten Polynoms, d.h. $A(t) = \sum_0^k e_i a_i t^i$. Die Liste $[1, 1, 0, 1]$ beispielsweise führt zu dem angesetzten Trendpolynom $A(t) = a_0 + a_1 t + a_3 t^3$.

10.6 Fitten eines SARIMAX-Modells

fit()

Die wichtigste Methode von SARIMAX ist `fit()`. Wie im maschinellen Lernen üblich passt es die Parameter des Modells an die beobachteten Daten an. Im Unterschied zu dem in Scikit-Learn realisierten Prinzip, dass aus einem Modell nach dem Fitten ein neues Modell entsteht, mit dem der Datenbestand verändert und damit ein weiteres Modell gefittet werden kann, erzeugt die Methode `fit()` in `statsmodels` als Ergebnis eine Instanz einer neuen Klasse, der Klasse `MLEResults`[3]. In der Dokumentation von `statsmodel`[4] wird entsprechend auch empfohlen, das Ergebnis des Fittens mit einer eigenen Variable zu benennen, also beispielsweise

```
1 from statsmodels.tsa.statespace.sarimax import SARIMAX
2 model = SARIMAX(y, order=(2,0,0), trend='ct')
3 result = model.fit()
```

Um jedoch einen der Grundgedanken des maschinellen Lernens auch hier aufzunehmen und fortzuführen, wird in diesem Skript das Ergebnis des Fittens der Modellparameter wieder in der Variablen des Modells gespeichert, wie in dem Beispiel in Zeile 3 die Variable `model`. Natürlich ist dieser leichte Notationsmissbrauch nur in der subjektiven Abwägung des Autors gegenüber dem Umstand gerechtfertigt, dass das ungefittete Modell normalerweise keine weitere Verwendung findet. Falls Sie diese Abwägung anders bewerten, können und sollten Sie selbstverständlich für Ihre Programme die Empfehlung der Dokumentation von `statsmodels` einhalten.

10.7 Prognosen eines SARIMAX-Modells

in-sample prediction

out-of-sample forecasting

In `statsmodels` wird grundsätzlich zwischen den beiden Begriffen *predict* und *forecast* unterschieden. Während *predict* vorwiegend die Durchführung einer sogenannten *in-sample prediction* bezeichnet, also einer Wiedergabe von Prognosewerten, die das gefittete Modell für ein zeitliches Fenster aus dem Stichprobenbereich liefert, wird unter *forecasting* die echte Vorhersage des Modells für einen zeitlichen Bereich aus der Zukunft der Stichprobe verstanden, *out-of-sample forecasting* genannt. Entsprechend gibt es für ein gefittetet Zeitreihenmodell zwei Methoden, `predict` und `forecast`.[3]

predict()

Die flexiblere Methode eines gefitteten SARIMAX-Modells zur Berechnung einer Prognose ist dabei jedoch `predict`, denn sie ermöglicht in Wirklichkeit sowohl die *in-sample prediction* als auch das *out-of-sample forecasting*. Die beiden wichtigen Parameter

[3] https://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.mlemodel.MLEResults.html

[4] https://www.statsmodels.org/stable/examples/notebooks/generated/statespace_sarimax_stata.html

sind von `predict` sind `start` und `end`, die die Indexe von Start- und Endzeitpunkt der Prognose bezogen auf die Stichprobe y bestimmen, also für n_0 und n_k beispielsweise

```
pred = model.predict(start=n_0, end=n_k)
```

Beide Parameter sind optional, ihre Standardwerte sind `start=0` und `end=`m, wobei m den letzten Index der Stichprobe y bezeichnet. Mit anderen Worten wird in diesem Fall also eine *in-sample prediction* über die gesamte Stichprobe durchgeführt.

Zur Berechnung eines reinen *out-of-sample forecasting* ist auch die Methode `forecast` aufrufbar, die als Parameter `steps` nur die Anzahl der Prognosewerte in die Zukunft nach dem letzten Wert der Stichprobe benötigt. Der Parameter ist optional, Standardwert ist allerdings 0, d.h. es wird überhaupt keine Vorhersage gemacht. In der Praxis wird man diese Methode nur verwenden, wenn explizit eine *out-of-sample* Prognose gewollt ist, für die meisten Fälle wird `predict` einsetzbar sein.

`forecast()`

Die Werte für die Parameter `start`, `end` und `step` müssen nicht ganzzahlige Werte sein, sondern können auch vom Typ `datetime` sein.

10.8 Wahl eines SARIMA-Modells

Ist eine Zeitreihe gegeben, so ist ihre Analyse wie beim maschinellen Lernen generell eine komplexe und von den vorliegenden Daten individuell abhängig. Es gibt hier kein eindeutig definiertes Verfahren, sondern eher eine Sammlung heuristischer Daumenregeln. Dennoch werden die folgenden Schritte als grobe Handlungsanweisung einer Zeitreihenanalyse allgemein anerkannt[5].

1. *Datenimport:* Daten als Pandas DataFrame einlesen, Struktur analysieren und Daten vorverarbeiten

2. *Erste Visualisierung:* Trajektorie der Zeitreihe plotten

3. *Datengrobanalyse:* Die Zeitreihe vorläufig kategorisieren, möglicherweise die Daten transformieren

4. *Bestimmung der Hyperparameter:* Ordnungsparameter $(p, d, q, (P, D, Q)_s)$ schätzen

5. *Diagnose:* Modell mit statistischen Kennzahlen evaluieren und ggf. zurück nach Schritt 3 oder 4.

6. *Bestimmung des Modells:* Das bestevaluierte Modell wählen.

Der erste Schritt besteht darin, die Daten in ein Programm zu importieren. Zwar sind `Python Pandas` die meist vorliegenden Datenformate bekannt, z.B. textbasierte CSV-Dateien (Endung `.csv` oder oft auch `xls`), SPSS (`.sav`). Häufig liegen die Daten in einer Struktur vor, die zunächst im Rahmen einer Vorverarbeitung (*preprocessing*) modifiziert werden muss, so dass sie eine verarbeitbare Zeitreihe, oder verarbeitbare Zeitreihen, ergeben.

Wie bei fast jeder Datenanalyse sollten wir zunächst die Trajektorie der Zeitreihe als Funktionsgraphen erstellen, um einen ersten Eindruck zu gewinnen. Sind die Werte wachsend oder periodisch? Gibt es einen Trend? Sind die Schwankungen um den

[5] Shumway und Stoffer (2017):S. 135ff.

Trend stets etwa gleich? Insbesondere zur Stabilierung der Varianzen kann eine *Box-Cox-Transformation* mit dem Parameter $\alpha \geqq 0$ verwendet werden:

$$y_t \quad \mapsto \quad y_t^{(\alpha)} = \begin{cases} \log y_t & \text{für } \alpha = 0, \\ \frac{1}{\alpha}\,(y_t^{\alpha} - 1) & \text{für } \alpha \neq 0. \end{cases} \tag{10.8}$$

Die gebräuchlichste ist die logarithmische Transformation ($\alpha = 0$), wobei in der Regel die Basis des Logarithmus für die Analyse keine Rolle spielt. Sie wird häufig bei Wachstumsprozessen angewendet.

10.9 Übungsaufgabe

Aufgabe 10.1. (a) Unter dem Link `https://fred.stlouisfed.org/series/GDPC1` kann man die historischen Daten des realen Bruttoinlandsprodukts der USA von März 1947 bis April 1978 erhalten, die in Beispiel 8.1 erwähnt wurden. Die Daten liegen als Quartalsdaten jeweils zum 1. April, 1. Juli, 1. Oktober und 1. Januar vor. Schreiben Sie ein Python-Programm, in dem die Daten eingelesen und als Pandas Series gespeichert werden und plotten Sie die Zeitreihe.

(b) Wenden Sie als Trainingsdaten die ersten 70 % der Zeitreihenwerte auf die Zeitreihe ein AR(2)-Modell mit SARIMAX mit linearem Trend an.

(c) Berechnen Sie die Prognose für die letzten 30 % der Zeitreihenwerte als Testdaten und plotten Sie sie mit Hilfe eines Pandas DataFrames gemeinsam mit der Trajektorie der gesamten Zeitreihe (also Trainings- und Testdaten).

A Appendix

A.1 Solutions to selected problems

Problem 1.1 (a) A random variable X on a sample space Ω is a mapping $X : \Omega \to \mathbb{R}$. (Actually the topic of this part of the problem is less a question about random variables than about the mathematical notation of a mapping between two sets!)

(b) The table of values of the random variable X reads:

dots ω	1	2	3	4	5	6
$X(\omega)$	0	1	0	1	0	1

Since the probability of each outcome equals $\frac{1}{6}$, we have $P(X = 1) = P(2)+P(4)+P(6) = \frac{1}{6}+\frac{1}{6}+\frac{1}{6} = \frac{1}{2}$. The implicit assumption made by this deduction is that the six outcomes are uniformly probable, i.e., that each occurs with the probability $\frac{1}{6}$. (This part of the problem thus more deals with the formal derivation than with the – intuitively "somehow logical" – result: Every random variable assigns a real number to each outcome $\omega \in \Omega$; but we can determine the probability distribution only if we know the probabilities of the outcomes.)

Problem 2.1 We first have to associate the three statements to their respective roles of inference, i.e., rule, premise, and conclusion:

Rule $A \Rightarrow B$*:*	"When it rains, the street is wet"
Premise A:	"It rains"
Conclusion B:	"The street is wet"

Therefore,

Deduction	**Induction**	**Abduction**
When it rains, the street is wet	It rains	When it rains, the street is wet
It rains	The street is wet	The street is wet
The street is wet	When it rains, the street is wet	H_1: It rains
		H_2: It does not rain.
		H_3: The street is sprayed.
		H_1: It rains

(Here the third hypothesis H_3 of course is only exemplary.)

Problem 2.2 We have $P(X \mid A) = \frac{1}{3} \cdot (\frac{2}{3})^4$ and $P(X \mid B) = \frac{2}{3} \cdot (\frac{1}{3})^4$. Let formally $M_1 = A$ denote the model that the coin A is chosen, and $M_2 = B$ that coin B is chosen, respectively. Assuming that the prior probabilities of each model is equal,

$$P(A) = P(B) = \frac{1}{2}$$

we obtain by Equation (2.8) the probability ratio

$$\frac{P(A \mid X)}{P(B \mid X)} = \frac{P(X \mid A)}{P(X \mid B)} \cdot \frac{P(B)}{P(A)} = \frac{P(X \mid A)}{P(X \mid B)} = \frac{\frac{1}{3} \cdot (\frac{2}{3})^4}{\frac{2}{3} \cdot (\frac{1}{3})^4} = \frac{1 \cdot 2^4}{2 \cdot 1^4} = 2^3 = 8, \tag{A.1}$$

in favor of type A.

Problem 4.1

Literal	Valid	Result / Explanation
`1.000e-0.2`	no	exponent after `e` must be an integer
`2e+1j`	yes	`20j`, a purely imaginary number
`0x567`	yes	`1383`, hexadecimal representation
`00x567`	no	Literals for hexadecimal numbers must have precisely one leading zero
`0o567`	yes	`375`, octal representation
`0o568`	no	8 is not an octal symbol
`'Größe'`	yes	`'Größe'`, string with Unicode symbols
`''Größe''`	no	Two apostrophs `''` define an empty string
`"Größe"`	yes	`'Größe'`, string with Unicode symbols
`b'Größe'`	no	Bytes can only contain ASCII symbols
`"Komm 'rein!"`	yes	A string in quotation marks may contain apostrophs – and vice versa
`00023e001`	yes	`230.0`, floating-point number; mantissa and exponent must be integers, but may have leading zeros and are interpreted as decimal numbers
`(1; 2; 3)`	no	Round brackets define tuples, the entries of which must be separated by commas

Problem 4.2

Object of Reality	Data Type	Exemplary Expression
radius of atoms	`float`	`3.2e-13`
name of flower	string (`str`)	`'Tulpe'`
name of participants of a race	`list` or `set` of Strings	`['Leonie', 'Stephanie', 'Alina']` or `{'Leonie', 'Stephanie', 'Alina'}`
score of a soccer match (e.g., 3:1)	tuple of two integers	`(3,1)`
name, prename and age of a person	tuple with three entries	`('Schmitz', 'Otto', 24)`
name, prename and age of a participant of a race	a list of tuples	`[('Meier', 'Leonie', 23), ('Müller', 'Stephanie', 21), ('Schmitz', 'Alina', 24)]`
table, in which the chemical element symbols are stored with their English and German names (e.g., H → hydrogen, Wasserstoff)	dictionary with symbols as keys and a list of English and German notions as values	`{'H': ['hydrogen','Wasserstoff'], 'O': ['oxygen','Sauerstoff'], 'C': ['carbon', 'Kohlenstoff']}`

Problem 4.3 Assuming the alphabet `'abc'` we obtain the output of all words with two letters as follows:

```
alphabet = 'abc'
for a in alphabet:
    for b in alphabet:
        print(a+b, end=' ')
```

With the parameter `end` the `print` method does not terminate with a line break (`'\n'`) but with a blank space (`' '`).

Problem 4.4 (a) For n elements the total number of k-digit combinations is n^k. Since here the number of base pairs is $n = 4$ and the number of digits is $k = 4$, the total number of combinations is $n^k = 4^4 = 256$.

(b) To print all 256 four-digit combinations of the base pairs AT, TA, GC and CG, it is appropriate in Python to firts define a list of four pairs as strings and then to run through a nested loop of depth four:

```
# Print out all DNA sequences with the four base pairs
base_pairs = ['AT', 'TA', 'GC', 'CG']
for a in base_pairs:
    for b in base_pairs:
        for c in base_pairs:
            for d in base_pairs:
                print(a, b, c, d)
```

Problem 4.5 The descreption of the random function `randint`, as given under

`https://numpy.org/devdocs/reference/random/generated/numpy.random.randint.html`,

reads `numpy.random.randint(low, high=None, size=None, dtype=int)`: it expects the value `low` mandatorily; if `high` is not set, a random number is returned from the intervall `[0, low)`, otherwise from the intervall `[low, high)`. Then a solution may look like:

```
from numpy.random import randint
import time
print('Multiplication trainer')
print('----------------------')
start = time.time()
for i in range(5):
    m = randint(1,10)
    n = randint(1,10)
    result = 0    # can never be a product of positive numbers
    while result != m*n:
        result = int(input(str(m) + '*' + str(n) + '='))
        if result == m*n:
            print('Correct!')
        else:
            print('Unfortunately wrong! Try again ...')
end = int(time.time() - start)
print('For the tasks you needed', end, 'seconds.')
```

Problem 4.6 (a) The function `randn`$(d_1, d_2, \ldots, d_n)$ of the module `numpy.random`, according to the API `https://numpy.org/devdocs/reference/random/generated/numpy.random.randn.html`, generates one or several standard normally distributes samples as an array of dimension n (or expressed geometrically ausgedrückt: a "tensor of order n"). If no parameter is inputted, a random number $z \in \mathbb{R}$ (an array of dimension 0) is returned, for an input of one parameter an array ("tensor of order 1") of length d_1, for two parameters a matrix as an array consisting of d_1 Arrays of length d_2 (i.e., $d_1 \times d_2$), etc. For instance,

```
np.random.randn(3)
```

generates an array with three random numbers, say

```
[0.94759771,0.1613764, -0.25537882]
```

whereas

```
np.random.randn(2,3)
```

generates an array of random arrays with 3 entries, say:

```
[[ 0.24416202,  0.67809254, -0.15421969],
 [-1.19499275,  0.15050603, -1.84171316]]
```

(b) If we want to achieve a random sequence the mebers of which are distributed accordinhg to a normal distribution $N(\mu, \sigma)$ with mean μ and standard deviation σ, we apply:

```
sigma * np.random.randn(...) + mu
```

(c) With (a) and (b) a program to generate an error bar diagram with a $N(0, \sigma)$-normally distributed random array with standard deviation $\sigma = 0{,}001 \cdot \max_{f\in[0,\infty)} \{B(f,\ 2{,}75)\}$ reads like:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from numpy import exp
4  from scipy.constants import pi, h, c, k
5
6  def planck(f,T):
7      return 2 * h * f**3 / (c**2 * exp(h*f / (k*T)) - 1)
8
9  n = 67
10 x = np.linspace(0, 7e11, n)
11 y = planck(x, 2.75)
12 sigma = 0.001 * max(y)
13 e = sigma*np.random.randn(n)
14
15 plt.errorbar(x, y, yerr=e)
16 plt.show()
```

For an error bar diagramm with $\sigma = 0{,}01 \cdot \max_{f\in[0,\infty)} \{B(f,\ 2{,}75)\}$ line 12 must be modified:

```
sigma = 0.01 * max(y)
```

(or `1e-2` instead of `0.01`), and for $\sigma = 0{,}1 \cdot \max_{f\in[0,\infty)} \{B(f,\ 2{,}75)\}$:

```
sigma = 0.1 * max(y)
```

Problem 5.1 Following the example of the case study from section 5.4.4 the program could look like as follows.

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

#=====================================================================
daten = pd.read_csv("datasets/semiconductor-electron-mobility.csv", sep="\t")
#=====================================================================

x = np.c_[daten["x"]].ravel() # extracts feature values as a column vector
y = np.c_[daten["y"]].ravel() # extracts feature values as a column vector

plt.scatter(x,y)
plt.show()

def f(x,t0,t1,t2,t3,t4,t5,t6):
    return (t0 + t1 * x + t2 * x**2 + t3 * x**3) / (1 + t4 * x + t5 * x**2 + t6 * x**3)

coefs, cov = curve_fit(f, x, y, p0=(1000,1000,400,40,1,0.5,0.05))
y_pred = f(x, *coefs)   # model predictions
print("Fitted coefficients:", [round(t,3) for t in coefs])

xs = np.linspace(min(x), max(x), 100)  # take 100 points for a smooth regression line ...
plt.scatter(x, y)
plt.plot(xs, f(xs, *coefs), 'red')
plt.show()

# Model evaluation:
from sklearn.metrics import r2_score
R2 = r2_score(y, y_pred)
print("R^2 =", f"{R2:.3%}")

def bic(e, k):
    return np.log(np.var(e)) + k*np.log(len(e))
print("BIC =", f"{bic(y - y_pred, len(coefs)):.0f}")
```

Results: The Python `curve_fit` solves this problem classified as difficult. Especially, we obtain $R^2 = 99.951\%$, BIC = 30. That is, the coefficient of determination is rather well, the value of the BIC is meaningful only in comparison to another model.

Problem 5.2 Following the example of the case study from section 5.4.4 the program could look like as follows.

```
%matplotlib inline
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

# 1.  Import data as a Pandas DataFrame, preprocess them for scipy curve_fit, and plot them:
df = pd.read_csv("./datasets/advertizing-and-sales.csv", sep='\t')  features = ["Advertizing expenses"]
target = "Sales volume"             # dependent variable
X = np.c_[df[features]].ravel()     # extracts feature values as a column vector
y = np.c_[df[target]].ravel()       # extracts target values as a column vector

plt.scatter(X, y)
plt.show()

# 2.  Curve fit:
def f1(x,a,b)  : return a + b*x
def f2(x,a,b)  : return a + b*np.sqrt(x)
def f3(x,a,b,c): return a + b*x**c
def f4(x,a,b,c): return a + b*np.exp(c*x)
coefs1, cov1 = curve_fit(f1, X, y)
coefs2, cov2 = curve_fit(f2, X, y)
coefs3, cov3 = curve_fit(f3, X, y, p0=(1,1,0.1))
coefs4, cov4 = curve_fit(f4, X, y, p0=(1,-1,0.05))

# 3.  Output model parameters:
print("Coefficients of models:")
```

```
27 print(" linear:",      [round(t,2) for t in coefs1])
28 print(" square root:", [round(t,2) for t in coefs2])
29 print(" power:",       [round(t,2) for t in coefs3])
30 print(" exponential:", [round(t,2) for t in coefs4])
31
32 # 4.  Plot data and regression curves:
33 xp = np.linspace(min(X), max(X), 100)   # take 100 points for smooth regression lines ...
34 plt.scatter(X, y)
35 plt.plot(xp, f1(xp, *coefs1), label="linear model")
36 plt.plot(xp, f2(xp, *coefs2), label="square root model")
37 plt.plot(xp, f3(xp, *coefs3), label="power model")
38 plt.plot(xp, f4(xp, *coefs4), label="exponential model")
39 plt.legend()
40 plt.show()
41
42 # 4 Evaluate models
43 # 4.1 Cofficients of determination:
44 from sklearn.metrics import r2_score
45 R2_1 = r2_score(y, f1(X, *coefs1))
46 R2_2 = r2_score(y, f2(X, *coefs2))
47 R2_3 = r2_score(y, f3(X, *coefs3))
48 R2_4 = r2_score(y, f4(X, *coefs4))
49
50 print("Coefficients of determination")
51 print(
52     " linear:", f"{R2_1:.2%},", "\tsquare root:", f"{R2_2:.2%},",
53     "\tpower:", f"{R2_3:.2%},", "\texponential:", f"{R2_4:.2%}"
54 )
55
56 #4.2 BICs:
57 def bic(e, k):
58     return np.log(np.var(e)) + k*np.log(len(e))
59
60 print(
61     " BIC₁ =",    f"{bic(y - f1(X, *coefs1), len(coefs1)):.1f},",
62     "\t\tBIC₂ =", f"{bic(y - f2(X, *coefs2), len(coefs2)):.1f},",
63     "\t\tBIC₃ =", f"{bic(y - f3(X, *coefs3), len(coefs3)):.1f},",
64     "\t\tBIC₄ =", f"{bic(y - f4(X, *coefs4), len(coefs4)):.1f}"
65 )
```

Note the input of the initial values `p0` for the model parameters in lines 22 and 23. The estimated model parameters and the corresponding coefficients of determination are listed in the following table.

Model	**Parameter**	R^2	**BIC**
linear:	$f(x) = 112.64 + 6{,}78\ x$	${R_1}^2 = 95.31\%$	10.5
square root:	$f(x) = 31.71 + 49{,}65\ \sqrt{x}$	${R_2}^2 = 98.67\%$	9.3
power:	$f(x) = 9.55 + 65.79\ x^{0.44}$	${R_3}^2 = 98.72\%$	12.0
exponential:	$f(x) = 359{,}22 - 283{,}84\ \mathrm{e}^{-0{,}05}$	${R_4}^2 = 98{,}82\%$	11.9

(A.2)

The regression curves of the models are shown in Figure A.1. Results: Thus, the expo-

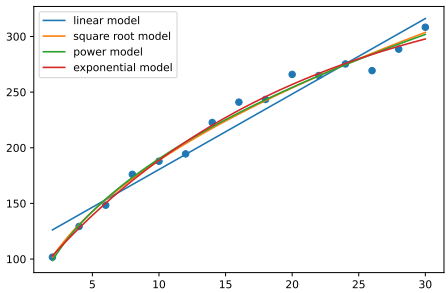


Figure A.1. Regression curves showing the effect of advertising

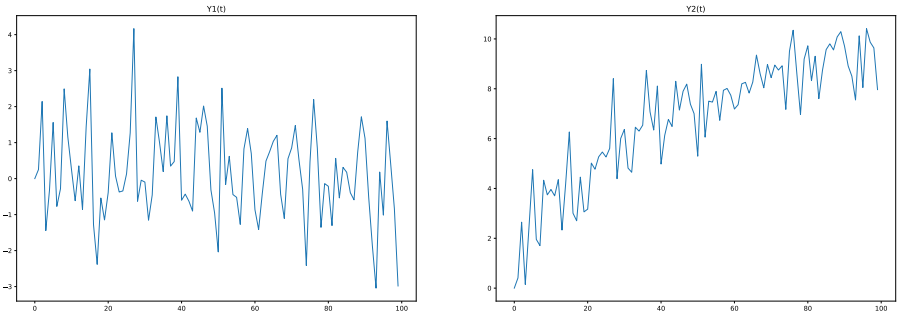
nential model has the best goodness of fit with a coefficient of determination of 98.82 %. However, if one weighs the complexity of the models against R^2, one can argue with

the BIC – and thus with Occam's razor – and conclude that, due to the lower number of parameters, the square root model with the lowest BIC of 9.3 can be considered as the "best" one.

Problem 7.1 (a)

```
import numpy as np
import matplotlib.pyplot as plt

def Y1(t, e):
    return np.sin(t) + e

def Y2(t, e):
    return np.sqrt(t) + e

size = 100
e = np.random.randn(size)
y1 = np.zeros(size)
y2 = np.zeros(size)

for t in range(1,size):
    y1[t] = Y1(t,e[t])

for t in range(1,size):
    y2[t] = Y2(t,e[t])

fig = plt.figure(figsize=(24,8))
ax1 = fig.add_subplot(121)
ax1.set_title("Y1(t)")
ax1.plot(y1)
ax2 = fig.add_subplot(122)
ax2.set_title("Y2(t)")
ax2.plot(y2)
plt.show()
```

The function plots are then given as follows:



(b) For Y_1, the mean and the standard deviation are constant in each case, i.e., Y_1 is stationary. However, since the mean and the standard deviation of Y_2 depend on time, Y_2 cannot be stationary. We can see these two findings in the function graphs: Y_1 moves in a corridor between -4 and 4 around its mean, while Y_2 shows growth.

(c*) The mean of f on the intervall $[0, t]$ is given approximately for great t by

$$\mu(t) = \frac{1}{t}\int_0^t f(\tau)\ \mathrm{d}\tau. \tag{A.3}$$

The innovation terms ε_t play no role here since they have the mean value zero. For μ_1 and μ_2 from (b), the following results are obtained

$$\mu_1(t) = \frac{1}{t}\int_0^t \sin\tau\ \mathrm{d}\tau = -\frac{\cos\tau}{\tau}\Big|_0^t = \frac{1-\cos t}{t} \quad \to \quad 0 \qquad \text{as } t \to \infty$$

and

$$\mu_2(t) = \frac{1}{t}\int_0^t \sqrt{\tau}\ \mathrm{d}\tau = \frac{2}{3}\sqrt{\tau}\Big|_0^t = \frac{2}{3}\sqrt{t} \quad \to \quad \infty \qquad \text{as } t \to \infty.$$

Thus for large values of t the mean of the first time series converges to 0, while that of the second diverges with $O(\sqrt{t})$.

Problem 8.1 (a) An implementation according to the task may look like as follows:

```
import numpy as np
import matplotlib.pyplot as plt

def AR1(phi1):
    size = 1000
    e = np.random.randn(size)
    y = np.zeros(size)
    for t in range(1,size):
        y[t] = phi1 * y[t-1] + e[t]
    return y

fig = plt.figure(figsize=(24,8))
ax1 = fig.add_subplot(411)
ax1.set_title("AR1(-1)")
ax1.plot(AR1(-1))
ax2 = fig.add_subplot(412)
ax2.set_title("AR1(0.5)")
ax2.plot(AR1(0.5))
ax3 = fig.add_subplot(413)
ax3.set_title("AR1(1)")
ax3.plot(AR1(1.0))
ax4 = fig.add_subplot(414)
ax4.set_title("AR1(1.01)")
ax4.plot(AR1(1.01))
plt.savefig('AR1-processes.pdf')
plt.show()
```

The otput is depicted in Figure A.2.

(b) For the first three graphs, the mean μ of the respective time series is zero, the variance of the second time series is $\sigma^2 = 1$, and of the third (a random walk) $\sigma^2 = t$.

(c) An AR(1) process with $|\varphi| < 1$ is stationary; with $\varphi = 1$ it is a random walk (example 7.4) and therefore non-stationary with example 8.5; with $\varphi > 1$ it is hyperexponentially growing or shrinking.

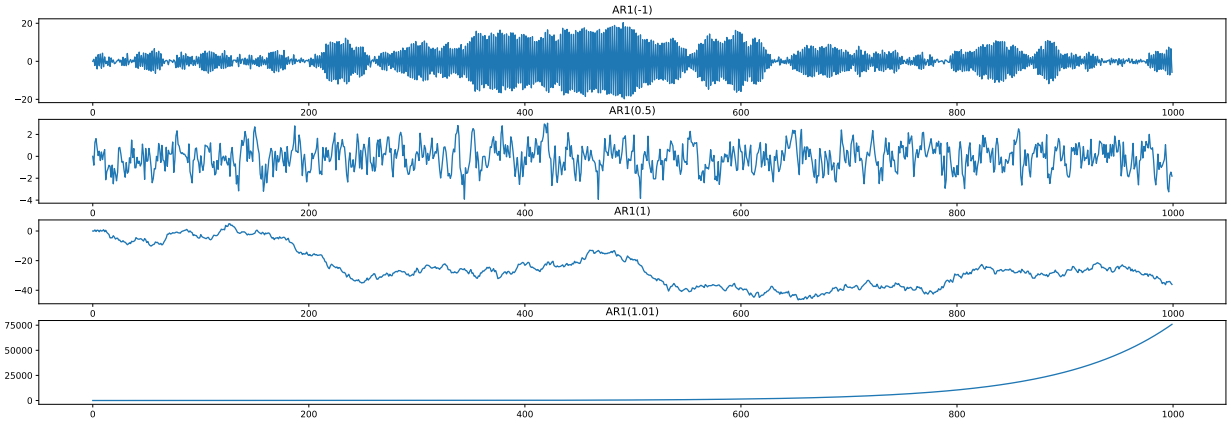


Figure A.2. AR(1)-Prozesse mit den Parametern $\varphi = 0.5$, $\varphi = 1$, $\varphi = 1.01$.

Problem 10.1 (a) = steps 1 and 2, (b) = step 3, (c) = steps 4 and 5:

```
import os, pandas as pd, matplotlib.pyplot as plt
from statsmodels.tsa.statespace.sarimax import SARIMAX

# 1.  Einlesen der CSV-Datei in Pandas:
verzeichnis  = "datasets"
datei  = "GDPC1.csv" # https://fred.stlouisfed.org/series/GDPC1

df = pd.read_csv(os.path.join(verzeichnis,  datei), sep=",")
df.index = pd.DatetimeIndex(df['DATE'], freq='QS-Apr')
del df['DATE']
df.rename(columns = {'GDPC1':'Real GDP'}, inplace=True)

# 2.  Filtern der gewünschten Daten und als Pandas Series speichern:
y = pd.Series(df['1947-04-01':'1978-04-01'][df.columns[0]])

# 3.  Konfigurieren der Trainings und Testdaten:
train_size = int(len(y) * 0.7)
test_size  = len(y) - train_size
y_train = y[:y.index[train_size]] # hier als Series
y_test  = y[y.index[train_size]:] # hier als Series

# 4.  Modell SARIMA((2,0,0) x (0,0,0)_0) anwenden:
model = SARIMAX(y_train, order=(2,0,0), trend='ct')
model = model.fit()
print(model.summary())

# 5.  Out-of-sample Prognose:
pred = model.predict(start=train_size, end=train_size+test_size-1)
pd.DataFrame({'Real':y,'Prediction':pred}).plot()
plt.show()
```

Instead of the `predict` method in step 5, we could have used the `forecast` method here:

```
# 5.  Out-of-sample Prognose:
forecast = model.forecast(steps=test_size - 1)
```

```
pd.DataFrame({'Real':y,'Forecast':forecast}).plot()
plt.show()
```

The respective function plots are depicted in Figure A.3. Note that in step 1 the measure-

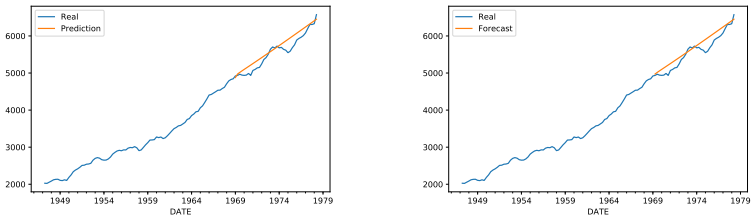


Figure A.3. Trajectory of the time series and prediction of the test data with `predict` (left) or with `forecast` (right)

ment period could have been set as `"QS-Jan"`, or also `"QS-Jul"` oder `"QS-Oct"`.

A.2 Heuser über Samuelsons Multiplikator

Der Mathematiker Harro Heuser (1927–2011) beschreibt in Kapitel 7 „Rekursive Definitionen und induktive Beweise. Kombinatorik“ seines Lehrbuchs über Analysis[1] als eine Anwendung der geometrischen Reihe in der Volkswirtschaft die Idee des Samuelson'schen Akzelerators. Der gesamte Abschnitt wird hier im originalen Wortlaut wiedergegeben. Mit „Mark“ wird hier das vom 21. Juni 1948 bis zum 31. Dezember 1998 in Deutschland gültige gesetzliche Zahlungsmittel bezeichnet.

Auswirkungen von Investitionen auf das Volkseinkommen

Angenommen, die (produzierenden und konsumierenden) Mitglieder einer Volkswirtschaft geben durchgehend einen Bruchteil q $(0 < q < 1)$ ihres Einkommens für Verbrauchsgüter aus (q ist die *Grenzneigung zum Verbrauch*). Nun möge ein Unternehmer eine Investition von K Mark tätigen (Bau einer Fabrik, Anschaffung von Maschinen usw.). Die *Erstempfänger* dieses Betrags (Maurer, Installateure, Maschinenbauer, ...) geben nach unserer Annahme qK Mark aus, die Empfänger *dieses* Betrags (*Zweitempfänger*) verbrauchen q^2K Mark usw. Nachdem die n-ten Empfänger q^nK Mark ausgegeben haben, sind insgesamt Ausgaben in Höhe von

$$K \sum_{i=0}^{n} q^i = K \, \frac{1-q^{n+1}}{1-q} = K \, \frac{1}{1-q} - K \frac{q^{n+1}}{1-q} \text{ Mark} \tag{A.4}$$

getätigt worden, und um diesen Betrag hat sich das Volkseinkommen erhöht (s. die „Volkswirtschaftslehre“ von Paul Samuelson, Köln-Deutz 1958, S. 248ff.[2]) [...] Wegen (A.4) wird also $K/(1-q)$ [für große n] hinreichend gut die durch die Erstinvestition von K Mark bewirkte Erhöhung des Volkseinkommens angeben.

[1] Heuser (1980):S. 67.
[2] heute: Samuelson und Nordhaus (1995:§24.B)

Literatur

Backhaus, K., B. Erichson und R. Weiber (2015). *Fortgeschrittene Multivariate Analysemethoden*. 3. Aufl. Springer Gabler: Berlin Heidelberg. DOI: 10.1007/978-3-662-46087-0.

Backhaus, K. et al. (2016). *Multivariate Analysemethoden*. 14. Aufl. Springer Gabler: Berlin Heidelberg. DOI: 10.1007/978-3-662-56655-8.

Bandelow, C. (1989). *Einführung in die Wahrscheinlichkeitstheorie*. 2. Aufl. BI Wissenschaftsverlag: Mannheim Wien Zürich.

Bauer, H. (1991). *Wahrscheinlichkeitstheorie*. 4. Aufl. Walter de Gruyter: Berlin New York.

Blanchard, O. J. (Mai 1981). „What is Left of the Multiplier Accelerator?“ In: *The American Economic Review* 71(2). http://www.jstor.org/stable/1815709, S. 150–154.

Bofinger, P. (2007). *Grundzüge der Volkswirtschaftslehre. Eine Einführung in die Wissenschaft von Märkten*. 2. Aufl. Pearson Studium: München.

Brandt, S. (1999). *Datenanalyse*. 4. Aufl. Spektrum Akademischer Verlag: Heidelberg Berlin.

Brockwell, P. J. und R. A. Davis (1991). *Time Series: Theory and Methods*. 2. Aufl. Springer. DOI: 10.1007/978-1-4419-0320-4.

— (2016). *Introduction to Time Series and Forecasting*. 3. Aufl. Springer. DOI: 10.1007/978-3-319-29854-2.

Burke, K. D. et al. (2018). „Pliocene and Eocene provide best analogs for near-future climates“. In: *Proceedings of the National Academy of Sciences* 115(52), S. 13288–13293. ISSN: 0027-8424. DOI: 10.1073/pnas.1809600115.

Büttner, U. (2008). *Weimar. Die überforderte Republik 1918–1933*. Klett-Cotta: Stuttgart.

Campbell, J. Y., A. W. Lo und A. C. MacKinlay (1997). *The Econometrics of Financial Markets*. Princeton University Press: Princeton.

Cowpertwait, P. S. P. und A. N. Metcalfe (2009). *Introductory Time Series with R*. Springer: Dordrecht Heidelberg London New York. DOI: 10.1007/978-0-387-88698-5.

de Vries, A. (2020). *Netzökonomie Lerneinheit 2. Einführung in das maschinelle Lernen mit Python*. Vorlesungsskript. Hagen. URL: https://fh-swf.sciebo.de/s/y80tdcBeby5mxow.

Deistler, M. und W. Scherrer (2018). *Modelle der Zeitreihenanalyse*. Birkhäuser: Cham. DOI: 10.1007/978-3-319-68664-6.

Denis, D. J. (2021). *Applied Univariate, Bivariate, and Multivariate Statistics Using Python: A Beginner's Guide to Advanced Data Analysis*. 2. Aufl. Wiley: Hoboken. ISBN: 9781119578147. DOI: 10.1002/9781119583004.

Downey, A. B. (2011). *Think Stats. Probability and Statistics for Programmers*. 1. Aufl. Green Tea Press: Needham, Massachusetts. URL: https://greenteapress.com/thinkstats/.

Durbin, J. und S. J. Koopman (2012). *Time Series Analysis by State Space Methods*. 2. Aufl. Oxford University Press: Oxford.

Felderer, B. und S. Homburg (1989). *Makroökonomik und neue Makroökonomik*. 4. Aufl. Springer-Verlag: Berlin etc.

Feldman, D. R. et al. (2015). „Observational determination of surface radiative forcing by CO_2 from 2000 to 2010“. In: *Nature* 519(7543), S. 339–343. DOI: 10.1038/nature14240.

Flach, P. A. und A. C. Kakas, Hrsg. (2000). *Abduction and Induction: Essays on their Relation and Integration*. Applied Logic Series. Springer Science+Business Media: Dordrecht. DOI: 10.1007/978-94-017-0606-3.

Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly: Sebastopol.

Goldberg, S. (1958). *Introduction to Difference Equations*. John Wiley & Sons: New York.

Goswami, A. (1997). *Quantum Mechanics*. 2. Aufl. Wm. C. Brown publishers: Dubuque, IA.

Handl, A. und T. Kuhlenkasper (2017). *Multivariate Analysemethoden: Theorie und Praxis mit R*. 3. Aufl. Statistik und ihre Anwendungen. Springer: Berlin, Heidelberg. DOI: 10.1007/978-3-662-54754-0.

Hastie, T., R. Tibshirani und J. Friedman (2009). *The Elements of Statistical Learning*. 2. Aufl. Springer: New York. DOI: 10.1007/b94608.

Heisenberg, W. (1948). „Der Begriff »abgeschlossene Theorie« in der modernen Naturwissenschaft“. In: *Dialectica* 2, S. 331–336.

Heuser, H. (1980). *Lehrbuch der Analysis. Teil 1*. B.G. Teubner: Stuttgart.

Hull, J. C. (2000). *Options, Futures & Other Derivatives*. 4. Aufl. Prentice-Hall International: Upper Saddle River, NJ.

IPCC (Aug. 2021). *AR 6 Climate Change 2021. The Physical Science Basis*. Cambridge University Press: Cambridge New York. URL: https://www.ipcc.ch/report/ar6/wg1/.

James, G. et al. (2013). *An Introduction to Statistical Learning*. Springer: New York Heidelberg Dordrecht London. DOI: 10.1007/978-1-4614-7138-7.

Jouzel, J. et al. (Aug. 2007). „Orbital and Millennial Antarctic Climate Variability over the Past 800,000 Years“. In: *Science* 317(5839), S. 793–797. DOI: 10.1126/science.1141038.

Kalvelage, T. (2018). „Chronisten der Erdgeschichte“. In: *Spektrum der Wissenschaft* 10, S. 50–55. URL: https://spektrum.de/artikel/1757380.

Katzenberger, M. (2013). *Algorithmen zur Losgrößenoptimierung*. Bd. 3. Hagener Berichte der Wirtschaftsinformatik. Books on Demand: Norderstedt.

Koyré, A. (1992). *The Astronomical Revolution. Copernicus – Kepler – Borelli*. Dover Publications: New York.

Krahl, D., U. Windheuser und F.-K. Zick (1998). *Data Mining: Einsatz in der Praxis*. Addison-Wesley-Longman: Bonn. ISBN: 9783827313492.

Kreiß, J.-P. und G. Neuhaus (2006). *Einführung in die Zeitreihenanalyse*. Springer: Berlin Heidelberg. DOI: 10.1007/3-540-33571-4.

Krugman, P. R. und R. E. Wells (2006). *Macroeconomics*. Worth Publishers: New York.

Kuhn, T. S. (1962). *The Structure of Scientific Revolutions*. University of Chicago Press: Chicago.

MacKay, D. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press: Cambridge.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill. ISBN: 9780071154673.

Murphy, J. J. (2016). *Technische Analyse der Finanzmärkte*. FinanzBuch Verlag: München.

Murphy, K. P. (2012). *Machine Learning. A Probabilistic Perspective*. MIT Press: Cambridge London.

Neusser, K. (2011). *Zeitreihenanalyse in den Wirtschaftswissenschaften*. 3. Aufl. Vieweg + Teubner: Wiesbaden. DOI: 10.1007/978-3-8348-8653-8.

Ng, A. und K. Soo (2018). *Data Science – was ist das eigentlich?! Algorithmen des maschinellen Lernens verständlich erklärt*. Springer: Berlin. DOI: 10.1007/978-3-662-56776-0.

O'Neill, B. C. et al. (2017). „The roads ahead: Narratives for shared socioeconomic pathways describing world futures in the 21st century". In: *Global Environmental Change* 42, S. 169–180. ISSN: 0959-3780. DOI: 10.1016/j.gloenvcha.2015.01.004.

Palma, W. (2016). *Time Series Analysis*. Wiley: Hoboken.

Penrose, R. (2004). *The Road to Reality*. Vintage Books: New York.

Pole, A., M. West und J. Harrison (1994). *Applied Bayesian Forecasting and Time Series Analysis*. Chapman & Hall: New York.

Pourret, O., P. Naim und B. Marcot, Hrsg. (2008). *Bayesian Networks. A Practical Guide to Applications*. John Wiley & Sons: Chichester.

Rinne, H. und K. Specht (2002). *Zeitreihen. Statistische Modellierung, Schätzung und Prognose*. Vahlen: München.

Rotmans, J. et al. (2000). „Visions for a sustainable Europe". In: *Futures* 32(9–10), S. 809–831. URL: https://www.pik-potsdam.de/ateam/avec/peyresq2003/internal/articles_pdf/europe_scenarios.pdf.

Russell, S. J. und P. Norvig (2022). *Artificial Intelligence. A Modern Approach*. Pearson: Harlow.

Samuelson, P. A. (1939). „Interactions between the Multiplier Analysis and the Principle of Acceleration". In: *The Review of Economics and Statistics* 21(2), S. 75–78. DOI: 10.2307/1927758.

Samuelson, P. A. und W. D. Nordhaus (1995). *Economics*. 15. Aufl. McGraw-Hill: New York etc.

Scheck, F. (2013). *Theoretische Physik 2. Nichtrelativistische Quantentheorie*. 3. Aufl. Springer Spektrum: Berlin Heidelberg.

Schönwiese, C.-D. (2008). *Klimatologie*. 3. Aufl. Eugen Ulmer: Stuttgart.

Sen, A. K. und M. S. Srivastava (1990). *Regression Analysis: Theory, Methods and Applications*. Springer Texts in Statistics. Springer: Berlin Heidelberg. DOI: 10.1007/978-3-662-25092-1.

Shumway, R. H. und D. S. Stoffer (2017). *Time Series Analysis and Its Applications*. 4. Aufl. Springer: Cham. DOI: 10.1007/978-3-319-52452-8.

Silver, D. et al. (2017). *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. http://arxiv.org/abs/1712.01815.

Spiess, A.-N. und N. Neumeyer (2010). „An evaluation of R^2 as an inadequate measure for nonlinear models in pharmacological and biochemical research: a Monte Carlo approach". In: *BMC Pharmacology* 10(6). ISSN: 1471-2210. DOI: 10.1186/1471-2210-10-6.

Subasi, A. (2020). *Practical Machine Learning for Data Analysis Using Python*. Academic Press Elsevier: London. ISBN: 9780128213803.

Tabachnick, B. G. und L. S. Fidell (2018). *Using Multivariate Statistics*. 7. Aufl. Pearson Education: Harlow. ISBN: 9780134790541.

Tollefson, J. (2020). „How hot will Earth get by 2100?" In: *Nature* 580, S. 443–445. DOI: https://doi.org/10.1038/d41586-020-01125-x.

— (2021). „Fünf Wege in wärmere Welten". In: *Spektrum der Wissenschaft* 2, S. 48–53. DOI: https://spektrum.de/artikel/1807463.

Unsöld, A. und B. Baschek (1999). *Der neue Kosmos. Einführung in die Astronomie und Astrophysik*. 6. Aufl. Springer Verlag: Berlin Heidelberg New York.

VanderPlas, J. (2018). *Data Science mit Python*. 1. Aufl. mitp: Frechen.

Vogel, J. (2015). *Prognose von Zeitreihen. Eine Einführung für Wirtschaftswissenschaftler*. Springer Gabler: Wiesbaden. DOI: 10.1007/978-3-658-06837-0.

von Weizsäcker, C. F. (1985). *Aufbau der Physik*. Carl Hanser Verlag: München Wien.

Weigend, M. (2019). *Python 3. Das umfassende Praxisbuch*. 8. Aufl. mitp: Frechen.

Wermuth, N. und R. Streit (2007). *Einführung in statistische Analysen. Fragen beantworten mit Hilfe von Daten*. Springer-Verlag: Berlin Heidelberg. DOI: 10.1007/978-3-540-33931-1.

WMO, Hrsg. (2021). *WMO Atlas of Mortality and Economic Losses from Weather, Climate and Water Extremes (1970–2019)*. WMO-No. 1267. World Meteorological Organization: Genève. ISBN: 978-92-63-11267-5. URL: https://library.wmo.int/doc_num.php?explnum_id=10769.

Zeh, H. D. (2012). *Physik ohne Realität: Tiefsinn oder Wahnsinn?* Springer: Berlin Heidelberg. DOI: 10.1007/978-3-642-21890-3_5.

Internetquellen

[MP] https://matplotlib.org/stable/tutorials/ – Matplotlib tutorial

[NumPy] https://realpython.com/numpy-tutorial/ NumPy tutorial

[Py] https://docs.python.org – Python documentation

[PyK] https://www.python-kurs.eu/ – Python tutorial (German)

[PyW] https://www.w3schools.com/python/ – Python tutorial on w3schools

[SciPy] https://scipy-lectures.org/ – SciPy lecture notes

[SKL] https://scikit-learn.org – free software machine learning library for Python

Index

`*`, 46, 50
`+`, 46
`-`, 46
`/`, 46
`MLEResults`, 124
`PolynomialFeautures`, 64
`%`, 46
`break`, 48
`coef_`, 81
`elif`, 47
`forecast`, 125
`intercept_`, 81
`lambda`, 50
`predict`, 90, 124
`quantile`, 77
`ravel`, 53
`score`, 90

A-Posteriori-Wahrscheinlichkeit, 10
A-Priori-Wahrscheinlichkeit, 10
abduction, 18
abgeschlossene Theorie, 21
abhängige Variable, 36
Abstand, 60
Abtastrate, 96
ACF, 114
ADF-Test, 121
anonyme Funktion, 50
Argument, 50
ARIMA-Prozess, 119
ARMA-Prozess, 113
Ausgleichungsrechnung, 71
Autokorrelationsfunktion, 114

Backshift-Operator, 114
Bartlett-Formel, 117
Bayes'sches Informationskriterium, 62
Bayes-Netz, 14
bedingt unabhängig, 13
bedingte Wahrscheinlichkeit, 9
Beobachtung, 12
Bestimmtheitsmaß, 61
bestärkendes Lernen, 33
BIC, 42, 62
Bigramm, 9
biplot, 85
bivariate Wahrscheinlichkeit, 9
Box-Cox-Transformation, 126
break, 48

charakteristisches Polynom, 105
coef_, 81
curve_fit, 71

Data Mining, 32
Data Science, 32
DataFrame, 52
Deduktion, 18
deduktives Modell, 23
Default-Werte von Funktionen, 50
deterministischen Modell, 27
Devianz, 81
Diagnose, 12
differenzenstationärer Prozess, 120
Differenzieren eines Prozesses, 119
Digraph, 14
Durbin-Levinson-Rekursion, 115

Einheitsdevianz, 81
elif, 47
empirische Autokorrelationsfunktion, 115
entpacken, Parameterliste –, 50
Estimator, 54
exogene Variable, 36

f-String, 45
fill_between, 78
forecasting, 124
Funktion, 49

Gauß'sches Modell, 62
GCM, 25
gemeinsame Wahrscheinlichkeit, 9
generalisiertes lineares Modell, 81
GLM, 81

Holt-Winters-Verfahren, 122
Hypothese, 18

in-sample prediction, 124
Induktion, 18
Innovation, 100
Integration eines Prozesses, 119
integrierter Prozess, 120
intercept_, 81
Intervallskala, 35
invertierbarer MA(q)-Prozess, 111
IPCC, 24

Jupyter Notebook, 44

kategoriales Skalenniveau, 34
kausaler linearer Prozess, 100

Kepler, Johannes (1571–1630), 74
Klassifikation, 41
Klimamodell, 24
Konfidenzintervall, 77
Konfidenzniveau, 77
kopernikanische Revolution, 21
Korrelogramm, 116
KPSS-Test, 121

Lambda-Ausdruck, 50
least square method, 60
Lernarten, 33
Lernen, Mechanisierung des –, 32
Library (Python), 50
Likelihood, 12
linearer Prozess, 99
Linkfunktion, 81
list comprehension, 48
loading (principal component), 84
log-log-Skala, 76

MA(q)-Prozess, 110
Markteffizienz, 101
maschinelles Lernen, 32, 41
mathematisches Modell, 24
Matrix, 53
mehrdimensionale Regression, 64
Messung, 12, 21
metrisches Skalenniveau, 34
MLEResults, 124
model, 98
Modell, 23, 24, 36, 40, 41
Modul (Python), 50
Modus, 34
mpl_toolkits.mplot3d, 66
multivariate Regression, 64
multivariate Zeitreihe, 96

nichtlineare Regression, 68
Nominalskala, 34
Nullhypothese, 30

Ockhams Rasiermesser, 27, 42
Ordinalskala, 34
out-of-sample forecasting, 124
Overfitting, 41, 71

p-Wert, 30
PACF, 115
Package (Python), 50
Pandas Series, 123
Parameter, 50
parameter-lineares Regressionsmodell, 61
Parameterliste entpacken, 50
partielle Autokorrelationsfunktion, 115
Peirce, Charles Sanders (1839–1914), 18
Penalty-Funktion, 81
Pipeline, 90
polynomielle Regression, 71
predict, 90, 124
Prediktor, 36
Prognose, 124
Prognosemodell, 90
Prozess
    kausaler linearer –, 100
    linearer –, 99
    stationärer –, 99

Quantentheorie, 20
Quantil, 78

R^2, 61
Random-Walk-Hypothese, 101
randint, 129
random function, 129
Random Walk, 99, 108, 120
Randwahrscheinlichkeit, 9
Rauschen, 96
ravel, 53
Realität, 19, 20
Regression, 41
Regression, nichtlineare –, 68
Regressionsmodell, 59
RegSSR, 61
Regularisierungsfunktion, 81
Residuen, 60
Residuum, 72
Revolution, 21
RPC, 25
RSS, 60

sampling rate, 96
SARIMAX, 122
scatterplot matrix, 89
Schock, 100
scipy.optimize, 64
score, 90
score (PCA), 84
score vector, 84
scree plot, 85
Series, 52
Skala, 34
skalare Zeitreihe, 96
Splat-Operator $*$, 50
SSR, 72
starred expression $*$, 50
stationärer Prozess, 99
statistisches Modell, 36, 40
Stichprobe, 77
stochastischer Prozess, 98
Straffunktion, 81
SVR, 71
Szenario, 25

Technische Analyse, 102
Tensor, 53
TSS, 61

unabhängige Variable, 36
unabhängige Zufallsvariable, 13

Underfitting, 41
Unit-Root-Tests, 121
univariate Zeitreihe, 96
unüberwachtes Lernen, 33
Ursache und Wirkung, 11

Verbundwahrscheinlichkeit, 9
Verhältnisskala, 35
voreingestellte Werte von Funktionen, 50

Wahrscheinlichkeit, 8
Wahrscheinlichkeitsverhältnis, 29
weißes Rauschen, 99
Weltklimarat, 24
Wienerprozess, 101
Wirkung, Ursache und –, 11
Wold Zerlegung, 100
Wold-Entwicklung, 100

Zerlegungssatz von Wold, 100
Zufallsvariable, 7, 98
    bedingt unabhängige –, 13

überwachtes Lernen, 33