

Aufgabe 1: Gleitkommazahlen

5 P.

Betrachten Sie ein Computersystem, auf dem einfach-genaue Gleitkommazahlen dargestellt werden als $(1 - 2b_s) \cdot m \cdot 2^e$ mit folgenden Eigenschaften:

- $b_s = 0$ oder $b_s = 1$
- 23 Binärstellen für die Mantisse m
- Die Gleitkommazahl ist normalisiert, d. h. $1 \leq m < 2$
- $-128 \leq e \leq 127$.

Welche der folgenden Zahlen lassen sich damit exakt darstellen? Welche näherungsweise?

- a) $1 = (1-2 \cdot 0) \cdot 1 \cdot 2^0$ $* 1 + 2^{-23} \cdot (0,6 \cdot 2^{23})$
- b) $18 = (1-2 \cdot 0) \cdot 1 \cdot 2^4 \cdot 2^3$
- c) $0,5 = (1-2 \cdot 0) \cdot 1 \cdot 2^{-1}$ $0,0999999999999999$ keine Näherung
- d) $0,1 = (1-2 \cdot 0) \cdot * \cdot 2^{-4}$ keine Näherung exakt
- e) $\frac{1}{3} = (1-2 \cdot 0) \cdot 2^{-2} \cdot 0,3333333333333333$ keine Näherung $2 \cdot 2^{-23} / (1-2 \cdot 0)$
- f) $10^{40} = (1-2 \cdot 0) \cdot 2^{132}$, keine Näherung

Aufgabe 2: Chi-Quadrat

- Erzeugen Sie mit der Funktion `numpy.random.chisquare` 1000 Zufallszahlen aus einer Chi-Quadrat-Verteilung mit 5 Freiheitsgraden.
- Erstellen Sie mit den zuvor erzeugten Zufallszahlen ein eindimensionales Histogramm mit Fehlerbalken (Die Fehler pro Bin sollen $\sqrt{N_i}$ mit N_i Einträgen pro Bin i sein).
- Führen Sie mit `scipy.stats.chi2.fit` einen Maximum Likelihood Fit an das in a) gezogenen Sample durch und vergleichen Sie die Parameter des Fits und der wahren Verteilung.
- Stellen Sie in dem Histogramm ebenfalls sowohl die gefittete, als auch die wahre Chi-Quadrat-Verteilung dar.

```
19
20     def sample(self, n=1):
21         """
22             n : int
23                 Anzahl der Datenpunkte die gesampled werden sollen.
24         """
25         # TODO: Implement
26         return sample # 1D array mit Laenge n
```

Implementieren Sie die Methoden `__init__`, `__call__`, `sample`. Eine kurze (und ausreichende) Einführung zu KDEs gibt es bei [de.wikipedia.org/wiki/Kerndichteschätzer](https://de.wikipedia.org/wiki/Kerndichtesch%C3%A4tzer).

Zeit	Raum	Abgabe im Moodle; Mails mit Betreff: [SMD1718]
Di. 10-12	CP-03-150	philipp2.hoffmann@udo.edu und jan.soedingrekso@udo.edu
Di. 16-18	P1-02-110	felix.neubuerger@udo.edu und tobias.hoinka@udo.edu
Di. 16-18	CP-03-150	simone.mender@udo.edu und maximilian.meier@udo.edu

Aufgabe 1: *Polynom*

3 P.

Werten Sie das Polynom $f(x) = (1-x)^6$ numerisch auf unterschiedliche Weise im Bereich $0,999 \leq x \leq 1,001$ an 1000 Stellen aus und stellen Sie das Ergebnis grafisch dar.

- Berechnung von $(1-x)^6$.
- Berechnung des ausmultiplizierten Polynoms (binomische Formel) auf naive Weise.
- Berechnung des ausmultiplizierten Polynoms mittels Horner-Schema.

Interpretieren Sie die Ergebnisse für 16-Bit, 32-Bit und 64-Bit Gleitkommazahlen.

Hinweis: Erstellen sie das Array für die x -Werte mit 16-Bit Präzision mit

```
np.linspace(a, b, N, dtype='float16').
```

Analog für 32-Bit und 64-Bit.

Aufgabe 2: *Grenzwert*

4 P.

- Berechnen Sie den Grenzwert

$$\lim_{x \rightarrow 0} (\sqrt{9-x} - 3)/x$$

- Versuchen Sie, obigen Grenzwert numerisch zu berechnen, indem Sie für x nacheinander $0,1, 0,01, 0,001, \dots$ bis 10^{-20} einsetzen. Interpretieren Sie das Ergebnis.

Aufgabe 3: *Numerische Stabilität*

3 P.

Betrachten Sie die Funktionen

- $f(x) = (x^3 + 1/3) - (x^3 - 1/3)$ und
- $g(x) = ((3 + x^3/3) - (3 - x^3/3))/x^3$.

Bestimmen Sie empirisch, für welche Bereiche von x (grob) das numerische Ergebnis

- vom algebraischen um nicht mehr als 1 % abweicht,

- gleich Null ist.
- c) Stellen Sie das Ergebnis in geeigneter Form graphisch dar (d. h. z. B. logarithmische x -Skala)!

Aufgabe 4: $e^- e^+ \rightarrow \gamma\gamma$

10 P.

Ein Term des differentiellen Wirkungsquerschnitts für die Reaktion $e^- e^+ \rightarrow \gamma\gamma$ ist gegeben durch

$$\frac{d\sigma}{d\Omega} = \frac{\alpha^2}{s} \left(\frac{2 + \sin^2(\theta)}{1 - \beta^2 \cos^2(\theta)} \right) .$$

mit

$$s = (2E_e)^2 \quad (E_e \text{ ist die Energie des } e^- \text{ oder } e^+ \text{ im Schwerpunktsystem}),$$

$$\beta = \sqrt{1 - \gamma^{-2}}$$

$$\gamma = \frac{E_e}{m_e} \quad (m_e = 511 \text{ keV})$$

und der Feinstrukturkonstante α .

- a) Ist diese Gleichung für $\frac{d\sigma}{d\Omega}$ numerisch stabil? In welchem Bereich von θ ist die Gleichung für $E_e = 50 \text{ GeV}$ numerisch instabil?
- b) Beheben Sie die Stabilitätsprobleme durch eine geeignete analytische Umformung. (Hinweis: Nutzen Sie $1 - \beta^2 = 1/\gamma^2$ und $1 = \sin^2(\theta) + \cos^2(\theta)$)
- c) Zeigen Sie, dass Sie die Stabilitätsprobleme behoben haben, indem Sie beide Gleichungen in den kritischen Intervallen darstellen.
- d) Berechnen Sie die Konditionszahl. Wie hängt diese von θ ab?
- e) Stellen Sie den Verlauf der Konditionszahl als Funktion von θ ($0 \leq \theta \leq \pi$) grafisch dar. In welchem Bereich ist das Problem gut, in welchem schlecht konditioniert?

Aufgabe 1

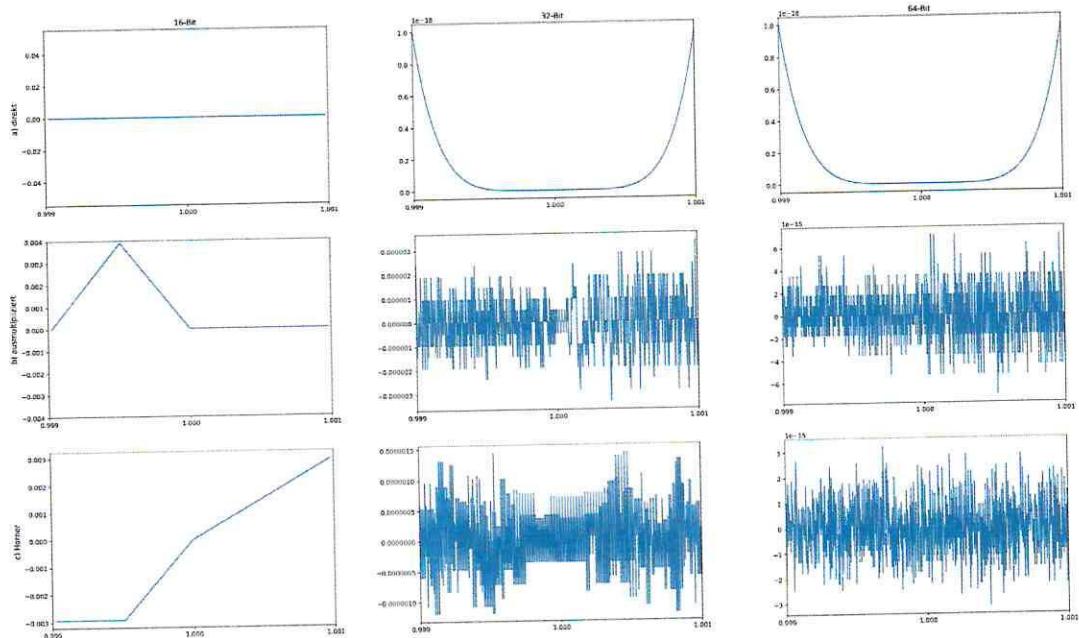


Abbildung 1: Ergebnisse zu Aufgabe 1.

Unsere Ergebnisse zu Aufgabe 1 finden sich in Abbildung 1. Wir gehen davon aus, dass ihr unsere Abgaben zum Korrigieren nicht ausdrückt, darum sollte die Darstellung so mit zoomen gut funktionieren und übersichtlicher sein, als alles einzeln darzustellen. In der ersten Zeile ist das Ergebnis von $(1 - x)^6$ in unveränderter Form dargestellt, in der zweiten Zeile die ausmultiplizierte Form und in der dritten Zeile wird mittels Horner'sches Schema ausgewertet. Die Spalten von links nach rechts verwenden 16-Bit, 32-Bit und 64-Bit Gleitkommazahlen.

Doch ↗

Allgemein ist zu sagen, dass wie erwartet mit erhöhter Genauigkeit der Gleitkommazahlen auch die Genauigkeit des Ergebnisses zunimmt. Eine Ausnahme ist mit der faktorisierten Form aus a) gegeben, denn hier liefern bereits 32-Bit Gleitkommazahlen das exakte Ergebnis. Die faktorisierte Form ist mit Abstand die beste, da sich die Fehler hier nicht akkumulieren und insgesamt weniger Rechenoperationen durchgeführt werden müssen. Bei den beiden anderen Formen müssen deutlich mehr Rechenoperationen durchgeführt werden wobei sich die Fehler fortpflanzen.

Gibt einen Unterschied zw.
 Horner und Ausmultipliziert

✓

3/3

Aufgabe 2

a)

Der analytische Grenzwert ergibt mit Hilfe des Satz von L'HOSPITAL sich wie folgt:

$$\lim_{x \rightarrow 0} \frac{\sqrt{9-x} - 3}{x} = \lim_{x \rightarrow 0} \frac{-\frac{1}{2}(9-x)^{-\frac{1}{2}}}{1} = -\frac{1}{6}. \quad (1)$$

b)

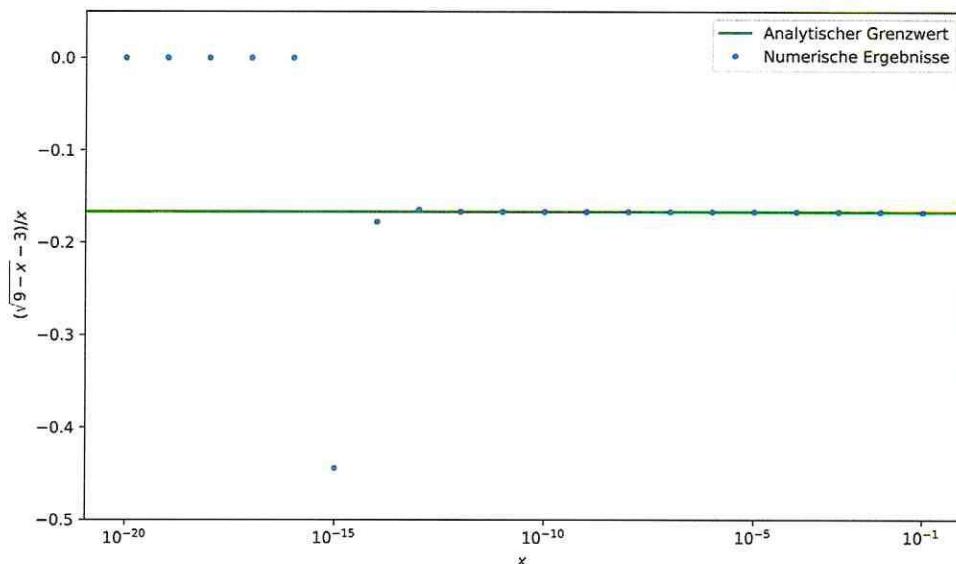


Abbildung 2: Vergleich des analytischen Grenzwertes mit dem numerischen Ergebnis.

Der verwendete PYTHON Datentyp *float* verwendet standardmäßig 53 Bit Präzision und damit wird eine Rechnergenauigkeit von 10^{-15} erreicht. Wie in Abbildung 2 gut zu erkennen ist, führt dies dazu, dass bei der Differenzbildung im Fall von $x < 10^{-15}$ das Ergebnis gezwungener Maßen auf exakt 9 gerundet wird, und der Zähler somit 0 ergibt. Anschließend wird durch $0 < x < 10^{-15}$ geteilt was zum Gesamtergebnis 0 führt. Für $x > 10^{-15}$ tritt dieses Rundungsproblem noch nicht auf, und das Ergebnis liegt sehr nahe am analytischen Grenzwert, da der Grenzwert schnell konvergiert.

✓ +4

Aufgabe 3

Es gilt

$$f(x) = g(x) = \frac{2}{3}, \quad \forall x. \quad (2)$$

empirisch!

Bis zur Stelle $x \approx 10^4$ nimmt f den analytisch korrekten Wert an und ist für $x > 10^5$ null. In der Darstellung g ist das Verhalten umgekehrt: Hier ist das numerische Ergebnis

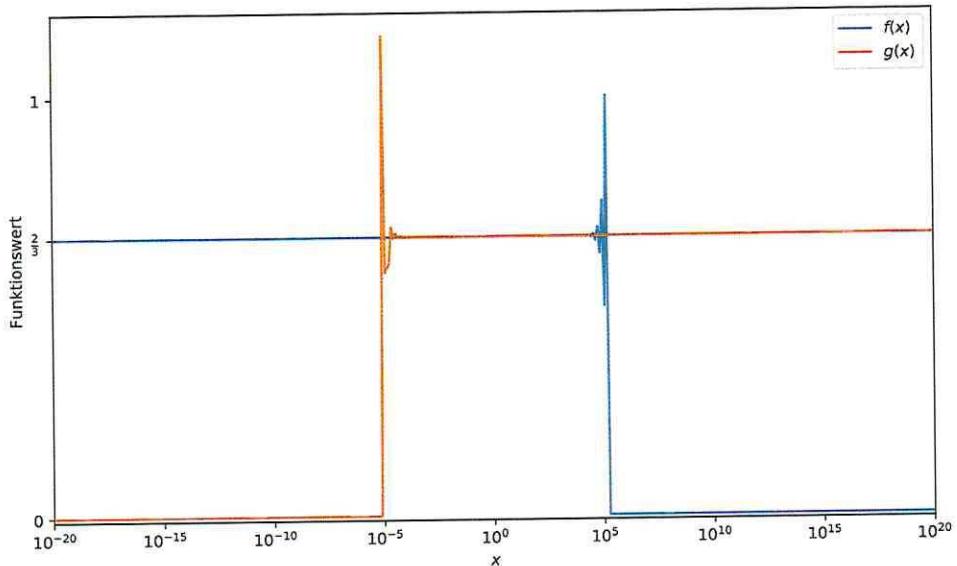


Abbildung 3: Vergleich der numerischen Ergebnisse.

so.
für etwa $x < 10^{-5}$ gleich Null und ab ungefähr $x = 10^{-4}$ gleich dem analytischen Wert.
Das Verhalten für f lässt sich dadurch erklären, dass für große x am Rande der Rechnergenauigkeit die Terme $\pm \frac{1}{3}$ verloren gehen, sodass sich $x^3 - x^3 = 0$ ergibt. Im Fall von g tritt das umgekehrte Problem auf, denn hier wird für sehr kleine x $\frac{x^3}{3}$ zu klein gegenüber 3 um bei der gegebenen Rechnergenauigkeit dargestellt werden zu können. Es ergibt sich so $3 - 3 = 0$. *✓3*

Aufgabe 4

$$\frac{d\sigma}{d\Omega} = \frac{\alpha^2}{s} \left(\frac{2 + \sin^2(\theta)}{1 - \beta^2 \cos^2(\theta)} \right) =: f(\theta) \quad (3)$$

a)

Die Funktion ist numerisch instabil, da $\beta \approx 1$ gilt und damit

$$1 - \beta^2 \cos^2(n \cdot \pi) \approx 0 \quad \forall n \in \mathbb{Z} \quad (4)$$

gilt. An den Stellen, an denen θ ein ganzzahliges Vielfaches von π ist, wird dann durch fast Null geteilt, was schnell zu Rundungsfehlern führt. *✓*

b)

Die folgende Umformung sorgt dafür, dass nicht mehr durch fast Null geteilt wird:

$$\frac{d\sigma}{d\Omega} = \frac{\alpha^2}{s} \left(\frac{2 + \sin^2(\theta)}{1 - \beta^2 \cos^2(\theta)} \right) =: f(\theta) \quad (5)$$

$$= \frac{\alpha^2}{s} \left(\frac{2 + \sin^2(\theta)}{\sin^2(\theta) + \cos^2(\theta) - \beta^2 \cos^2(\theta)} \right) \quad (6)$$

$$= \frac{\alpha^2}{s} \left(\frac{2 + \sin^2(\theta)}{\sin^2(\theta) + (1 - \beta^2) \cos^2(\theta)} \right) \quad (7)$$

$$= \frac{\alpha^2}{s} \left(\frac{2 + \sin^2(\theta)}{\sin^2(\theta) + \frac{1}{\gamma^2} \cos^2(\theta)} \right) \quad (8)$$

$$= \frac{\alpha^2}{s} \gamma^2 \left(\frac{2 + \sin^2(\theta)}{\gamma^2 \sin^2(\theta) + \cos^2(\theta)} \right) \quad (9)$$

c)

✓

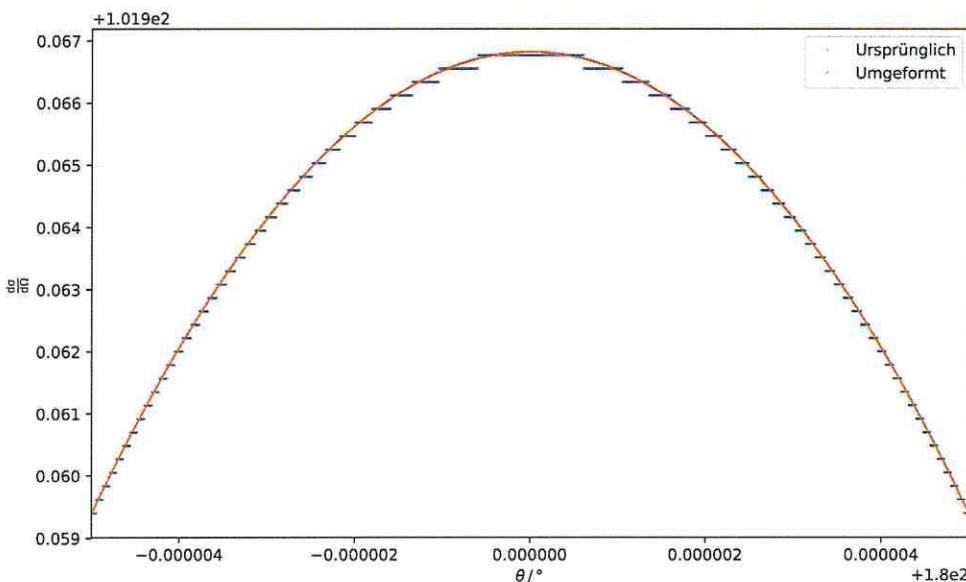


Abbildung 4: Vergleich der numerisch stabilen und instabilen Form.

✓

Die Unterschiede zwischen den beiden Formen (3) und (9) sind in Abbildung 4 gut zu erkennen. Dabei wurden die ursprüngliche Funktion in blau und die umgeformte Funktion in orange gezeichnet in dem Bereich um $\theta = 180^\circ$ geplottet. Es ist gut zu erkennen, dass die ursprüngliche Form Plateaus bildet. Sie liefert also für mehrere Werte von θ den gleichen Wert und springt dann am Ende des Plateaus plötzlich auf einen anderen Wert. Dieses Verhalten ist bei der umgeformten Funktion nicht mehr zu beobachten.

d)

$$\begin{aligned}
 f'(\theta) &= \frac{-2 \sin(\theta) \cos(\theta) (1 + \beta^2)}{(1 - \beta^2 \cos^2(\theta))^2} \\
 \Rightarrow K &= \left| \theta \frac{f'(\theta)}{f(\theta)} \right| \\
 &= \left| \theta \frac{-2 \sin(\theta) \cos(\theta) (1 + \beta^2) (1 - \beta^2 \cos^2(\theta))}{(1 - \beta^2 \cos^2(\theta))^2 (2 + \sin^2(\theta))} \right| \\
 &= \left| \theta \frac{-2 \sin(\theta) \cos(\theta) (1 + \beta^2)}{(1 - \beta^2 \cos^2(\theta)) (2 + \sin^2(\theta))} \right|
 \end{aligned} \tag{10}$$

e)

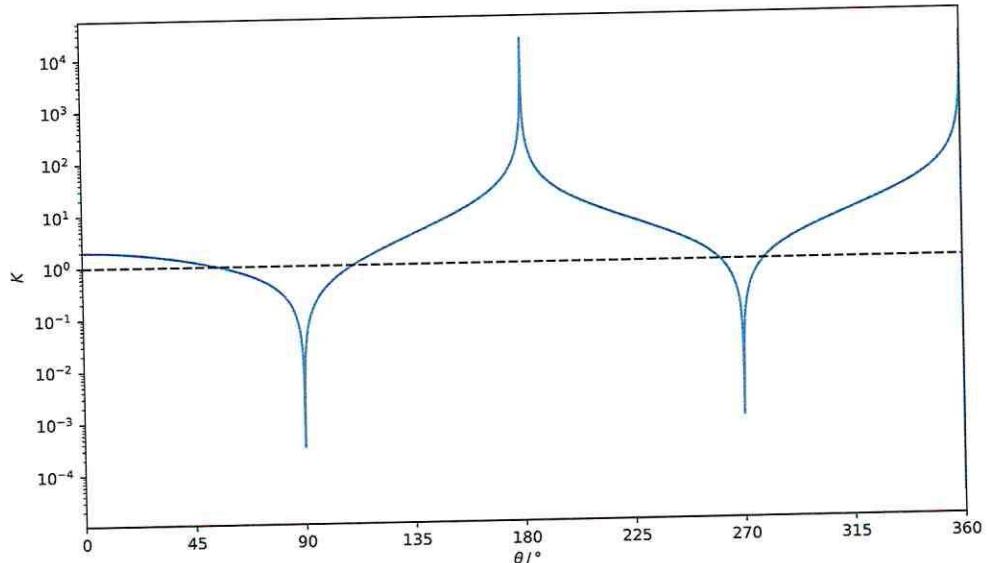


Abbildung 5: Grafische Darstellung des Verlaufs der Konditionszahl.

Ein Problem hat für $K < 1$ eine Fehlerdämpfung und für $K > 1$ eine Fehlerverstärkung. Die Bereiche sind in Abbildung 5 durch die gestrichelte Linie abgetrennt. Auffällig ist, dass das Problem bei $\theta = \frac{\pi}{2}$ besonders gut, und bei $\theta = \pi$ besonders schlecht konditioniert ist.

✓
10/10

Code für blatt00

Burkowitz, Bange, Harnisch

26. Oktober 2017

```
..../blatt00/Burkowitz_Bange_Harnisch/aufgl.py

1 import matplotlib.pyplot as plt
2 import numpy as np
3
4
5 # _____ Funktionen _____
6 def horner(x, a):
7     """Hornerschema für Polynom mit Koeffizientenvektor a"""
8     f = a[0]
9     # Lässt sich diese Schleife irgendwie schlau durch numpy Befehle ersetzen?
10    for i in range(1, len(a)):
11        f = f*x+a[i]
12    return f
13
14
15 def f1(x):
16     """Das Polynom in faktorisierte Form"""
17     return (1-x)**6
18
19
20 def f2(x):
21     """Das Polynom in ausmultiplizierter Form"""
22     return x**6 - 6*x**5 + 15*x**4 - 20*x**3 + 15*x**2 - 6*x + 1
23
24
25 def aufgl():
26     x16 = np.linspace(0.999, 1.001, 1000, dtype='float16')
27     x32 = np.linspace(0.999, 1.001, 1000, dtype='float32')
28     x64 = np.linspace(0.999, 1.001, 1000, dtype='float64')
29
30     # Koeffizientenvektor für Horner
31     a = ([1, -6, 15, -20, 15, -6, 1])
32
33     f, axarr = plt.subplots(3, 3) # , sharex='col', sharey='row'
34     axarr[0][0].plot(x16, f1(x16))
35     axarr[0][1].plot(x32, f1(x32))
36     axarr[0][2].plot(x64, f1(x64))
37     axarr[1][0].plot(x16, f2(x16))
38     axarr[1][1].plot(x32, f2(x32))
39     axarr[1][2].plot(x64, f2(x64))
40     axarr[2][0].plot(x16, horner(x16, a))
41     axarr[2][1].plot(x32, horner(x32, a))
42     axarr[2][2].plot(x64, horner(x64, a))
43
44     f.tight_layout()
45     f.set_figwidth(25)
46     f.set_figheight(15)
47     plt.setp(axarr, xlim=[0.999, 1.001], xticks=[0.999, 1.000, 1.001])
48     # , ylim=[-0.5e-18, 1.5e-18]
49     f.subplots_adjust(wspace=0.3)
50     f.subplots_adjust(hspace=0.2)
51     axarr[1][0].set_ylim([-0.004, 0.004])
52     cols = ['16-Bit', '32-Bit', '64-Bit']
53     rows = ['a) direkt', 'b) ausmultipliziert', 'c) Horner']
54
55     for ax, col in zip(axarr[0], cols):
56         ax.set_title(col)
```

```

28     plt.legend()
29     plt.ylim(-0.01, 1.2)
30     plt.xlim(1e-20, 1e20)
31     plt.xlabel(r"$x$")
32     plt.ylabel("Funktionswert")
33     # plt.show()
34     plt.savefig("A3.pdf")
35
36
37 # ----- Main -----
38 if __name__ == '__main__':
39     aufg3()


```

..../blatt00/Burkowitz_Bange_Harnisch/aufg4.py

```

1 import matplotlib.pyplot as plt
2 from scipy.constants import alpha
3 import numpy as np
4 from pylab import rcParams
5 # from scipy.misc import derivative as diff
6
7 rcParams['figure.figsize'] = 10, 5.8
8 rcParams['legend.numpoints'] = 1
9
10 # ----- Konstanten -----
11 E_e = 50
12 s = (2*E_e)**2
13 gamma = E_e/511e-6
14 beta = np.sqrt(1 - gamma**(-2))
15
16
17 # ----- Funktionen -----
18 def dwq(theta):
19     """Differentieller Wirkungsquerschnitt in ursprünglicher Form"""
20     return alpha**2*s*(2 + np.sin(theta)**2)/(1 - beta**2*np.cos(theta)**2)
21
22
23 def dwq2(theta):
24     """Differentieller Wirkungsquerschnitt in numerisch stabiler Form"""
25     return gamma**2*alpha**2*s*(2 + np.sin(theta)**2) \
26         / (gamma**2*np.sin(theta)**2 + np.cos(theta)**2)
27
28
29 # def k(x):
30     """Numerisch berechnete Konditionszahl"""
31     # return abs(x*diff(dwq2, x, dx=1e-8)/dwq2(x))
32
33
34 def K(x):
35     """Analytisch berechnete Konditionszahl"""
36     return abs(x**2*np.sin(x)*np.cos(x)*(1 + beta**2) \
37         / ((1 - beta**2*np.cos(x)**2)*(2 + np.sin(x)**2)))
38
39
40 def aufg4():
41     # theta = np.arange(0.5, 361, 1)
42     theta = np.linspace(179.999995, 180.000005, 1e4)
43     plt.plot(theta, dwq(np.deg2rad(theta)), ".", ms=0.5, label="Ursprünglich")
44     plt.plot(theta, dwq2(np.deg2rad(theta)), "x", ms=0.5, label="Umgeformt")
45     plt.xlim(179.999995, 180.000005)
46     plt.xlabel(r"$\theta$")
47     plt.ylabel(r"$\frac{d}{d\theta} \sigma(\Omega)$")
48     plt.legend()
49     # plt.show()
50     plt.savefig("A4_stab.pdf")
51     plt.clf()
52
53     theta = np.linspace(0, 360, 1e4)
54     plt.plot(theta, K(theta/180*np.pi))
55     # plt.plot(theta, k(theta/180*np.pi))
56     plt.xticks([0, 45, 90, 135, 180, 225, 270, 315, 360])

```

```
57     plt.xlim(0, 360)
58     plt.yscale("log")
59     plt.axhline(1, ls="--", color="k")
60     plt.xlabel(r"$\theta_t / {}^\circ$")
61     plt.ylabel(r"$KS$")
62     # plt.show()
63     plt.savefig("A4_kond.pdf")
64
65
66 # _____ Main _____
67 if __name__ == '__main__':
68     aufg4()
```

Zeit	Raum	Abgabe im Moodle; Mails mit Betreff: [SMD1718]
Di. 10-12	CP-03-150	philipp2.hoffmann@udo.edu und jan.soedingrekso@udo.edu
Di. 16-18	P1-02-110	felix.neubuerger@udo.edu und tobias.hoinka@udo.edu
Di. 16-18	CP-03-150	simone.mender@udo.edu und maximilian.meier@udo.edu

Aufgabe 5: *Maxwell'sche Geschwindigkeitsverteilung* 6 P.

Die Wahrscheinlichkeitsdichte des Betrags der Geschwindigkeit v der Moleküle in einem idealen Gas bei der absoluten Temperatur T ist

$$f(v) = N \cdot \exp\left(-\frac{mv^2}{2k_B T}\right) \cdot 4\pi v^2,$$

dabei ist m die Molekulmasse, k_B die Boltzmannkonstante und N die Normalisierungskonstante.

Zur Beantwortung der Fragen müssen Sie zuerst N bestimmen! Drücken Sie die Ergebnisse als Funktion von m und T aus, sowie die anderen Ergebnisse als Funktion von v_m .

Hinweis zu c): Eine analytische Lösung ist hier nicht möglich, benutzen Sie ein numerisches Verfahren.

Wie groß sind

- a) die wahrscheinlichste Geschwindigkeit v_m ,
- b) der Mittelwert der Geschwindigkeit $\langle v \rangle$,
- c) der Median der Geschwindigkeit $v_{0,5}$,
- d) die volle Breite auf halber Höhe der Verteilung (v_{FWHM}) und
- e) die Standardabweichung der Geschwindigkeit σ_v .

Aufgabe 6: *Würfel* 6 P.

Nutzen Sie für diese Aufgabe die Schreibweise für Wahrscheinlichkeiten aus der Vorlesung (z.B. $P(W_{\text{rot}} + W_{\text{blau}} = 42) = \dots$, $P(W_{\text{rot}} + W_{\text{blau}} = 42 | W_{\text{rot}} = 4) = \dots$). Sie würfeln mit zwei Würfeln, einem roten und einem blauen. Wie groß ist die Wahrscheinlichkeit dafür, dass

- a) die Summe der Punkte 9 ergibt,
- b) die Summe der Punkte 9 oder mehr ergibt,

- c) ein Würfel 4, der andere 5 Punkte zeigt,
- d) der rote Würfel 4, der blaue 5 Punkte zeigt?

Sie werfen die Würfel so, dass der blaue Würfel hinter einen Gegenstand rollt, so dass Sie ihn zunächst nicht sehen können. Der rote Würfel zeigt eine 4. Nachdem Sie das gesehen haben, wie groß ist dann die Wahrscheinlichkeit dafür, dass

- e) die Summe der Punkte 9 ergibt,
- f) die Summe der Punkte 9 oder mehr ergibt,
- g) der rote Würfel 4, der blaue 5 Punkte zeigt?

Aufgabe 7: Zweidimensionale Gaußverteilung

8 P.

Eine zweidimensionale Gaußverteilung sei durch folgende Parameter gekennzeichnet:

$$\mu_x = 4, \quad \mu_y = 2, \quad \sigma_x = 3,5, \quad \sigma_y = 1,5 \quad \text{und} \quad \text{Cov}(x, y) = 4,2$$

- a) Wie groß ist der Korrelationskoeffizient?
- b) Zeigen Sie, dass die Kurven konstanter Wahrscheinlichkeitsdichte Ellipsen sind.
- c) Zeichnen Sie die Ellipse, bei der $f(x, y)$ auf das $1/\sqrt{e}$ -fache des Maximums abgefallen ist. Zeichnen Sie die Werte μ_x , μ_y , $\mu_x \pm \sigma_x$ und $\mu_y \pm \sigma_y$ in Ihrer Zeichnung ein.
- d) Geben Sie eine Rotationsmatrix \mathbf{M} an, so dass die Variablen $(x', y')^\top = \mathbf{M}(x, y)^\top$ unkorreliert sind. Wie groß sind $\sigma_{x'}$ und $\sigma_{y'}$? Zeichnen Sie diese Werte in die Zeichnung ein.
- e) Wie lang sind die Hauptachsen der Ellipse und welchen Winkel bilden sie mit den Koordinatenachsen? Zeichnen Sie diese Werte in die Zeichnung ein.
- f) Wie lauten die bedingten Wahrscheinlichkeitsdichten $f(x | y)$ und $f(y | x)$? Zeichnen Sie diese Werte in die Zeichnung ein.
- g) Wo liegen die bedingten Mittelwerte $E(x | y)$ und $E(y | x)$? Zeichnen Sie diese Werte in die Zeichnung ein.

Aufgabe 5

In dieser Aufgabe wird die MAXWELL'sche Geschwindigkeitsverteilung in der Form

$$f(v) = N 4\pi v^2 \exp\left(-\frac{mv^2}{2k_B T}\right) \quad (1)$$

betrachtet. Zuerst ist die Verteilung zu normieren. Dafür kann die Gammafunktion verwendet werden, es geht aber auch viel schöner ohne. Dafür definieren wir zuerst

$$I := \int_0^\infty \exp(-\alpha t x^2) dx. \quad (2)$$

Um I auszuwerten wird quadriert und anschließend in Polarkoordinaten gewechselt:

$$I^2 = \left(\int_0^\infty \exp(-\alpha t x^2) dx \right) \left(\int_0^\infty \exp(-\alpha t x^2) dx \right) \quad (3)$$

$$= \int_0^\infty \int_0^\infty \exp(-\alpha t(x^2 + y^2)) dx dy \quad (4)$$

$$= \int_0^{\pi/2} d\phi \int_0^\infty \exp(-\alpha t r^2) r dr \quad (5)$$

$$= \frac{\pi}{4} \int_0^\infty \exp(-\alpha t u) du \quad (6)$$

$$= \frac{\pi}{4\alpha t}. \quad (7)$$

Wegen $I > 0$ ergibt sich somit

$$I = \sqrt{I^2} = \frac{1}{2} \sqrt{\frac{\pi}{\alpha t}}. \quad (8)$$

Das für die Normierung zu berechnende Integral lässt sich nun auswerten:

$$\int_0^\infty x^2 \exp(-\alpha x^2) = -\frac{1}{\alpha} \frac{d}{dt} I \Big|_{t=1} = \frac{1}{4} \sqrt{\pi} (\alpha t)^{-\frac{3}{2}} \Big|_{t=1} = \frac{\sqrt{\pi}}{4\alpha^{\frac{3}{2}}}. \quad (9)$$

Damit erhalten wir

$$\int_0^\infty f(v) dv = N 4\pi \frac{\sqrt{\pi}}{4} \left(\frac{2k_B T}{m} \right)^{\frac{3}{2}} \stackrel{!}{=} 1 \quad (10)$$

$$\Rightarrow N = \left(\frac{m}{2\pi k_B T} \right)^{\frac{3}{2}}. \quad (11)$$

a)

Die wahrscheinlichste Geschwindigkeit v_m ist die Stelle, an der f maximal wird. Daher bilden wir die Ableitung

$$\frac{d}{dv} f = 4\pi N v \exp\left(-\frac{mv^2}{2k_B T}\right) \left(2 - \frac{mv^2}{k_B T}\right) \stackrel{!}{=} 0. \quad (12)$$

Das Extremum bei 0 ist offenbar ein Tiefpunkt, da f keine negativen Werte annimmt und $f(0) = 0$. Daraus folgt, dass die beiden anderen Extrema Maxima sind, wobei das positive der beiden an der gesuchten Stelle

$$v_m = \sqrt{\frac{2k_B T}{m}} \quad (13)$$

liegt.

b)

Der Erwartungswert des Geschwindigkeitsbetrags lässt sich wie folgt berechnen:

$$\langle v \rangle_f = \int_0^\infty v f(v) dv = 4\pi N \int_0^\infty v^3 \exp\left(-\frac{mv^2}{2k_B T}\right) dv \quad (14)$$

$$= 4\pi N \int_0^\infty \frac{u}{2} \exp\left(-\frac{mu}{2k_B T}\right) du \quad (15)$$

$$= 2\pi N \left(\left[-\frac{2uk_B T}{m} \exp\left(-\frac{mu}{2k_B T}\right) \right]_{u=0}^{u=\infty} + \frac{2k_B T}{m} \int_0^\infty \exp\left(-\frac{mu}{2k_B T}\right) du \right) \quad (16)$$

$$= 2\pi N \left(\frac{k_B T}{m} \right)^2 \stackrel{(11)}{=} 2\sqrt{\frac{2k_B T}{\pi m}} = \frac{2}{\sqrt{\pi}} v_m. \quad (17)$$

c)

Gesucht ist der Median $v_{0,5}$, für den gilt

$$\int_0^{v_{0,5}} f(v) dv = \frac{1}{2}. \quad (18)$$

Es ist nicht möglich, eine analytische Lösung für $v_{0,5}$ zu finden, da für $f(v)$ keine Stammfunktion gefunden werden kann. Daher lösen wir die Gleichung numerisch, dafür muss diese zuerst in eine dimensionslose Form gebracht werden. Mit dem vorherigen Ergebnis (13) gilt

$$N = \pi^{-\frac{3}{2}} v_m^{-3} \Rightarrow f(v) = \frac{4}{\sqrt{\pi} v_m} \frac{v^2}{v_m^2} \exp\left(-\frac{v^2}{v_m^2}\right). \quad (19)$$

Mit der Substitution $x = \frac{v}{v_m}$ nimmt (18) die für numerische Integration geeignete Form

$$\int_0^{v_{0,5}} f(v) dv = \frac{4}{\sqrt{\pi}} \int_0^{\frac{v_{0,5}}{v_m}} x^2 e^{-x^2} dx \stackrel{!}{=} \frac{1}{2} \quad (20)$$

an. Das Integral kann nun sukzessive mit einer beliebigen NEWTON-COTES-Formel berechnet werden, bis es den gewünschten Wert von $\frac{1}{2}$ überschreitet. Wir verwenden die Trapezregel. Um den Performance-Limitierungen des PYTHON-Interpreters möglichst gut zu entgehen, verwenden wir außerdem die Pakete NUMPY bzw. NUMEXPR. Es werden in einer Schleife immer N Funktionsauswertungen parallelisiert durchgeführt und die Teilsummen anschließend aufsummiert. Für optimale Performance sollte N so gewählt werden, dass der Hauptspeicher maximal ausgelastet wird. Außerdem ist die Lösung für umso stabiler, umso größer N gewählt wird, da seltener gerundet werden muss. Prinzipiell haben wir die Berechnung so implementiert, dass beliebige Genauigkeiten erreicht werden können. So haben wir den Median mit einer Genauigkeit von 10^{-10} als

$$v_{0,5} = 1,0876520285v_m \quad (21)$$

bestimmt.

d)

Die volle Breite auf halber Höhe v_{FWHM} ergibt sich als Differenz der Lösungen von

$$f(v) - \frac{1}{2}f(v_m) = x^2 \exp(-x^2) - \frac{1}{2e} \stackrel{!}{=} 0 \quad (22)$$

mit $x = \frac{v}{v_m}$. Das Ergebnis lässt sich zwar unter Verwendung der LAMBERTSCHEN-W-Funktion analytisch darstellen, diese besitzt aber bekanntlich selbst keine elementare Darstellung. Daher lösen wir erneut numerisch. Da (22) einfach analytisch abgeleitet werden kann, bietet sich die Verwendung des NEWTON-Verfahrens an. Die Ableitung lautet

$$\frac{d}{dx} \left(f(v) - \frac{1}{2}f(v_m) \right) = 2x \exp(-x^2)(1 - x^2). \quad (23)$$

Aufgrund der quadratischen Konvergenz des NEWTON-Verfahrens ergeben sich bereits nach wenigen Iterationen mit den Startwerten 0,8 und 1,2 die gesuchten Nullstellen

$$x_1 = 0,481623247971, \quad x_2 = 1,63656560822 \quad (24)$$

$$\Rightarrow v_{FWHM} = v_m(x_2 - x_1) = 1,15494236025v_m. \quad (25)$$

e)

Da wir den Mittelwert der Geschwindigkeit $\langle v \rangle$ bereits in b) bestimmt haben, bietet es sich an die Standardabweichung der Geschwindigkeit σ_v über die Form

$$\sigma_v^2 = \langle v^2 \rangle - \langle v \rangle^2 \quad (26)$$

zu bestimmen, denn damit muss nur noch

$$\langle v^2 \rangle = \int_0^\infty v^2 f(v) dv = 4\pi N \int_0^\infty v^4 \exp\left(-\frac{v^2}{v_m^2}\right) dv \quad (27)$$

ausgewertet werden. Hierzu verwenden wir erneut die Definition (2):

$$\int_0^\infty x^4 \exp(-\alpha x^2) dx = \frac{1}{\alpha^2} \frac{d^2}{dt^2} I \Big|_{t=1} \stackrel{(8)}{=} \frac{3\sqrt{\pi}}{8} \alpha^{-\frac{5}{2}}. \quad (28)$$

Damit ergibt sich

$$\langle v^2 \rangle = 4\pi(\pi v_m^2)^{-\frac{3}{2}} \frac{3\sqrt{\pi}}{8} v_m^5 = \frac{3}{2} v_m^2 \quad (29)$$

$$\Rightarrow \sigma_v = \sqrt{\langle v^2 \rangle - \langle v \rangle^2} = \sqrt{\frac{3}{2} - \frac{4}{\pi}} v_m. \quad (30)$$



Aufgabe 6

In dieser Aufgabe werden die Wahrscheinlichkeiten für ausgewählte Ereignisse, die bei einem Wurf mit einem roten und einem blauen optimalen sechsseitigen Würfel auftreten können, betrachtet.

a) $P(W_{\text{rot}} + W_{\text{blau}} = 9) = \frac{4}{6} \cdot \frac{1}{6} = \frac{1}{9}$



b) $P(W_{\text{rot}} + W_{\text{blau}} \geq 9) = \frac{1}{6} \cdot \frac{4}{6} + \frac{1}{6} \cdot \frac{3}{6} + \frac{1}{6} \cdot \frac{2}{6} + \frac{1}{6} \cdot \frac{1}{6} = \frac{10}{36}$



c) $P((W_{\text{rot}} = 4 \wedge W_{\text{blau}} = 5) \vee (W_{\text{rot}} = 5 \wedge W_{\text{blau}} = 4)) = \frac{2}{6} \cdot \frac{1}{6} = \frac{1}{18}$



d) $P(W_{\text{rot}} = 4 \wedge W_{\text{blau}} = 5) = \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}$



Der rote Würfel zeigt jetzt immer 4, danach wird das Ergebnis des blauen Würfels anschaut.

e) $P(W_{\text{rot}} + W_{\text{blau}} = 9 | W_{\text{rot}} = 4) = 1 \cdot \frac{1}{6} = \frac{1}{6}$



f) $P(W_{\text{rot}} + W_{\text{blau}} \geq 9 | W_{\text{rot}} = 4) = 1 \cdot \frac{2}{6} = \frac{1}{3}$



g) $P(W_{\text{blau}} = 5 | W_{\text{rot}} = 4) = 1 \cdot \frac{1}{6} = \frac{1}{6}$



Aufgabe 7

Die Rechnungen und dazugehörige Erklärung sind handschriftlich am Ende zu finden. Die zu Erstellende „Zeichnung“ ist mit Abbildung 1 gegeben. Dazu ist noch anzumerken:

- c) Die gewünschte Ellipse ist die 1σ -Ellipse, und als solche gekennzeichnet. Die Erwartungswerte sind zusammen mit den Standardabweichungen als Fehlerbalken dargestellt.
- d) Die Größe der Werte σ_x' und σ_y' sind durch die (halbe) Länge der Hauptachsen dargestellt.
- e) Die Länge der Hauptachsen sind $2\sigma_x'$ und $2\sigma_y'$ (vgl. handschriftlicher Teil). Der Winkel zwischen Hauptachsen und Koordinatenachsen ist der negative Drehwinkel zur Diagonalisierung der Kovarianzmatrix (etwa 20° , vgl. handschriftlicher Teil).

- f) Die bedingten Wahrscheinlichkeitsdichten $f(x|y)$ und $f(y|x)$ lassen sich unmöglich noch in der selben Abbildung darstellen, da sie selbst zweidimensionale Verteilungen sind. Wir gehen davon aus, dass der Copy & Paste Satz in der Teilaufgabe eigentlich nicht dazugehört.
- e) Die bedingten Erwartungswerte $E(x|y)$ und $E(y|x)$ sind als durchgezogene Geraden dargestellt, ihre Standardabweichungen als dazu parallele, gepunktete Linien in der selben Farbe.

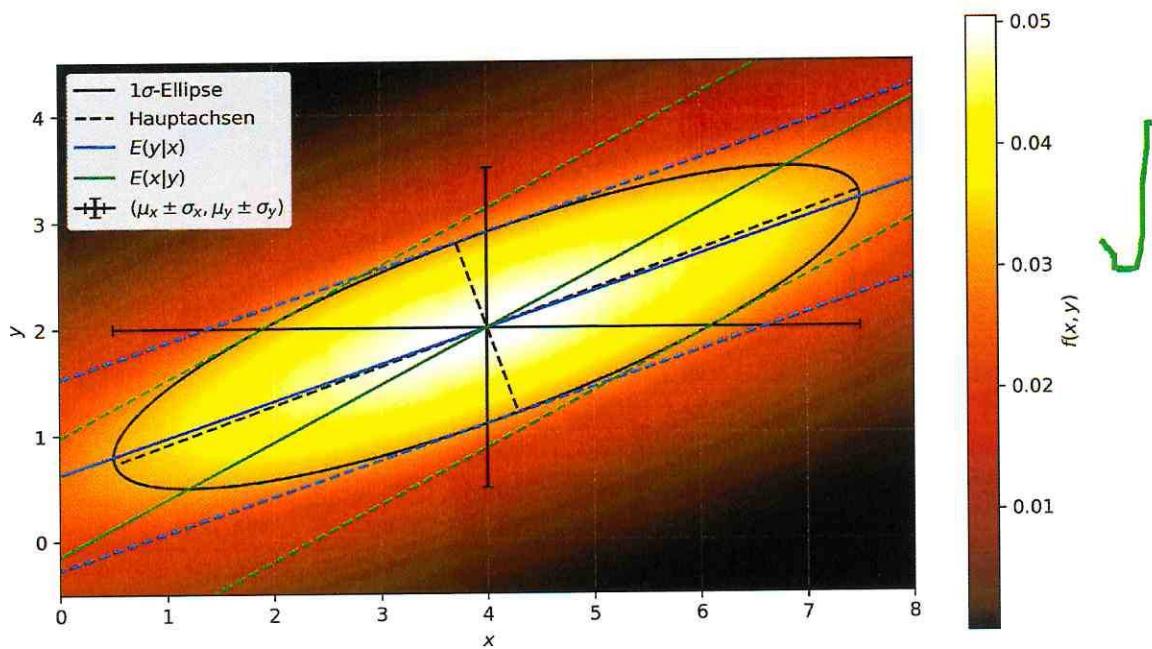


Abbildung 1: Graphische Darstellung der Ergebnisse von Aufgabe 7.

7) a) Der Korrelationskoeffizient ρ berechnet sich wie folgt:

$$\rho(x, y) = \frac{\text{cov}(x, y)}{\sigma(x) \cdot \sigma(y)} = \frac{0,2}{3,5 \cdot 1,5} = 0,8$$



b) Die Wahrscheinlichkeitsdichte $f(x, y)$ wird konstant gesetzt:

$$f(x, y) = k \exp\left[-\frac{1}{2} (\vec{x} - \vec{\alpha})^T B (\vec{x} - \vec{\alpha})\right] \stackrel{!}{=} \text{const}$$

$$B = \frac{1}{\sigma_x^2 \sigma_y^2 - \text{cov}^2(x, y)} \begin{pmatrix} \sigma_y^2 & -\text{cov}(x, y) \\ -\text{cov}(x, y) & \sigma_x^2 \end{pmatrix}$$

$$\det B = \left(\frac{1}{\sigma_x^2 \sigma_y^2 - \text{cov}^2(x, y)} \right)^2 (\sigma_x^2 \sigma_y^2 - \text{cov}^2(x, y))$$

$$k = \left(\frac{\det B}{(2\pi)^2} \right)^{1/2} = \left[\frac{1}{(2\pi)^2 (\sigma_x^2 \sigma_y^2 - \text{cov}^2(x, y))} \right]^{1/2}$$

$$f(x, y) =: C (= \text{const})$$

$$\Leftrightarrow -2 \ln\left(\frac{C}{k}\right) = (\vec{x} - \vec{\alpha})^T B_o (\vec{x} - \vec{\alpha})$$

$$= (x - \mu_x, y - \mu_y) \frac{1}{\sigma_x^2 \sigma_y^2 - \text{cov}^2(x, y)} \begin{pmatrix} \sigma_y^2 & -\text{cov}(x, y) \\ -\text{cov}(x, y) & \sigma_x^2 \end{pmatrix} \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}$$

$$= \frac{1}{\sigma_x^2 \sigma_y^2 - \text{cov}^2(x, y)} ((x - \mu_x) \sigma_y^2 - (y - \mu_y) \text{cov}(x, y), (y - \mu_y) \sigma_x^2 - (x - \mu_x) \text{cov}(x, y)) \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}$$

$$= \frac{1}{\sigma_x^2 \sigma_y^2 - \text{cov}^2(x, y)} \left[(x - \mu_x)^2 \sigma_y^2 - 2(x - \mu_x)(y - \mu_y) \text{cov}(x, y) + (y - \mu_y)^2 \sigma_x^2 \right]$$

$$= \frac{1}{1 - \rho^2} \left[\left(\frac{x - \mu_x}{\sigma_x} \right)^2 - 2 \frac{x - \mu_x}{\sigma_x} \frac{y - \mu_y}{\sigma_y} \frac{\text{cov}(x, y)}{\sigma_x \sigma_y} + \left(\frac{y - \mu_y}{\sigma_y} \right)^2 \right]$$

$$= \frac{1}{1 - \rho^2} [u_x^2 + u_y^2 - 2u_x u_y \rho] := E, \quad u_i = \frac{x - \mu_i}{\sigma_i}$$

Dies entspricht nach Addition von $2 \ln\left(\frac{C}{k}\right)$
der allgemeinen Ellipsenform:

$$ax^2 + by^2 + cx + dy + e = 0$$



c) Da die Amplitude von $f(x, y) = k$ ist, muss, um $f(x, y)$ da zu zeichnen, wo die Funktion auf das $\frac{1}{\sqrt{e}}$ -fache abgefallen ist, die obere Gleichung mit $\frac{e}{k} = e^{-\frac{1}{2}}$ geplottet werden.
 Das plotten geschieht mit der Funktion `matplotlib.pyplot.contour(...)`, wobei die Schnittpunkte eines Grids mit Punkten aus $[0, 8] \times [-0,5 \times 4,5]$ und $E(x, y)$ geplottet werden. Da die linke Seite der Gleichung $= 1$ ist, soll nur die Höhenlinie mit dem Wert 1 gezeichnet werden.
 Diese Ellipse gibt die 1- σ -Umgebung an.



d) Um die Rotationsmatrix M zu bestimmen, wird die Kovarianzmatrix zuerst mit einer allgemeinen Rotationsmatrix transformiert.
 Da die Nebendiagonalelemente einer Kovarianzmatrix Null sind, wenn die Variablen unkorreliert sind, kann über diese Einträge der benötigte Drehwinkel α berechnet werden.

$$\begin{aligned}
 M & \quad \text{COV} \quad M^T \\
 &= \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \begin{pmatrix} \sigma_x^2 & \text{cov}(x, y) \\ \text{cov}(x, y) & \sigma_y^2 \end{pmatrix} \begin{pmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{pmatrix} \\
 &= \begin{pmatrix} \sigma_x^2 \cos^2 \alpha - \text{cov}(x, y) \sin \alpha & \text{cov}(x, y) \cos \alpha - \sigma_y^2 \sin \alpha \\ \sigma_x^2 \sin^2 \alpha + \text{cov}(x, y) \cos \alpha & \text{cov}(x, y) \sin \alpha + \sigma_y^2 \cos \alpha \end{pmatrix} \cdot M^T \\
 &= \begin{pmatrix} \sigma_x^2 \cos^2 \alpha - 2 \text{cov} \sin \alpha \cos \alpha + \sigma_y^2 \sin^2 \alpha \\ \sigma_x^2 \sin^2 \alpha + \text{cov} \cos^2 \alpha - \text{cov} \sin^2 \alpha - \sigma_y^2 \cos \alpha \sin \alpha \\ \sigma_x^2 \sin \alpha \cos \alpha - \text{cov} \sin^2 \alpha + \text{cov} \cos^2 \alpha - \sigma_y^2 \cos \alpha \sin \alpha \\ \sigma_x^2 \sin^2 \alpha + 2 \text{cov} \cos \alpha \sin \alpha + \sigma_y^2 \cos^2 \alpha \end{pmatrix} \\
 &\doteq \begin{pmatrix} \sigma_{x'}^2 & 0 \\ 0 & \sigma_{y'}^2 \end{pmatrix}
 \end{aligned}$$

$$\Rightarrow \sigma_x^2 \sin \alpha \cos \alpha - \text{cov} \sin^2 \alpha + \text{cov} \cos^2 \alpha - \sigma_y^2 \cos \alpha \sin \alpha = 0$$

$$\Leftrightarrow (\sigma_x^2 - \sigma_y^2) \sin \alpha \cos \alpha = \text{cov} (\sin^2 \alpha - \cos^2 \alpha)$$

$$\Leftrightarrow \frac{\sigma_x^2 - \sigma_y^2}{\text{cov}} = \frac{\sin^2 \alpha - \cos^2 \alpha}{\sin \alpha \cos \alpha} = -2 \cot(2\alpha)$$

$$\Leftrightarrow \frac{\text{cov}}{\sigma_x^2 - \sigma_y^2} = -\frac{1}{2} \tan(2\alpha) = \frac{1}{2} \tan(-2\alpha)$$

$$\Rightarrow -2\alpha = \arctan \left(\frac{2\text{cov}}{\sigma_x^2 - \sigma_y^2} \right)$$

$$\Rightarrow \alpha = -\frac{1}{2} \arctan \left(\frac{2\text{cov}}{\sigma_x^2 - \sigma_y^2} \right) \approx -20,0^\circ$$

$$\Rightarrow \sigma_x^2 = [\sigma_x^2 \cos^2 \alpha + \sigma_y^2 \sin^2 \alpha - 2 \text{cov} \cos \alpha \sin \alpha]^{1/2} \approx 3,71$$

$$\sigma_y^2 = [\sigma_x^2 \sin^2 \alpha + \sigma_y^2 \cos^2 \alpha + 2 \text{cov} \cos \alpha \sin \alpha]^{1/2} \approx 0,849$$

$$M = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}$$

$$u_y = \frac{y - \mu_y}{\sigma_y} \quad \frac{du_y}{dy} = \frac{1}{\sigma_y}$$

$$f) f(y|x) = \frac{f(x,y)}{g(x)}, \quad g(x) = \int_{-\infty}^{\infty} f(x,y) dy \quad dy = \sigma_y du_y$$

$$g(x) = k \sigma_y \int_{-\infty}^{\infty} \exp \left[-\frac{1}{2(1-s^2)} (u_x^2 + u_y^2 - 2u_x u_y s) \right] du_y$$

$$= k \sigma_y \int_{-\infty}^{\infty} \exp \left[-\frac{1}{2(1-s^2)} \underbrace{(u_x^2 + u_y^2 - 2u_x u_y s + u_x^2 s^2 - u_x^2 s^2)}_{(u_y - u_x s)^2} \right] du_y$$

$$= k \sigma_y \int_{-\infty}^{\infty} \exp \left[-\frac{1}{2(1-s^2)} (u_y - u_x s)^2 \right] du_y \cdot \exp \left[-\frac{1}{2(1-s^2)} u_x^2 (1-s^2) \right]$$

$$= k \sigma_y \exp \left[-\frac{u_x^2}{2} \right] \sqrt{2(1-s^2) \cdot \pi}$$

$$\Rightarrow f(y|x) = \underbrace{\frac{1}{\sigma_y} (2(1-s^2) \cdot \pi)^{1/2} \cdot \exp \left[-\frac{1}{2(1-s^2)} (u_y^2 + u_x^2 - 2u_x u_y s + \frac{u_x^2}{2}) \right]}_{=: k_y}$$

$$= k_y \exp \left[-\frac{1}{2(1-s^2)} (u_y^2 + u_x^2 - 2u_x u_y s - u_x^2 (1-s^2)) \right]$$

$$= \tilde{k}_y \exp \left[-\frac{1}{2(1-s^2)} (u_y^2 - 2u_x u_y s + u_x^2 s^2 + u_x^2 - u_x^2) \right]$$

$$= \tilde{k}_y \exp \left[-\frac{1}{2(1-s^2)} (u_y - u_x s)^2 \right]$$

$$f(y|x) = c, \quad c = \frac{1}{\sqrt{\pi}} \quad (\text{1-d-Umgebung})$$

$$\Rightarrow h(e^{-y/2}) = -\frac{1}{2}$$

$$= -\frac{1}{2(1-s^2)} (u_y - u_x s)^2$$

$$= -\frac{1}{2(1-s^2)} \left(\frac{y-\mu_y}{\sigma_y} - \frac{x-\mu_x}{\sigma_x} s \right)^2$$

$$\Leftrightarrow 1 = \frac{1}{1-s^2} \left(\frac{y-\mu_y}{\sigma_y} - \frac{x-\mu_x}{\sigma_x} s \right)^2$$

$$\overline{f(x|y)} = \frac{f(x,y)}{h(y)}, \quad h(y) = \int_{-\infty}^{\infty} f(x,y) dx, \quad u_x = \frac{x-\mu_x}{\sigma_x}, \quad dx = \sigma_x du_x$$

$$h(y) = k \sigma_x \int_{-\infty}^{\infty} \exp \left[-\frac{1}{2(1-s^2)} (u_x^2 + u_y^2 - 2u_x u_y s) \right] du_x$$

$$= k \sigma_x \int_{-\infty}^{\infty} \exp \left[-\frac{1}{2(1-s^2)} (u_x^2 - 2u_x u_y s + u_y^2 s^2 + u_y^2 s^2 + u_y^2) \right] du_x$$

$$= k \sigma_x \int_{-\infty}^{\infty} \exp \left[-\frac{1}{2(1-s^2)} (u_x - u_y s)^2 \right] du_x \exp \left[-\frac{1}{2(1-s^2)} u_y^2 (1-s^2) \right]$$

$$= k \sigma_x \exp \left[-\frac{u_y^2}{2} \right] \sqrt{2(1-s^2) \pi}$$

$$\Rightarrow f(x|y) = \underbrace{\frac{1}{\sigma_x} \left(\frac{1}{2(1-s^2)} \pi \right)^{-1/2}}_{=: \tilde{k}_x} \exp \left[-\frac{1}{2(1-s^2)} (u_x^2 + u_y^2 - 2u_x u_y s) + \frac{u_y^2}{2} \right]$$

$$= \tilde{k}_x \exp \left[-\frac{1}{2(1-s^2)} (u_x^2 + u_y^2 - 2u_x u_y s - u_y^2 (1-s^2)) \right]$$

$$= \tilde{k}_x \exp \left[-\frac{1}{2(1-s^2)} (u_x - u_y s)^2 \right]$$

$$f(x|y) \stackrel{!}{=} c; \frac{c}{k_x} \stackrel{!}{=} e^{-\gamma_2} \quad (\text{no-Umgebung})$$

$$\Rightarrow \ln(e^{-\gamma_2}) = -\gamma_2$$

$$= -\frac{1}{2(1-\beta^2)} (u_x - u_y s)^2$$

$$\Leftrightarrow 1 = \frac{1}{1-\beta^2} \left(\frac{x-\mu_x}{\sigma_x} - \frac{y-\mu_y}{\sigma_y} s \right)^2$$

$$g) E(y|x) = \int_{-\infty}^{\infty} y f(y|x) dy, \quad u_y := \frac{y-\mu_y}{\sigma_y}, \quad dy = \sigma_y du_y \\ y = \sigma_y u_y + \mu_y$$

$$= k_y \sigma_y \int_{-\infty}^{\infty} (\sigma_y u_y + \mu_y) \exp \left[-\frac{1}{2(1-\beta^2)} (u_y - u_x s)^2 \right] du_y$$

$$= k_y \sigma_y \left[\mu_y + \sqrt{2(1-\beta^2)} u_x s + \sigma_y \sqrt{2(1-\beta^2)} u_y \right]$$

$$= \mu_y + \sigma_y u_x s$$

$$= \mu_y + \sigma_y s \frac{x-\mu_x}{\sigma_x}$$

$$\overline{E(x|y)} = \int_{-\infty}^{\infty} x f(x|y) dx, \quad \left| \begin{array}{l} u_x = \frac{x-\mu_x}{\sigma_x}, \quad \frac{du_x}{dx} = \frac{1}{\sigma_x}, \quad dx = \sigma_x du_x, \\ x = \sigma_x u_x + \mu_x \end{array} \right.$$

$$= k_x \sigma_x \int_{-\infty}^{\infty} (\sigma_x u_x + \mu_x) \exp \left[-\frac{1}{2(1-\beta^2)} (u_x - u_y s)^2 \right] du_x$$

$$= \mu_x + \sigma_x u_y s$$

$$= \mu_x + \sigma_x s \frac{y-\mu_y}{\sigma_y}$$

SMO A5

$$f(v) = N \exp\left(-\frac{mv^2}{2kT}\right) 4\pi v^2$$

Normierung:

$$(fg)' = f'g + fg'$$

$$\Rightarrow \int f'g' = \int f g' - \int f' g$$

$$\int_0^\infty f(v) dv = 1$$

$$= 4\pi N \int_0^\infty v^2 \exp\left(-\frac{mv^2}{2kT}\right) dv$$

Ausarbeiten

$$I := \int_0^\infty \exp(-\alpha t x^2) dx$$

$$\begin{aligned} I^2 &= \int_0^\infty \exp(-\alpha t x^2) dx \int_0^\infty \exp(-\alpha t y^2) dy \\ &= \int_0^\infty \int_0^\infty \exp(-\alpha t(x^2+y^2)) dx dy \end{aligned}$$

→ Polar $x = r \sin \varphi$ $y = r \cos \varphi$

$$= \int_0^{\pi/2} d\varphi \int_0^\infty \exp(-\alpha t r^2) r dr$$

$$= \frac{\pi}{4} \cdot \frac{1}{2} \int_0^\infty \exp(-\alpha t u) du$$

$$= +\frac{\pi}{4} \frac{1}{2t} \underbrace{\textcircled{1}}_{\text{wegen } I > 0}$$

$$\Rightarrow I_0 = +\sqrt{I^2} = \alpha \frac{\pi}{2} \sqrt{\frac{\pi}{2t}}$$

~~$$\text{Daher: } \frac{d}{dt} |I| = +\frac{1}{2} \sqrt{\frac{\pi}{2t}} \frac{-3/2}{t^2} \Big|_{t=1} = \frac{\sqrt{\pi/2}}{2}$$~~

~~$$\partial R M = \frac{4\pi}{V2\pi k} = \frac{8e^{-2}}{2} = \frac{4e^{-2}}{2}$$~~

Dann: ϵ :

$$\int_0^\infty x^2 e^{-\epsilon x^2} dx = \frac{d}{dt} I \Big|_{t=1} = -\frac{d}{dt} \frac{1}{2} \sqrt{\pi} (\epsilon t)^{-\frac{1}{2}} \Big|_{t=1}$$

$$= \frac{1}{4} \sqrt{\frac{\pi}{\epsilon}} \times \cancel{\frac{1}{2} \sqrt{\pi} t^{-3/2}} \Big|_{t=1} = \cancel{\frac{1}{2}}$$

$$= \frac{1}{4} \sqrt{\frac{\pi}{\epsilon}} \cancel{\frac{1}{2}} = \frac{\sqrt{\pi}}{4 \epsilon^{3/2}} //$$

$$\Rightarrow \int_0^\infty f(v) dv = 4\pi N \cdot \frac{\sqrt{\pi}}{4} \left(\frac{2\pi kT}{m}\right)^{3/2}$$

$$\Rightarrow N = \left(\frac{2\pi kT}{m}\right)^{3/2} //$$

$$N = \left(\frac{m}{2\pi kT}\right)^{3/2}$$

a)

$$f' = -N \frac{m}{k_B T} 4\pi v^2 \exp\left(-\frac{mv^2}{2k_B T}\right)$$

$$+ 2\pi m v \exp\left(-\frac{mv^2}{2k_B T}\right)$$

~~$\partial f'/\partial v = 0$~~

$$= 4\pi m v \exp\left(-\frac{mv^2}{2k_B T}\right) \left(2 - \frac{m}{k_B T} v^2\right) \stackrel{!}{=} 0$$

$$\Rightarrow v_m = \sqrt{\frac{2k_B T}{m}}$$

b)

$$\langle v \rangle_f = \int_0^\infty v f(v) dv = 4\pi N \int_0^\infty v^3 \exp\left(-\frac{mv^2}{2k_B T}\right) dv$$

~~$u = v^2 \Rightarrow du = 2v dv$~~

$$\Rightarrow 4\pi N \int_0^\infty \frac{u}{2} \exp\left(-\frac{mu}{2k_B T}\right) du$$

$$= 2\pi N \cdot \left[\left. \frac{u^2 k_B T}{m} \exp\left(-\frac{mu}{2k_B T}\right) \right]_0^\infty$$

$$= 0$$

$$+ \int_0^{2k_B T/m} \exp\left(-\frac{mu}{2k_B T}\right) du \Big)$$

$$= 2\pi N \left(\frac{k_B T}{m}\right)^2 = 2\pi \left(\frac{m}{2\pi k_B T}\right)^{1/2} \left(\frac{2k_B T}{m}\right)^2$$

$$= 2 \sqrt{\frac{2k_B T}{\pi m}} = \frac{2}{\sqrt{\pi}} v_m$$

c)

$$\frac{1}{2} \doteq \int_0^{V_m} f(v) dv$$

$$= N_{\text{FA}} \int_0^{V_m} v^3 \exp\left(-\frac{v^2}{V_m^2}\right) dv$$

~~$$x = \frac{v^2}{V_m^2} \Rightarrow \frac{dx}{dv} = 2 \frac{v}{V_m^2} \Rightarrow dv = 2 \frac{V_m^2}{v} dx$$~~

d)

$$x_{\text{FWHM}} = \frac{V_{\text{FWHM}}}{V_m}$$

$$f(x) = x_{\text{FWHM}}^2 \exp(-x_{\text{FWHM}}^2) - \frac{\exp(-1)}{2} = 0$$

~~$$f' = 2x \exp(-x^2) + -2x^3 \exp(-x^2)$$~~

$$= 2x \exp(-x^2) (1 - 2x^2)$$

Zeit	Raum	Abgabe im Moodle; Mails mit Betreff: [SMD1718]
Di. 10-12	CP-03-150	philipp2.hoffmann@udo.edu und jan.soedingrekso@udo.edu
Di. 16-18	P1-02-110	felix.neubuerger@udo.edu und tobias.hoinka@udo.edu
Di. 16-18	CP-03-150	simone.mender@udo.edu und maximilian.meier@udo.edu

Aufgabe 8: Zufallszahlen verschiedener Verteilungen

5 P.

Die Zufallsvariable x möge der Wahrscheinlichkeitsdichte

$$f(x) = \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & \text{sonst} \end{cases}$$

(Gleichverteilung zwischen 0 und 1) genügen.

- Mit welcher Wahrscheinlichkeit nimmt x einen Wert zwischen $\frac{1}{2}$ und $\frac{1}{3}$ an?
- Mit welcher Wahrscheinlichkeit nimmt x den exakten Wert $\frac{1}{2}$ an?
- Mit welcher Wahrscheinlichkeit liefert ein Zufallsgenerator auf einem Computer den exakten Wert $\frac{1}{2}$? Der Generator soll sein Ergebnis in Form einer binären Gleitkommazahl mit einer Mantisse von 23 Binärstellen darstellen.
- Mit welcher Wahrscheinlichkeit liefert derselbe Zufallsgenerator den exakten Wert $\frac{2}{3}$?

Aufgabe 9: Zufallszahlengeneratoren

7 P.

Linear-kongruente Zufallszahlengeneratoren erzeugen eine neue ganzzahlige Zufallszahl aus der vorhergehenden durch die Vorschrift

$$x_n = (a \cdot x_{n-1} + b) \mod m.$$

Division durch m ergibt dann eine zwischen 0 und 1 gleichverteilte reelle Zufallszahl.

- Programmieren Sie einen solchen Zufallszahlengenerator mit $b = 3$ und $m = 1024$. Bestimmen Sie die Periodenlänge in Abhängigkeit des Parameters a , indem Sie für a Werte aus einem angemessenen Bereich verwenden. Stellen Sie den Zusammenhang von Periodenlänge und a in einem Plot dar. Wie groß ist die maximale Periodenlänge? Für welche Werte von a ist die Periodenlänge maximal? Lassen sich die erhaltenen Werte mit den Regeln für gute linear-kongruente Generatoren erklären? Hinweis: In dieser Aufgabe sollte der Startwert x_0 unverändert bleiben.
- Verwenden Sie für die folgenden Aufgaben einen linear-kongruenten Zufallszahlengenerator mit den Parametern $a = 1601$, $b = 3456$ und $m = 10\,000$.

- b) Exponentialgesetz: $f(t) = Ne^{-t/\tau}$ in den Grenzen 0 bis ∞ (N = Normierungskonstante)
- c) Potenzgesetz: $f(x) = Nx^{-n}$ in den Grenzen x_{\min} bis x_{\max} ($n \geq 2$, N = Normierungskonstante)
- d) Cauchy-Verteilung:

$$f(x) = \frac{1}{\pi} \frac{1}{1+x^2}$$

in den Grenzen $-\infty$ bis ∞

- e) Die durch das (im Moodle unter *empirisches_histogramm.npy* zu findene) Histogramm gegebene empirische Verteilung. Die Datei enthält Binzentren (*bin_mid*) und die Höhen (*hist*). Das Histogramm besteht aus 50 Bins zwischen 0,0 und 1,0.

Zum Einlesen und Darstellen dieses Histogramms können Sie z.B. so vorgehen:

```
1 from matplotlib import pyplot as plt
2 import numpy as np
3 data = np.load("empirisches_histogramm.npy")
4 plt.hist(data['bin_mid'], bins=np.linspace(0., 1., 51),
5           weights=data['hist'])
6 plt.show()
```

d)

Um zu testen, ob die Zahl $\frac{2}{3}$ darstellbar ist, wird berechnet, ob $\frac{2}{3}$ ein ganzzahliges Vielfaches von $\frac{1}{2^{23}-1}$ ist (f)

$$x \frac{1}{2^{23}-1} = \frac{2}{3} \quad (6)$$

$$\Leftrightarrow x = \frac{2(2^{23}-1)}{3} \quad (7)$$

$$\approx 5592404.66667 \notin \mathbb{N}. \quad (8)$$

Damit ist die Zahl $\frac{2}{3}$ nicht exakt darstellbar und die Wahrscheinlichkeit sie zu berechnen ist Null.

5/6

Aufgabe 9

a)

Gegeben ist der Linear-kongruenten Zufallszahlengenerator

$$x_n = (ax_{n-1} + b) \bmod m \quad (9)$$

mit $b = 3$ und $m = 1024$. Damit ist sofort klar, dass die maximale auftretende Periodenlänge nicht größer als $m = 1024$ sein kann. Wir bestimmen die Periodenlänge für alle a von 0 bis 1024. Das Ergebnis ist in Abbildung 1 dargestellt. Zusätzlich ist in Abbildung 2 der Bereich für a von 0 bis 32 höher aufgelöst dargestellt. Für die erzeugten Abbildungen wurde der Startwert $x_0 = 1$ verwendet, das Ergebnis ist unseren Tests nach jedoch unabhängig vom Startwert. Die maximale Periodenlänge ist wie erwartet $2^{10} = 1024$.

Das Verhalten lässt sich sehr gut durch die Regeln für gute linear-kongruente Generatoren [1, S. 7] erklären: Die Regeln 1 und 2 sind erfüllt. Die Regel 3 ist für alle ungeraden Zahlen erfüllt, da die Primfaktorzerlegung von m durch 2^{10} gegeben ist. Wie gut zu erkennen ist, ist die Periodenlänge für alle geraden a minimal eins, was die Regel bestätigt. Für alle a , für die gilt, dass $a - 1$ außerdem durch 4 teilbar ist (Regel 4) ist die Periodenlänge maximal. Leider haben wir gerade keine Zeit mehr uns länger Gedanken darüber zu machen, welchen Regeln die übrigen Werte folgen, aber offenbar bildet sich bei jedem $a + 1 = 2^n$ eine neue Ebene mit der Periodenlänge 2^{11-n} .

c)

Das geforderte Histogramm ist durch Abbildung 3 gegeben. Das Ergebnis wurde für viele verschiedene Startwerte berechnet und scheint unabhängig von diesem zu sein. Es entspricht nicht den Anforderungen an einen guten Zufallszahlengenerator, da offenbar keine echte Gleichverteilung vorliegt, sondern manche Zahlen signifikant wahrscheinlicher sind als andere.

✓

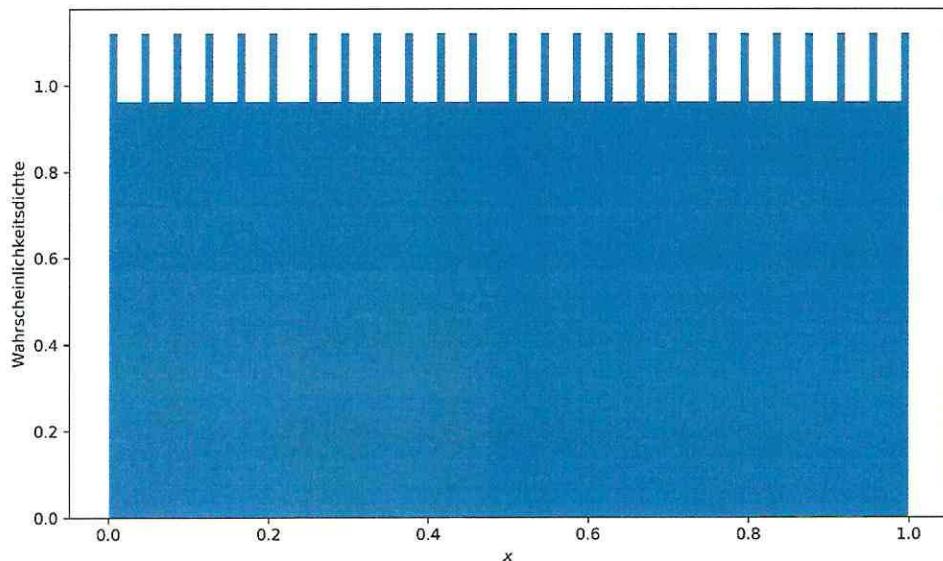


Abbildung 3: Histogramm für 10000 nach 9 erzeugte Zufallszahlen.

d)

Die geforderten Streudiagramme sind durch die Abbildungen 4 und 5 gegeben. Es ist klar zu erkennen, dass die Zahlen von einem schlechten Zufallszahlengenerator generiert wurden, da sich Ebenen bilden. Es besteht also eine vorhersagbare Korrelation zwischen aufeinanderfolgenden Zufallszahlen.

✓

e)

Die geforderten Streudiagramme sind durch die Abbildungen 6 und 7 gegeben. Es ist klar zu erkennen, dass die Zahlen von einem guten Zufallszahlengenerator generiert wurden, da sich keine erkennbaren Muster bilden. Es besteht also keine vorhersagbare Korrelation zwischen aufeinanderfolgenden Zufallszahlen.

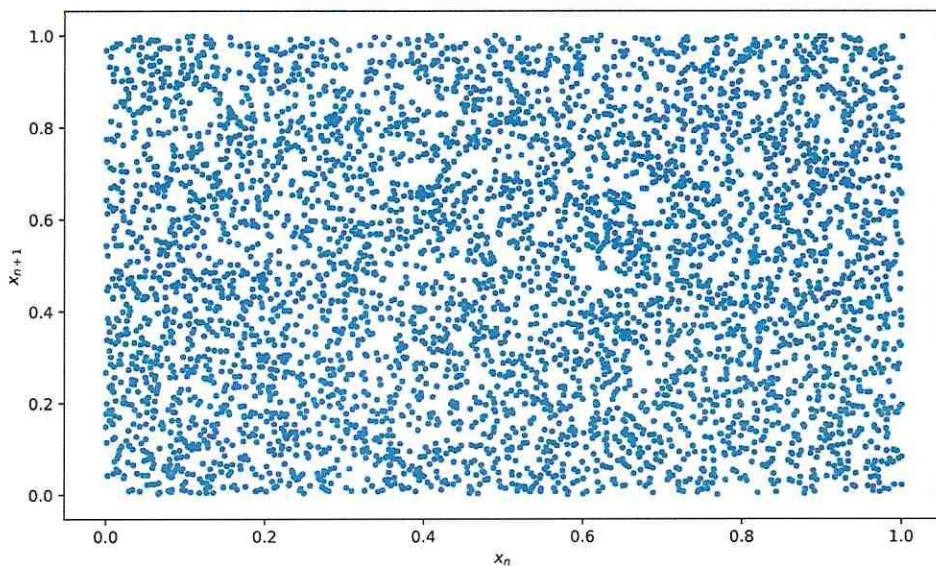


Abbildung 6: Zweidimensionales Streudiagramm für den NUMPY Zufallszahlengenerator.

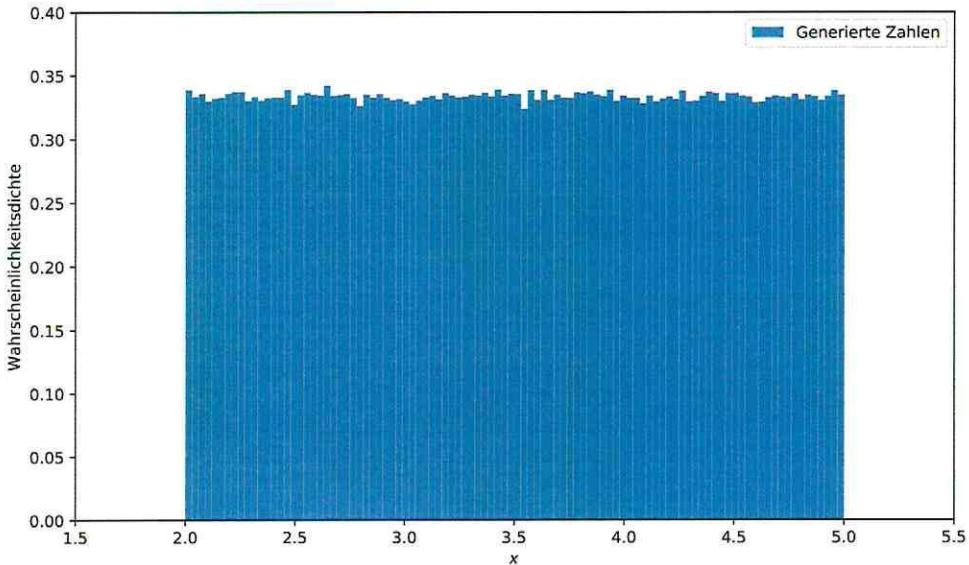


Abbildung 8: Histogramm der transformierten Gleichverteilung mit den Grenzen $x_{\min} = 2$ und $x_{\max} = 5$.

b)

Die gewünschte Dichtefunktion

$$f(t) = N \exp\left(-\frac{t}{\tau}\right), \quad t \in [0, \infty) \quad (11)$$

lässt sich integrieren. Daher kann zur Erzeugung dieser Verteilung die Transformationsmethode verwendet werden. Für die Verteilungsfunktion ergibt sich

$$F(t) = \int_0^t N \exp\left(-\frac{x}{\tau}\right) dx = 1 - \frac{\exp\left(-\frac{t}{\tau}\right)}{\tau}, \quad (12)$$

mit der Normierungskonstante $N = \tau^{-1}$. Diese kann nun invertiert werden:

$$z = F(\underline{x}) = 1 - \frac{\exp\left(-\frac{t}{\tau}\right)}{\tau} \quad (13)$$

$$\Leftrightarrow (1-z)\tau = \exp\left(\frac{t}{\tau}\right) \quad (14)$$

$$\Rightarrow \underline{x} = -\tau \log((1-z)\tau), \quad z \in \left[1 - \frac{1}{\tau}, 1\right]. \quad (15)$$

Wenn z gleichverteilte Zufallszahlen mit den angegebenen Grenzen sind, dann sind die so transformierten x mit der gewünschten Dichte f verteilt. z kann hier natürlich wieder gemäß (10) erzeugt werden, diese Transformation ließe sich auch direkt hier einsetzen. Zur Verifikation erstellen wir auf diese Weise ein Histogramm mit $\tau = 1$, dieses ist in Abbildung 9 dargestellt.

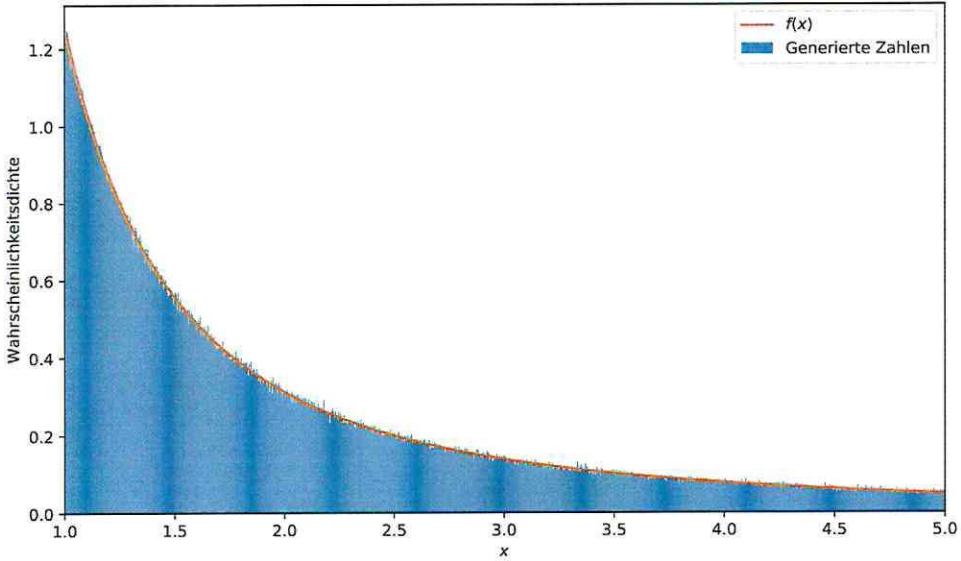


Abbildung 10: Histogramm der generierten Verteilung aus Aufgabe 10c).

d)

Die Wahrscheinlichkeitsdichte der CAUCHY-Verteilung

$$f(x) = \frac{1}{\pi} \frac{1}{1+x^2}, \quad x \in (-\infty, \infty) \quad (21)$$

ist bereits normiert und lässt sich integrieren. Daher verwenden wir wieder die Transformationsmethode. Die Verteilungsfunktion lautet

$$F(x) = \frac{1}{\pi} \int_{-\infty}^x f(x') dx' = \frac{\arctan x}{\pi} + \frac{1}{2}. \quad (22)$$

Invertieren der Verteilungsfunktion ergibt

$$F(x) = z \quad (23)$$

$$\Rightarrow x = -\cot(\pi z), \quad z \in [0, 1] \quad (24)$$

Wenn z gleichverteilte Zufallszahlen zwischen 0 und 1 sind, dann sind die so transformierten x CAUCHY-verteilten. Zur Verifikation erstellen wir auf diese Weise ein Histogramm, dieses ist in Abbildung 11 dargestellt.

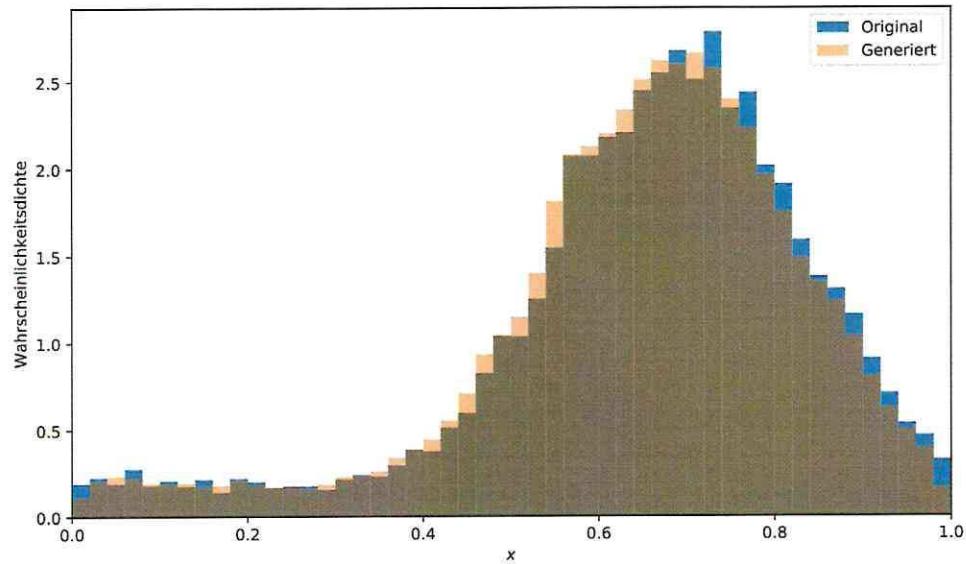


Abbildung 12: Darstellung des mittels interpolierter Verteilungsfunktion generierten Histogramms zusammen mit dem ursprünglichen Histogramm aus Aufgabe 10e)

Literatur

- [1] Wolfgang Rhode *Generation von Zufallszahlen* https://moodle.tu-dortmund.de/pluginfile.php/618120/mod_resource/content/1/Generation%20Zufallszahlen.pdf

Code für blatt02

Bange, Burkowitz, Harnisch

13. November 2017

```
.../blatt02/Bange_Burkowitz_Harnisch/aufg10.py

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy.interpolate import interp1d
4 from pylab import rcParams
5
6
7 # _____ a) _____
8 def dist_a(x_min=0, x_max=1, N=1e6):
9     """Liefert zwischen x_min und x_max gleichverteilte Zufallszahlen"""
10    return (x_max - x_min)*np.random.uniform(size=int(N)) + x_min
11
12
13 def aufg10a(x_min, x_max, N=1e6):
14     """Validiert das Ergebnis für Aufgabe 10a graphisch"""
15     plt.hist(dist_a(x_min, x_max, N), bins=100, density=True,
16             label="Generierte Zahlen")
17     plt.xlim(1.5, 5.5)
18     plt.ylim(0, 0.4)
19     plt.xlabel(r"\$x\$")
20     plt.ylabel("Wahrscheinlichkeitsdichte")
21     plt.legend()
22     # plt.show()
23     plt.savefig("A10a.pdf")
24     plt.clf()
25
26
27 # _____ b) _____
28 def dist_b(tau, N=1e6):
29     """Liefert N nach f(x)=exp(-x/tau)/tau verteilte Zufallszahlen"""
30     return -tau*np.log((1 - np.random.uniform(
31         low=1, high=1 - 1/tau, size=int(N)))*tau)
32
33
34 def aufg10b(tau, N=1e6):
35     """Validiert das Ergebnis für Aufgabe 10b graphisch"""
36     t = dist_b(tau, N)
37
38     x = np.linspace(0, 6, 10000)
39     plt.hist(t, bins=1000, density=True, label="Generierte Zahlen")
40     plt.plot(x, 1/tau*np.exp(-x/tau), label=r"\$f(t)\$")
41     plt.xlim(0, 6)
42     plt.xlabel(r"\$t\$")
43     plt.ylabel("Wahrscheinlichkeitsdichte")
44     plt.legend()
45     # plt.show()
46     plt.savefig("A10b.pdf")
47     plt.clf()
48
49
50 # _____ c) _____
51 def dist_c(n, x_min, x_max, N=1e6):
52     """
53     Liefert N nach  $(1-n)/(x_{\max}^{(1-n)} - x_{\min}^{(1-n)})/x^n$  verteilte
54     Zufallszahlen
55
```

```

54 """
55     if n < 2 or not isinstance(n, int):
56         raise ValueError("n muss ein Integer größer oder gleich 2 sein")
57     return (np.random.uniform(size=int(N))*(x_max***(1 - n) - x_min***(1 - n))
58             + x_min***(1 - n))***(1/(1 - n))
59
60
61
62 def aufg10c(n, x_min, x_max, N=1e6):
63     """Validiert das Ergebnis für Aufgabe 10c graphisch"""
64     x = dist_c(n, x_min, x_max, N)
65
66     x1 = np.linspace(x_min, x_max, 10000)
67     plt.hist(x, bins=1000, density=True, label="Generierte Zahlen")
68     plt.plot(x1, (1 - n)/(x_max***(1 - n) - x_min***(1 - n))/x1**n,
69               label=r"$f(x)$")
70     plt.xlim(x_min, x_max)
71     plt.xlabel(r"$x$")
72     plt.ylabel("Wahrscheinlichkeitsdichte")
73     plt.legend()
74     # plt.show()
75     plt.savefig("A10c.pdf")
76     plt.clf()
77
78
79 # ----- d) -----
80
81 def dist_d(N=1e6):
82     """Liefert N cauchy-verteilte Zufallszahlen"""
83     return -1/np.tan(np.pi*np.random.uniform(size=int(N)))
84
85
86
87 def aufg10d(N=1e6):
88     """Validiert das Ergebnis für Aufgabe 10d graphisch"""
89     x = dist_d(N)
90
91     x1 = np.linspace(-5, 5, 10000)
92     # Histogramm ist hier sehr problematisch, weil auch sehr große und sehr
93     # kleine Werte noch relativ wahrscheinlich sind darum einschränken,
94     # dadurch ist die Normierung aber nicht mehr ganz exakt
95     # folgendes funktioniert warum auch immer nicht korrekt:
96     # plt.hist(x, bins=np.concatenate(([min(x)], np.linspace(-10, 10, 1000),
97     #                                     [max(x)])),
98     #           density=True, label="Generierte Zahlen")
99     # daher gute Näherung nehmen
100    plt.hist(x, bins=np.linspace(-100, 100, 10000), density=True,
101              label="Generierte Zahlen")
102    plt.plot(x1, 1/(np.pi*(1 + x1**2)), label=r"$f(x)$")
103    # plt.plot(x1, np.arctan(x1)/np.pi + 0.5)
104    plt.xlim(-5, 5)
105    plt.xlabel(r"$x$")
106    plt.ylabel("Wahrscheinlichkeitsdichte")
107    plt.legend()
108    # plt.show()
109    plt.savefig("A10d.pdf")
110    plt.clf()
111
112
113 # ----- e) -----
114 def dist_e(N=1e6):
115     """
116         Liefert N nach der empirischen Verteilung verteilte Zufallszahlen unter
117         Verwendung einer interpolierten Verteilungsfunktion
118     """
119
120     data = np.load("empirisches_histogramm.npy")
121     # 1/50 = 0.02 ist die Binbreite
122     cumsum = np.cumsum(data["hist"])/(np.sum(data["hist"])*0.02)*0.02
123     # Inverse der Verteilungsfunktion interpolieren
124     # Würde man natürlich normalerweise speichern und nicht jedes mal neu
125     # interpolieren
126     inverse = interp1d(cumsum, data["bin_mid"])
127     return inverse(np.random.uniform(cumsum[0], cumsum[-1], size=int(N)))
128

```

```

129
130 def aufg10e():
131     """Validiert das Ergebnis für Aufgabe 10e graphisch"""
132     data = np.load("empirisches_histogramm.npy")
133     plt.hist(data['bin_mid'], bins=np.linspace(0, 1, 51),
134               weights=data['hist'], density=True, label="Original")
135     plt.hist(dist_e(1e7), bins=np.linspace(0, 1, 51), density=True, alpha=0.5,
136               label="Generiert")
137     plt.legend()
138     plt.xlabel(r"$x$")
139     plt.ylabel("Wahrscheinlichkeitsdichte")
140     plt.xlim(0, 1)
141     # plt.show()
142     plt.savefig("A10e.pdf")
143
144
145 # --- Main ---
146 if __name__ == '__main__':
147     np.random.seed(1234)
148     aufg10a(2, 5)
149     aufg10b(1)
150     aufg10c(2, 1, 5)
151     aufg10d()
152     aufg10e()

```

```
.../blatt02/Bange_Burkowitz_Harnisch/aufg9.py
```

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from mpl_toolkits.mplot3d import Axes3D
4 from pylab import rcParams
5
6
7 # --- a) ---
8 def findPeriod(a, x0=1):
9     """
10     Findet die Periodenlänge des Linear-Kongruenten-Generators aus Aufgabe
11     9a) in Abhängigkeit von a (und x0)
12     """
13
14     numbers = [x0]
15     while not np.isin([numbers[-1]], numbers[:-1]):
16         numbers.append((a*numbers[-1] + 3) % 1024)
17
18     numbers = np.asarray(numbers)
19     first = np.where(numbers[:-1] == numbers[-1])
20     return len(numbers) - first[0][0] - 1
21
22
23 def aufg9a():
24     """
25     Findet für alle ganzen a von 0 bis 99 die Periodenlänge und erstellt
26     daraus einen Plot
27     """
28     fig, ax = plt.subplots()
29     a = np.arange(0, 1024)
30     ax.plot(a, np.vectorize(findPeriod)(a), ".")
31     ax.set_xlabel(r"$a$")
32     ax.set_ylabel("Periodenlänge")
33     ax.set_yscale("log", basey=2)
34     ax.set_yticks([1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024])
35     ax.set_xticks(range(0, 1025, 64))
36     ax.grid(axis="y")
37     # plt.show()
38     fig.savefig("A9a_full.pdf")
39     ax.set_xticks(range(0, 33))
40     ax.set_xlim(-1, 33)
41     fig.savefig("A9a_zoom.pdf")

```

```

43     plt.clf()
44
45
46
47
48 # _____ b) _____
49 def uniform_b(x0, n):
50     """
51     Liefert ein Array mit den ersten n Zufallszahlen des Generators aus
52     Aufgabenteil b)
53     """
54     numbers = [x0]
55     while len(numbers) < n + 1:
56         numbers.append((1601*numbers[-1] + 3456) % 1e4)
57     return np.asarray(numbers[1:])/1e4
58
59
60 # _____ c) _____
61 def aufg9c():
62     """Erstellt das in Aufgabe 9c geforderte Histogramm"""
63     # x0 = [1, 10, 223, 5412, 2812, 899, 1e4 - 39]
64     # for x in x0:
65         # plt.hist(uniform_b(x, 1e6), bins=100, normed=True)
66         # plt.show()
67         # plt.clf()
68     # plt.hist(uniform_b(42, 1e6), bins=100, density=True)
69     plt.hist(uniform_b(42, 10000), bins=100, density=True)
70     plt.xlabel(r"$x$")
71     plt.ylabel("Wahrscheinlichkeitsdichte")
72     # plt.savefig("A9c.pdf")
73     plt.savefig("A9c.png", dpi=300)
74     # plt.show()
75     plt.clf()
76
77
78 # _____ d) _____
79 def aufg9d():
80     """Erstellt die in Aufgabe 9d geforderten Streudiagramme"""
81     numbers = uniform_b(1, 1e4).reshape(5000, 2).T
82     plt.scatter(numbers[0], numbers[1], marker=".")
83     plt.xlabel(r"$x_n$")
84     plt.ylabel(r"$x_{n+1}$")
85     # plt.show()
86     plt.savefig("A9d_2D.pdf")
87     plt.clf()
88
89     numbers = uniform_b(1, 9999).reshape(3333, 3).T
90     fig = plt.figure()
91     ax = fig.add_subplot(111, projection='3d')
92     ax.scatter(numbers[0], numbers[1], numbers[2], marker=".")
93     ax.view_init(elev=10, azim=20)
94     ax.set_xlabel(r"$x_n$")
95     ax.set_ylabel(r"$x_{n+1}$")
96     ax.set_zlabel(r"$x_{n+2}$")
97     # plt.show()
98     plt.savefig("A9d_3D.pdf", tight_layout=True)
99     # plt.savefig("A9d.png", dpi=300, tight_layout=True)
100    plt.clf()
101
102
103 # _____ e) _____
104 def aufg9e():
105     """Erstellt die in Aufgabe 9e geforderten Streudiagramme"""
106     numbers = np.random.uniform(size=10000)
107
108     numbers_tuples = numbers.reshape(5000, 2).T
109     plt.scatter(numbers_tuples[0], numbers_tuples[1], marker=".")
110     plt.xlabel(r"$x_n$")
111     plt.ylabel(r"$x_{n+1}$")
112     # plt.show()
113     plt.savefig("A9e_2D.pdf")
114     plt.clf()
115

```

```

116 numbers_triplets = numbers[:-1].reshape(3333, 3).T
117 fig = plt.figure()
118 ax = fig.add_subplot(111, projection='3d')
119 ax.scatter(numbers_triplets[0], numbers_triplets[1], numbers_triplets[2],
120             marker=".", alpha=0.75)
121 ax.view_init(elev=10, azim=20)
122 ax.set_xlabel(r" $x_n$ ")
123 ax.set_ylabel(r" $x_{n+1}$ ")
124 ax.set_zlabel(r" $x_{n+2}$ ")
125 # plt.show()
126 plt.savefig("A9e_3D.pdf")
127 plt.clf()
128
129
130 # ----- f) -----
131 def aufg9f(N=1e5):
132     """Testet, für welche Startwerte uniform_b wie oft den Wert 0.5 liefert"""
133     count = []
134     for x in range(0, 1025):
135         numbers = uniform_b(x, N)
136         count.append(np.count_nonzero(numbers == 0.5)/N)
137     count = np.asarray(count)
138     # print(count)
139     idx = np.where(count > 0)[0]
140     print(idx/8)
141     print(count[idx])
142
143
144 # ----- Main -----
145 if __name__ == '__main__':
146     np.random.seed(1234)
147     aufg9a()
148     aufg9c()
149     aufg9d()
150     aufg9e()
151     aufg9f()

```

Zeit	Raum	Abgabe im Moodle; Mails mit Betreff: [SMD1718]
Di. 10-12	CP-03-150	philipp2.hoffmann@udo.edu und jan.soedingrekso@udo.edu
Di. 16-18	P1-02-110	felix.neubuerger@udo.edu und tobias.hoinka@udo.edu
Di. 16-18	CP-03-150	simone.mender@udo.edu und maximilian.meier@udo.edu

Aufgabe 11: *Simulationskette für NeutrinoDetector*

20 P.

Verwenden Sie für diese Aufgabe die Pythonbibliothek `pandas`. Füllen Sie die Ergebnisse der einzelnen Teilaufgaben, abgesehen von der Letzten, in ein `DataFrame` und speichern Sie dieses am Ende in einer `hdf5`-Datei `NeutrinoMC.hdf5` mit dem Key `Signal`. Die Ergebnisse der letzten Teilaufgabe sollen in ein eigenes `DataFrame` geschrieben und in der selben `hdf5`-Datei unter dem Key `Background` gespeichert werden. Zum Schreiben einer `hdf5`-Datei besitzt das `DataFrame` die Methode `to_hdf()`.

Wichtig: Die erstellte `hdf5`-Datei soll **nicht** mit abgegeben werden! Das fertige Programm muss ohne die bereits existierende `hdf5`-Datei funktionieren.

a) **Signal MC**

Der Fluss der Neutrinos ist gegeben durch:

$$\Phi = \Phi_0 \cdot \left(\frac{E}{\text{TeV}} \right)^{-\gamma}.$$

Hierbei ist der spektrale Index $\gamma = 2,7$, die untere Energiegrenze 1 TeV und die obere Energiegrenze unendlich.

Simulieren Sie mit Hilfe der Transformationsmethode 10^5 Signalereignisse und speichern Sie diese in dem `DataFrame` unter dem Key `Energy`.

b) **Akzeptanz** Die Wahrscheinlichkeit, ein Ereignis zu detektieren, ist energieabhängig. Dies muss bei der Simulation mit berücksichtigt werden. Die Detektionswahrscheinlichkeit lässt sich durch folgende Gleichung beschreiben:

$$P(E) = (1 - e^{-E/2})^3.$$

Nutzen Sie das Neumann'sche Rückweisungsverfahren, um die Detektorakzeptanz für die in Teil a) simulierten Signal-Ereignisse zu berücksichtigen. Speichern Sie die Ergebnisse in Form einer *Maske* (True-False-Folge) unter dem Key `AcceptanceMask` in dem `DataFrame`. Stellen Sie das Ergebnis von a) und b) in einem Plot dar. Hinweis: Nutzen Sie eine log-log Darstellung.

c) Energiemessung

Ein realistischer Detektor besitzt nur eine endliche Energieauflösung. Zudem wird die Energie nicht direkt gemessen, sondern mit Hilfe energiekorrelierter Observablen rekonstruiert. Eine solche Observable ist beispielsweise die Anzahl der Photomultiplier, die angesprochen haben (im Folgenden als Hits bezeichnet).

Die Anzahl der Hits lässt sich aus einer Normalverteilung mit folgenden Eigenschaften ziehen:

$$\mathcal{N}(10 \cdot E/\text{TeV}, 2 \cdot E/\text{TeV}).$$

Hierbei bezeichnet $\mathcal{N}(10E/\text{TeV}, 2E/\text{TeV})$ die Normalverteilung mit Mittelwert $10E$ und Standardabweichung $2E$ (E in TeV). Wandeln Sie die Zahl der Hits in eine ganze Zahl um, da nur ganze Anzahlen an Hits auftreten können. Achten Sie ebenfalls darauf, dass für die Anzahl der Hits N nur Werte oberhalb von Null in Frage kommen. Ziehen Sie gegebenenfalls eine neue Zufallszahl, falls diese Bedingung verletzt wird. Simulieren Sie die Anzahl der Hits in Abhängigkeit der zuvor simulierten Energie und speichern Sie die Anzahl unter dem Key `NumberOfHits`. Nutzen Sie hierzu die aus der Vorlesung bekannte Polarmethode, um die Normalverteilung zu realisieren.

d) Ortsmessung

Betrachten Sie im Folgenden einen quadratischen Flächendetektor mit der Kantenlänge 10 Längeneinheiten. Das Signal trifft am Punkt (7,3) auf den Detektor. Die Ortsauflösung ist wiederum energieabhängig. Simulieren Sie die Orte zu den zuvor erzeugten Ereignissen, indem Sie sowohl für die x - als auch für die y -Richtung eine Normalverteilung annehmen. Hierbei ist σ energieabhängig und gegeben durch

$$\sigma = \frac{1}{\log_{10}(N + 1)},$$

wobei N die zuvor bestimmte Anzahl der Hits ist. Achten Sie hier wiederum darauf, dass die gezogenen Ereignisse innerhalb des Detektors liegen (ggf. neue Zufallszahlen ziehen). Speichern Sie die Koordinaten der Orte unter den Keys `x` und `y`. Stellen Sie die erhaltenen Orte in einem zweidimensionalen Histogramm dar.

e) Untergrund MC

Die Zahl der erwarteten Untergrund-Ereignisse ist groß im Verhältnis zum erwarteten Signal. Erzeugen Sie einen neuen `DataFrame` mit den Keys `NumberOfHits`, `x` und `y`.

Simulieren Sie 10^7 Untergrund-Ereignisse mit folgenden Eigenschaften:

- Der Zehner-Logarithmus der Anzahl der Hits folgt einer Normalverteilung mit $\mu = 2$ und $\sigma = 1$.
- Die x - bzw. y -Koordinaten der Ereignisse sind um den Mittelpunkt des Detektors normalverteilt. Hierbei ist $\sigma = 3$.
- Zwischen der x - und der y -Koordinate besteht eine Korrelation von $\rho = 0.5$.

Stellen Sie die Orte der Untergrundereignisse in einem zweidimensionalen und den Logarithmus der Anzahl der Hits in einem eindimensionalen Histogramm dar.

Würfelt man standardnormalverteilte Zufallszahlen x^* bzw. y^* , so ergeben sich die normalverteilten Zufallszahlen x bzw. y mit beliebigem μ , σ und Korrelationskoeffizientem ρ aus der Transformation:

$$x = \sqrt{1 - \rho^2} \cdot \sigma \cdot x^* + \rho \cdot \sigma \cdot y^* + \mu$$
$$y = \sigma \cdot y^* + \mu.$$

Aufgabe 11: *Simulationskette für Neutrinoendetektor*

a) Signal MC

Mit Hilfe der Transformationsmethode werden 10^5 Signalereignisse des Neutrinoflusses

$$\Phi = \Phi_0 \left(\frac{E}{\text{TeV}} \right)^{-\gamma}, \quad (1)$$

mit $\gamma = 2.7$, erzeugt. Für die Transformationsmethode wird der Fluss zunächst von 1 TeV bis E integriert und dann invertiert. Das Ziehen von gleichverteilten Zufallszahlen u im Intervall $[0, \frac{\Phi_0 1 \text{TeV}}{\gamma-1}]$ und Einsetzen in

$$\frac{E}{1 \text{TeV}} = \left(1 + \frac{u(1-\gamma)}{\Phi_0 1 \text{TeV}} \right)^{\frac{1}{1-\gamma}} \quad (2)$$

liefert die gewünschten Signale. Diese werden in einem DataFrame mit dem Key Energy abgespeichert.

Aufgabe 11a

Freitag, 10. November 2017 16:38

$$11.\text{ a}) \quad \Phi = \Phi_0 \left(\frac{E}{\text{TeV}} \right)^{-\gamma}, \quad \gamma = 2, 7$$

$$u = g(E) = \int_{1\text{TeV}}^E \Phi_0 \left(\frac{E'}{\text{TeV}} \right)^{-\gamma} dE' \quad \left(\begin{array}{l} \tilde{E} = \frac{E'}{\text{TeV}} \\ \frac{d\tilde{E}}{dE'} = \frac{1}{\text{TeV}} \end{array} \right)$$

$$g(E) = \int_1^{\frac{E}{\text{TeV}}} \Phi_0 \cdot 1\text{TeV} \quad \tilde{E}^{-\gamma} d\tilde{E}$$

$$= \frac{1}{-\gamma + 1} \Phi_0 \cdot 1\text{TeV} \left(\left(\frac{E}{\text{TeV}} \right)^{-\gamma+1} - 1 \right) = u$$

$$\Rightarrow u + \frac{\Phi_0 \cdot 1\text{TeV}}{1-\gamma} = \frac{\Phi_0 \cdot 1\text{TeV}}{1-\gamma} \left(\frac{E}{\text{TeV}} \right)^{-\gamma+1}$$

$$\Leftrightarrow \left(\frac{E}{\text{TeV}} \right)^{-\gamma+1} = \frac{u(1-\gamma)}{\Phi_0 \cdot 1\text{TeV}} + 1$$

$$\Leftrightarrow \frac{E}{\text{TeV}} = \left[1 + \frac{u(1-\gamma)}{\Phi_0 \cdot 1\text{TeV}} \right]^{\frac{1}{1-\gamma}}, \quad [\Phi_0] = \text{TeV}^3$$

$$u(1\text{TeV}) = \frac{\Phi_0 \cdot 1\text{TeV}}{1-\gamma} (1-1) = 0$$

$$u(\infty) = \frac{\Phi_0 \cdot 1\text{TeV}}{1-\gamma} (0-1) = \frac{\Phi_0 \cdot 1\text{TeV}}{\gamma-1}$$

$$\Rightarrow u \in \left[0, \frac{\Phi_0 \cdot 1\text{TeV}}{\gamma-1} \right] \quad \checkmark$$

Ihr weilt wohl nicht normieren ;)

b) Akzeptanz

Da die Detektionswahrscheinlichkeit energieabhängig ist, müssen die in Aufgabenteil a) erzeugten Signale modifiziert werden. Dazu wird für jede zuvor bestimmte Energie die Detektionswahrscheinlichkeit nach der Gleichung

$$P(E) = (1 - e^{-E/2})^3 \quad (3)$$

bestimmt. Dann werden 10^5 gleichverteilte Zufallszahlen r zwischen 0 und 1 mit `numpy.random.uniform()` gezogen und es wird eine Maske erstellt, indem elementweise verglichen wird, ob $r < P(E)$ gilt. Diese Maske wird in dem DataFrame unter dem Key `AcceptanceMask` gespeichert. In Abbildung 1 sind die Histogramme der erzeugten und der akzeptierten Signale überlagert doppelt-logarithmisch dargestellt. Der sichtbare blaue Bereich entspricht dem Anteil des erzeugten Signals, der verworfen wird.

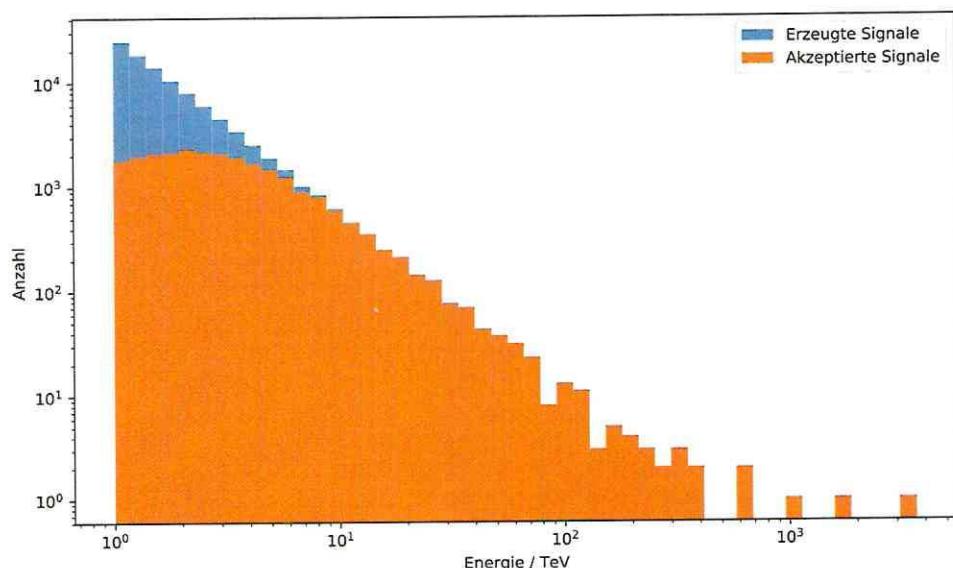


Abbildung 1: Histogramm der erzeugten und akzeptierten Signale. Aufgetragen ist die Energie gegen die Häufigkeit.

c) Energiemessung

In diesem Aufgabenteil wird die Anzahl der Hits, also die Anzahl der Photomultiplier, die angesprochen haben, für die akzeptierten Energien simuliert. Dazu werden mit der Polarmethode ~~eine~~ normalverteilte Zufallszahlen mit dem Mittelwert $10E$ und der Standardabweichung $2E$ generiert. Da die Anzahl der Hits eine ganze, nicht negative Zahl sein muss, werden alle negativen Zufallszahlen verworfen und die Anzahl wird auf eine ganze Zahl gerundet. Die Anzahl Hits wird in dem DataFrame unter dem Key `NumberOfHits` gespeichert. Des Weiteren wird ein Histogramm für die Hits erstellt, das in Abbildung 2 zu sehen ist.

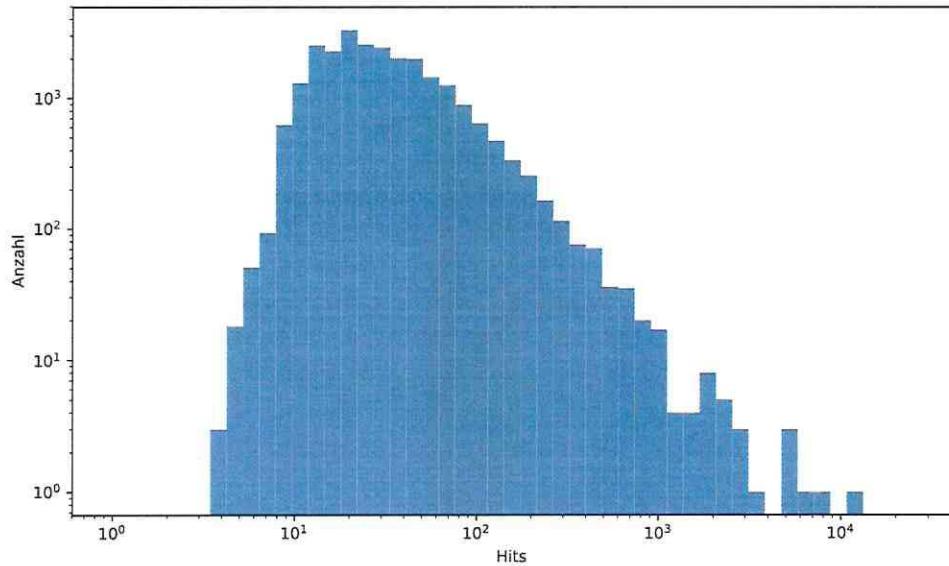


Abbildung 2: Histogramm für die Hits in Abhängigkeit der akzeptierten Energien.



d) Ortsmessung

Es wird ein quadratischer Flächendetektor mit der Kantenlänge 10 Längeneinheiten simuliert, bei dem im Punkt (7,3) das Signal auftrifft. Die energieabhängige Ortsauflösung wird durch eine Normalverteilung in x-Richtung und eine Normalverteilung in y-Richtung mit der Standardabweichung $\sigma = (\log_1 0(N + 1))^{-1}$ simuliert. Es gilt $\mu_x = 7$ und $\mu_y = 3$. Gezogene Ergebnisse, die außerhalb der Fläche des Detektors liegen werden verworfen und neu gezogen. Die Koordinaten der Orte werden mit den Keys x und y im DataFrame gespeichert. Aus den erhaltenen Orten wird ein zweidimensionales Histogramm erstellt, das in Abbildung 3 dargestellt ist.

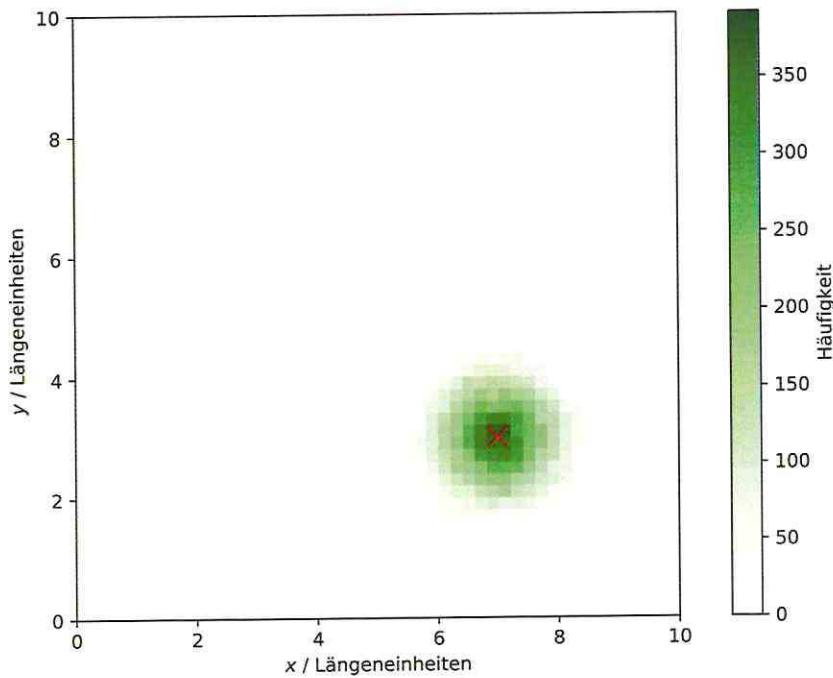


Abbildung 3: Zweidimensionales Histogramm für die Orte, an denen das Signal detektiert wird.

✓

e) Untergrund MC

Zuletzt werden noch 10^7 Untergrundereignisse simuliert. Der Zehner-Logarithmus der Anzahl der Hits folgt dabei einer Normalverteilung mit $\mu = 2$ und $\sigma = 1$, die x- und y-Koordinate sind um den Mittelpunkt (5,5) normalverteilt und zwischen x und y besteht eine Korrelation von $\rho = 0.5$:

$$x = \sqrt{1 - \rho^2} \cdot \sigma \cdot x^* + \rho \cdot \sigma y^* + \mu \quad (4)$$

$$y = \sigma \cdot y^* + \mu. \quad (5)$$

Abbildung 4 zeigt ein zweidimensionales Histogramm der Orte der Untergrundereignisse und der Logarithmus der Anzahl der Hits ist in einem eindimensionalen Histogramm in Abbildung 5 dargestellt.

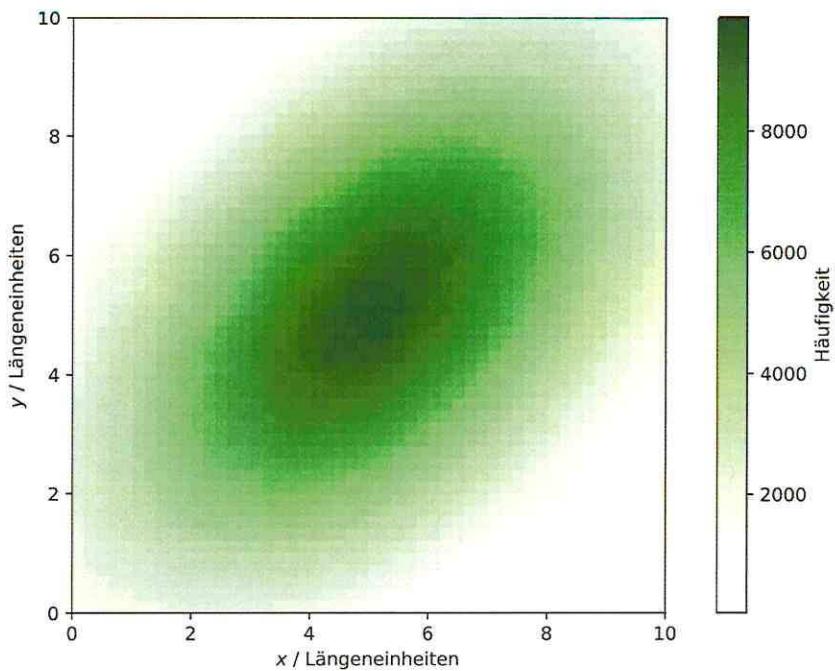


Abbildung 4: Zweidimensionales Histogramm der Orte der Untergrundereignisse.

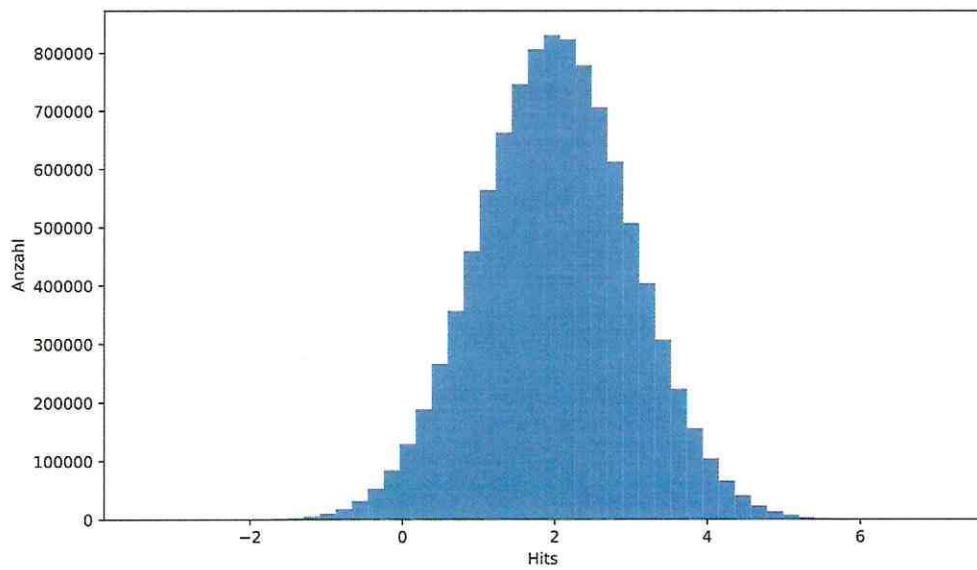


Abbildung 5: Histogramm des Logarithmus der Anzahl der Hits des Untergrunds.

Code für blatt03

Bange, Burkowitz, Harnisch

17. November 2017

```
..../blatt03/Bange_Burkowitz_Harnisch/A11.py

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import os
5 from pylab import rcParams
6
7 rcParams['figure.figsize'] = 10, 5.8
8 rcParams['legend.numpoints'] = 1
9
10 size = 10**5
11 backgroundSize = 10**7
12 # Phi_0 = 1
13
14
15 def aufg11():
16     # Signal erstellen
17     print("Signaldaten erstellen")
18     # 1e5 gleichverteilte Zufallszahlen zwischen u(Emin = 0) und u(Emax = inf)
19     # "-10*epsilon", damit auf keinen Fall bei der Berechnung von
20     # E der Fall 1/0 auftritt
21     u = np.random.uniform(0, 1 / 1.7 - 10 * np.finfo(np.float64).eps,
22                           size=size)
23     signal = pd.DataFrame({"Energy": (1 - 1.7*u)**(-1/1.7)})
24
25     # Akzeptanz
26     print("Akzeptanz berechnen")
27     # Gleichverteilte Zahlen zwischen 0 und 1
28     r = np.random.uniform(0, 1, size=size)
29     # Detektionswahrscheinlichkeit
30     P = (1 - np.exp(-1 * signal["Energy"]/2))**3
31     # Test ob r < P(E) ist
32     signal = signal.assign(AcceptanceMask=np.less(r, P))
33     print("Anteil der akzeptierten Werte:",
34           len(signal["AcceptanceMask"])[signal["AcceptanceMask"]])/size)
35
36     print("Signaldaten und Akzeptanz plotten")
37     plt.yscale('log', nonposy='clip')
38     plt.xscale("log")
39     plt.hist(signal["Energy"],
40             bins=np.logspace(0, np.log10(np.amax(signal["Energy"])), 50),
41             label="Erzeugte Signale")
42     # signal["Energy"][signal["AcceptanceMask"]] sind die signal["Energy"]
43     # Werte, bei denen signal["AcceptanceMask"] == True ist
44     plt.hist(signal["Energy"][signal["AcceptanceMask"]],
45             bins=np.logspace(0, np.log10(np.amax(signal["Energy"])), 50),
46             label="Akzeptierte Signale")
47     plt.legend(loc="best")
48     plt.xlabel("Energie / TeV")
49     plt.ylabel("Anzahl")
50     plt.savefig("Energie.pdf")
51
52     # Energiemessung
53     def Normalverteilung(sigma, mu, low=-np.inf, high=np.inf):
54         while(True):
55             v1 = np.random.uniform(-1, 1)
56             v2 = np.random.uniform(-1, 1)
57             s = v1**2 + v2**2
```

```

58         if(s >= 1):
59             continue
60         x1 = sigma*v1*np.sqrt(-2*np.log(s)/s) + mu
61         if((x1 < low) | (x1 > high)):
62             x2 = sigma*v2*np.sqrt(-2*np.log(s)/s) + mu
63             if((x2 < low) | (x2 > high)):
64                 continue
65             return x2
66         return x1
67
68 print("Anzahl der Hits berechnen")
69 numberOfHits = np.zeros(size)
70 i = 0
71 while(i < size):
72     if(~signal["AcceptanceMask"][i]):
73         i += 1
74         continue
75     numberOfHits[i] = np.round(Normalverteilung(2*signal["Energy"][i],
76                                              10*signal["Energy"][i],
77                                              low=0))
78     i += 1
79
80 signal = signal.assign(numberOfHits=numberOfHits)
81
82 print("Hits plotten")
83 plt.clf()
84 plt.yscale('log', nonposy='clip')
85 plt.xscale("log")
86 plt.hist(signal["numberOfHits"][(signal["AcceptanceMask"])],
87          bins=np.logspace(0, np.log10(np.amax(signal["numberOfHits"]))),
88          50)
89 plt.xlabel("Hits")
90 plt.ylabel("Anzahl")
91 plt.savefig("Hits.pdf")
92
93 # Ortsmessung
94 print("Orte berechnen")
95 x = np.zeros(size)
96 y = np.zeros(size)
97 i = 0
98 while(i < size):
99     if(~signal["AcceptanceMask"][i]):
100        i += 1
101        continue
102     sigma = 1/(np.log10(signal["numberOfHits"][i] + 1))
103     x[i] = Normalverteilung(sigma, 7, low=0, high=10)
104     y[i] = Normalverteilung(sigma, 3, low=0, high=10)
105     i += 1
106
107 signal = signal.assign(x=x, y=y)
108
109 print("Orte plotten")
110 plt.clf()
111 heatmap, xedges, yedges = \
112     np.histogram2d(signal["x"][(signal["AcceptanceMask"])],
113                     signal["y"][(signal["AcceptanceMask"])],
114                     bins=50, range=[[0, 10], [0, 10]])
115 extent = [xedges[0], xedges[-1], yedges[0], yedges[-1]]
116 plt.imshow(heatmap.T, extent=extent, origin='lower', cmap="Greens")
117 plt.colorbar(label="Häufigkeit")
118 plt.xlabel(r"$\$x\$ / \text{Längeneinheiten}$")
119 plt.ylabel(r"$\$y\$ / \text{Längeneinheiten}$")
120 plt.plot(7, 3, "rx", ms=10)
121 plt.savefig("Orte.pdf", bbox_inches='tight')
122
123 signal.to_hdf("NeutrinoMC.hdf5", key="Signal")
124
125 # Background
126 print("Background Daten erstellen...")
127 os.system('setterm -cursor off')
128 # numberOfHits = np.zeros(backgroundSize)

```

```
200
201
202 #----- Main -----
203 if __name__ == '__main__':
204     np.random.seed(1234)
205     aufg11()
```

Aufgabe 13: *Trennende Geraden*

5 P.

Gegeben seien die Populationen P_0 und P_1 aus der Aufgabe „Zwei Populationen“. Nutzen Sie das dort erstellt HDF-File für diese Aufgabe. (Sie finden die Datei ebenfalls im Moodle.) Außerdem seien die Projektionsgeraden

$$g_1(x) = 0 \quad (1)$$

$$g_2(x) = -\frac{3}{4}x \quad (2)$$

$$g_3(x) = -\frac{5}{4}x \quad . \quad (3)$$

gegeben.

- Stellen Sie die beiden Populationen zusammen in einem zweidimensionalen Scatterplot dar und zeichnen Sie die drei Projektionsgeraden ein. Im Folgenden werden diese beiden Populationen mit P_0 und P_1 bezeichnet.
- Projizieren Sie die Punkte aus Population P_0 und P_1 jeweils auf die Geraden g_1 , g_2 und g_3 . Bestimmen und normieren Sie vorher den Projektionsvektor und wählen sie das Vorzeichen so, dass die projizierte Population P_0 rechts (zu größeren Werten) von P_1 liegt. Stellen Sie die projizierten Populationen P_0 und P_1 für jede Projektion in einem eigenen, eindimensionalen Histogramm dar.
- Betrachten Sie P_0 als Signal und P_1 als Untergrund. Berechnen Sie die Effizienz und die Reinheit des Signals als Funktion eines Schnittes λ_{cut} in den projizierten Räumen und stellen Sie die Ergebnisse für jede Projektion in einem eigenen Plot dar.

Aufgabe 14: *Fisher-Diskriminante: Per Hand*

5 P.

Führen Sie eine lineare Diskriminanzanalyse nach Fisher per Hand durch.

Population 0: (2;2;1) (2;3;2) (2;1;2) (1;2;0) (3;2;0)

Population 1: (2,5;2,5;0) (2,5;1,5;0) (4;2;0) (5,5;2,5;0) (5,5;1,5;0)

- Berechnen Sie die Mittelwerte $\vec{\mu}$ und Streumatrizen S_i , sowie die Streuung innerhalb der Klassen und zwischen den Klassen (S_W und S_B).
- Berechnen Sie die Eigenwerte und Eigenvektoren der Matrix $S_W^{-1} \cdot S_B$.
- Wie lautet die Projektion $\vec{\lambda}$? Verifizieren Sie dies, indem Sie $S_W^{-1}(\vec{\mu}_0 - \vec{\mu}_1)$ berechnen.

Aufgabe 12

a)

Die untenstehenden Rechnungen zeigen, dass die Population P_1 ebenfalls eine 2D Normalverteilung mit den Parametern

$$\begin{aligned}\mu_y' &= a + b\mu_x = 3,1, \\ \sigma_y' &= \sqrt{\sigma_{y|x}^2 + b^2\sigma_x^2} = \frac{\sqrt{541}}{10} \approx 2,326 \text{ und} \\ \rho &= b \frac{\sigma_x}{\sigma_y} \approx 0,90286\end{aligned}$$

✓

beschreibt.

b)

Beide Populationen werden in einem zweidimensionalen Scatter-Plot dargestellt. Dazu werden Gaußverteilungen mit 10000 Werten pro Population und den Parametern

$$\begin{aligned}P_0 : \mu_x &= 0, \mu_x = 3, \sigma_x = 3.5, \sigma_y = 2.6 \text{ und } \rho = 0.9 \\ P_1 : \mu_x &= 6, \mu_x = 3.1, \sigma_x = 3.5, \sigma_y = \frac{\sqrt{541}}{10} \text{ und } \rho \approx 0.90286\end{aligned}$$

erzeugt und gemeinsam in Abbildung 1 dargestellt.

5/5

A 12 a)

$$f_{\text{GauB}} = \frac{1}{\sigma \sqrt{2\pi}} \exp \left[-\frac{1}{2} \left(\frac{x-\mu}{\sigma} \right)^2 \right]$$

$$f_{2D} = \frac{1}{2\pi \sigma_x \sigma_y \sqrt{1-\beta^2}} \exp \left[-\frac{1}{2(1-\beta^2)} \left(\left(\frac{x-\mu_x}{\sigma_x} \right)^2 + \left(\frac{y-\mu_y}{\sigma_y} \right)^2 - 2\beta \left(\frac{x-\mu_x}{\sigma_x} \right) \left(\frac{y-\mu_y}{\sigma_y} \right) \right) \right]$$

$$\Rightarrow f = \frac{1}{2\pi \sigma_x \sigma_{y|x}} \exp \left[-\frac{1}{2} \left(\frac{x-\mu_x}{\sigma_x} \right)^2 \right] \exp \left[-\frac{1}{2} \left(\frac{y-\mu_{y|x}}{\sigma_{y|x}} \right)^2 \right]$$

$$= \frac{1}{2\pi \sigma_x \sigma_{y|x}} \exp \left[-\frac{1}{2} \left(\frac{x^2 + \mu_x^2 - 2x\mu_x}{\sigma_x^2} + \frac{y^2 + \mu_{y|x}^2 - 2y\mu_{y|x}}{\sigma_{y|x}^2} \right) \right]$$

$$= \frac{1}{2\pi \sigma_x \sigma_{y|x}} \exp \left[-\frac{1}{2} \left(\left(\frac{x-\mu_x}{\sigma_x} \right)^2 + \frac{y^2 + a^2 + 2abx + b^2x^2 - 2ya - 2yb^2}{\sigma_{y|x}^2} \right) \right]$$

$$= \underbrace{\frac{1}{2\pi \sigma_x \sigma_{y|x}}}_{\Rightarrow \sigma_{y|x} = \sigma_y \sqrt{1-\beta^2}} \exp \left[-\frac{1}{2} \left(\left(\frac{x-\mu_x}{\sigma_x} \right)^2 + \left(\frac{y-a}{\sigma_{y|x}} \right)^2 + \frac{2abx + b^2x^2 - 2yb^2}{\sigma_{y|x}^2} \right) \right]$$

$$\Rightarrow \sigma_{y|x} = \sigma_y \sqrt{1-\beta^2}$$

$$= \underbrace{\frac{1}{2\pi \sigma_x \sigma_y \sqrt{1-\beta^2}} \exp \left[-\frac{1}{2} \left(\left(\frac{x-\mu_x}{\sigma_x} \right)^2 + \frac{1}{(1-\beta^2)} \left(\frac{y-a}{\sigma_y} \right)^2 + \frac{1}{(1-\beta^2)} \frac{2abx + b^2x^2 - 2yb^2}{\sigma_y^2} \right) \right]}_{=: A}$$

$$A = \frac{x^2 + \mu_x^2 - 2x\mu_x}{\sigma_x^2} + \frac{1}{(1-\beta^2)} \left(\frac{y-a}{\sigma_y} \right)^2 + \frac{1}{(1-\beta^2)} \frac{2abx + b^2x^2 - 2yb^2}{\sigma_y^2}$$

$$= \frac{1}{1-\beta^2} \left(\left(\frac{y-a}{\sigma_y} \right)^2 + \frac{x^2 + \mu_x^2 - 2x\mu_x}{\sigma_x^2} \right)$$

$$- \frac{\beta^2 x^2 \sigma_y^2 + \beta^2 \mu_x^2 \sigma_y^2 - 2\beta^2 \mu_x \sigma_y^2 x - 2abx \sigma_x^2 - b^2 x^2 \sigma_x^2 + 2yb^2 x \sigma_x^2}{\sigma_x^2 \sigma_y^2}$$

$$A = \frac{1}{1-s^2} \left(\left(\frac{x-\mu_x}{\sigma_x} \right)^2 + \frac{\gamma^2 + \alpha^2 - 2\gamma\alpha}{\sigma_y'^2} \right.$$

$$\left. - \frac{s^2 x^2 \sigma_y'^2 + s^2 \mu_x^2 \sigma_y'^2 - 2s^2 \mu_x \sigma_y'^2 x - 2ab x \sigma_x^2 - b^2 x^2 \sigma_x^2 + 2\gamma b x \sigma_x^2}{\sigma_x^2 \sigma_y'^2} \right)$$

$$=: \frac{1}{1-s^2} \left(\left(\frac{x-\mu_x}{\sigma_x} \right)^2 + B \right)$$

$$BO1, 17g) \quad \mu_{Y|X} = \mu'_y + s \frac{\sigma'_y}{\sigma_x} (x - \mu_x)$$

$$\text{Bei uns: } \mu_{Y|X} = a + b x$$

$$\Rightarrow a = \mu'_y - s \frac{\sigma'_y}{\sigma_x} \mu_x, \quad b = s \frac{\sigma'_y}{\sigma_x}$$

$$\alpha^2 = \mu_y'^2 + s^2 \frac{\sigma_y'^2}{\sigma_x^2} \mu_x^2 - 2\mu'_y s \frac{\sigma'_y}{\sigma_x} \mu_x$$

$$\Rightarrow B = \frac{1}{\sigma_x^2 \sigma_y'^2} \left(\underbrace{y^2 \sigma_x^2}_{\text{green}} + \underbrace{\mu_y'^2 \sigma_x^2}_{\text{green}} + \underbrace{s^2 \sigma_y'^2 \mu_x^2}_{\text{red}} - 2\mu'_y s \sigma'_y \sigma_x \mu_x \right.$$

$$- \underbrace{2\gamma \mu'_y \sigma_x^2}_{\text{green}} + \underbrace{2\gamma s \sigma'_y \sigma_x \mu_x}_{\text{green}} - \underbrace{s^2 x^2 \sigma_y'^2}_{\text{orange}} - \underbrace{s^2 \mu_x^2 \sigma_y'^2}_{\text{red}}$$

$$+ \underbrace{2s^2 \mu_x \sigma_y'^2 x}_{\text{yellow}} + \underbrace{2\mu'_y s \sigma'_y \sigma_x x}_{\text{yellow}} - \underbrace{2s^2 \sigma_y'^2 \mu_x x}_{\text{yellow}}$$

$$\left. + \underbrace{s^2 \sigma_y'^2 x^2}_{\text{orange}} - 2\gamma s \sigma'_y \sigma_x x \right)$$

$$= \underbrace{\left(\frac{x-\mu'_y}{\sigma'_y} \right)^2}_{\text{green}} + \underbrace{0}_{\text{red}} + \underbrace{0}_{\text{red}} + \underbrace{0}_{\text{yellow}}$$

$$- \frac{2s}{\sigma_x \sigma_y'} \left(\mu'_y \mu_x - \gamma \mu_x - \mu'_y x + x \gamma \right)$$

$$= \left(\frac{y-\mu'_y}{\sigma'_y} \right)^2 - 2s \left(\frac{\mu_x - x}{\sigma_x} \right) \left(\frac{\mu'_y - y}{\sigma'_y} \right)$$

$$\Rightarrow f = \frac{1}{2\pi \sigma_x \sigma_y' \sqrt{1-s^2}} \exp \left[-\frac{1}{2(1-s^2)} \left(\left(\frac{x-\mu_x}{\sigma_x} \right)^2 + \left(\frac{y-\mu'_y}{\sigma'_y} \right)^2 \right. \right.$$

$$\left. \left. - 2s \left(\frac{x-\mu_x}{\sigma_x} \right) \left(\frac{y-\mu'_y}{\sigma'_y} \right) \right) \right]$$

$$\sigma_{y|x} = \sigma_y^1 \sqrt{1 - \rho^2}$$

$$\rho = b \frac{\sigma_x}{\sigma_y}$$

$$\mu_{y|x} = a + b x$$

$$\mu_{y|x} = \mu_y^1 + \rho \frac{\sigma_y^1}{\sigma_x} (x - \mu_x)$$

$$\Rightarrow \mu_y^1 = a + b x - b(x - \mu_x) = a + b \mu_x = 3,1$$

$$\sigma_{y|x} = \sigma_y^1 \sqrt{1 - b^2 \frac{\sigma_x^2}{\sigma_y^2}}$$

$$\Rightarrow \sigma_{y|x}^2 = \sigma_y^{12} \left(1 - b^2 \frac{\sigma_x^2}{\sigma_y^2} \right) = \sigma_y^{12} - b^2 \sigma_x^2$$

$$\Rightarrow \sigma_y^1 = \sqrt{\sigma_{y|x}^2 + b^2 \sigma_x^2} = \sqrt{54 \cdot 1}$$

$$\Rightarrow \rho = b \frac{\sigma_x}{\sqrt{\sigma_{y|x}^2 + b^2 \sigma_x^2}} \approx 0,90286$$

Aufgabe 13

a)

Mit den in Aufgabe 12 erzeugten Datenpunkten für die Populationen P_0 und P_1 , die in einem HDF-File gespeichert wurden, wird erneut ein zweidimensionaler Scatter-Plot erstellt, in den zusätzlich die Projektionsgeraden

$$\begin{aligned}g_1(x) &= 0 \\g_2(x) &= -\frac{3}{4}x \\g_3(x) &= -\frac{5}{4}x\end{aligned}$$

eingezzeichnet werden. Der Plot ist in Abbildung 2 dargestellt.

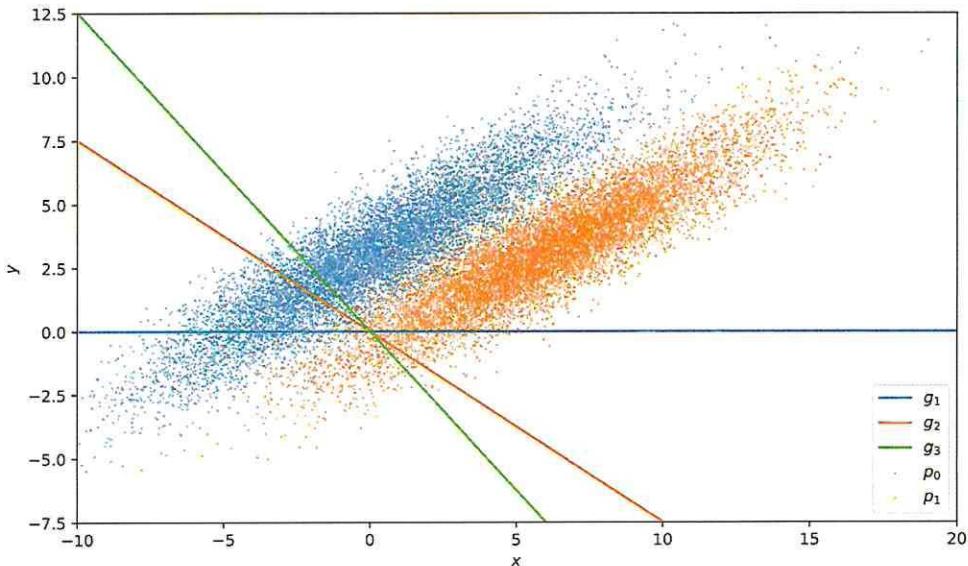


Abbildung 2: Zweidimensionaler Scatter-Plot für die Populationen P_0 und P_1 und die drei Projektionsgeraden.

b)

Zunächst werden die normierten Projektionsvektoren für die drei Geraden bestimmt. Die Projektionsvektoren können dabei direkt aus der Steigung der Geraden abgelesen werden. Das Vorzeichen wird jeweils so gewählt, dass P_0 nach der Projektion rechts von P_1 liegt.

A 13

$$g_1(x) = 0$$

$$\Rightarrow \vec{P}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$g_2(x) = -\frac{3}{4}x$$

$$\Rightarrow \vec{P}_2 = \frac{1}{\sqrt{16+9}} \begin{pmatrix} 4 \\ -3 \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 4 \\ -3 \end{pmatrix}$$

5/5

$$g_3(x) = -\frac{5}{4}x$$

$$\Rightarrow \vec{P}_3 = \frac{1}{\sqrt{25+16}} \begin{pmatrix} 4 \\ -5 \end{pmatrix} = \frac{1}{\sqrt{41}} \begin{pmatrix} 4 \\ -5 \end{pmatrix}$$

Für die Projektion wird für jeden Datenpunkt \vec{x}_i das Skalarprodukt mit den Projektionsvektoren \vec{p}_j gebildet:

$$x_{i,j,\text{projiziert}} = \vec{x}_i \cdot \vec{p}_j. \quad (1)$$

Die projizierten Datenpunkte sind für jeden Projektionsvektor getrennt in den Histogrammen in Abbildung 3, 4 und 5 dargestellt.

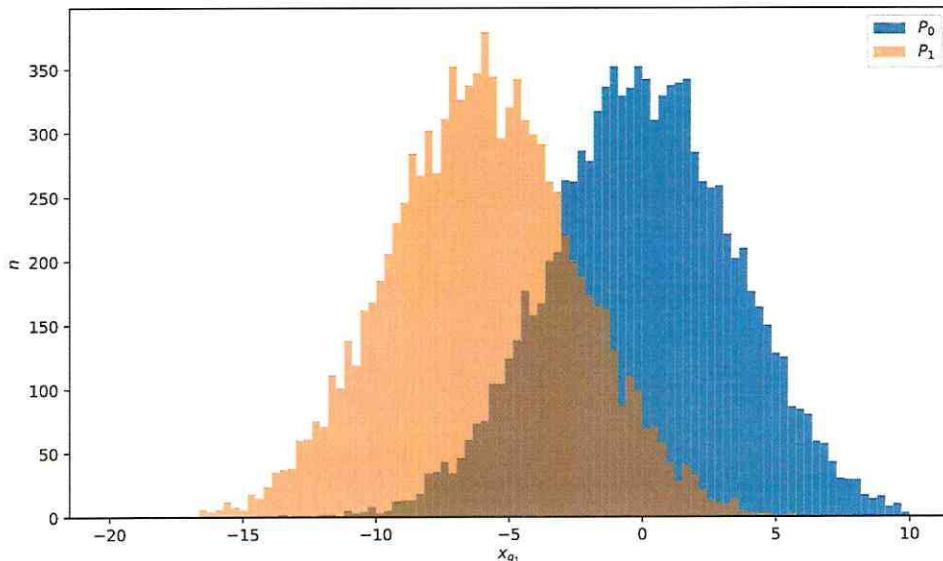


Abbildung 3: Histogramm für die Projektion der Populationen P_0 und P_1 auf die Gerade g_1 .

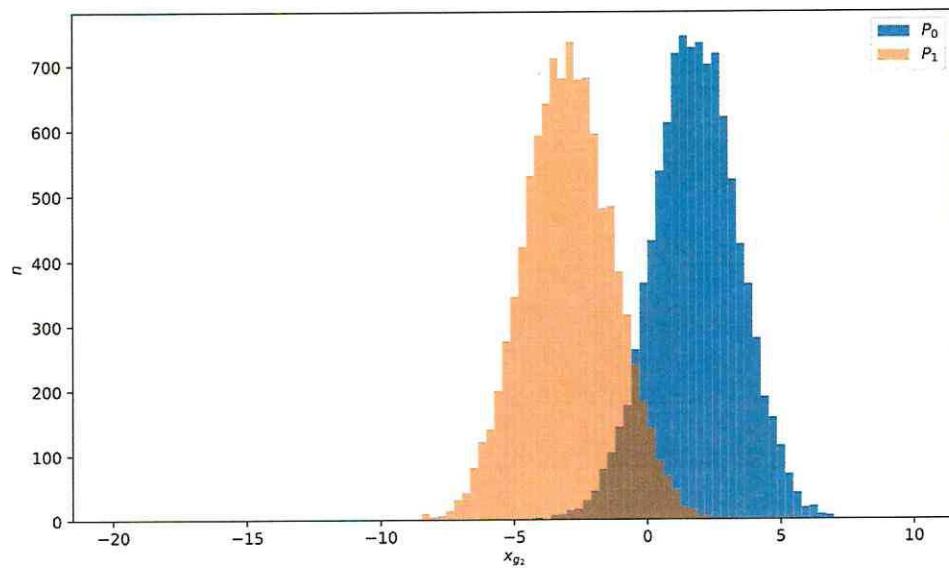


Abbildung 4: Histogramm für die Projektion der Populationen P_0 und P_1 auf die Gerade g_2 .

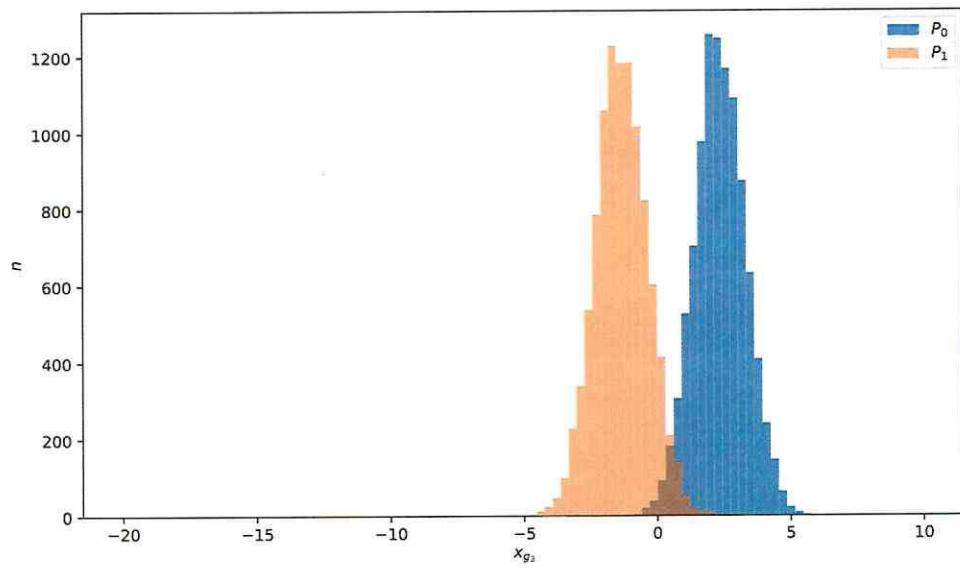


Abbildung 5: Histogramm für die Projektion der Populationen P_0 und P_1 auf die Gerade g_3 .

c)

Die Population P_0 wird als Signal betrachtet, P_1 stellt den Untergrund dar. Abhängig vom gewählten Schnitt λ_{cut} wird für jede Projektion die Effizienz, die Reinheit und die Genauigkeit bestimmt. Dazu werden die projizierten Datenpunkte sortiert und die Anzahl der true positiv (tp), false negative (fn), der true negative (tn) und der false positive (fp) wird abhängig vom Schnitt gezählt.

$$\begin{aligned}\text{Effizienz} &= \frac{\text{tp}}{\text{tp} + \text{fp}} \\ \text{Reinheit} &= \frac{\text{tp}}{\text{tp} + \text{fn}} \\ \text{Genauigkeit} &= \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}}\end{aligned}$$

Die Effizienz, Reinheit und Genauigkeit sind für jede der drei Projektionen in den Abbildungen 6, 7 und 8 dargestellt.

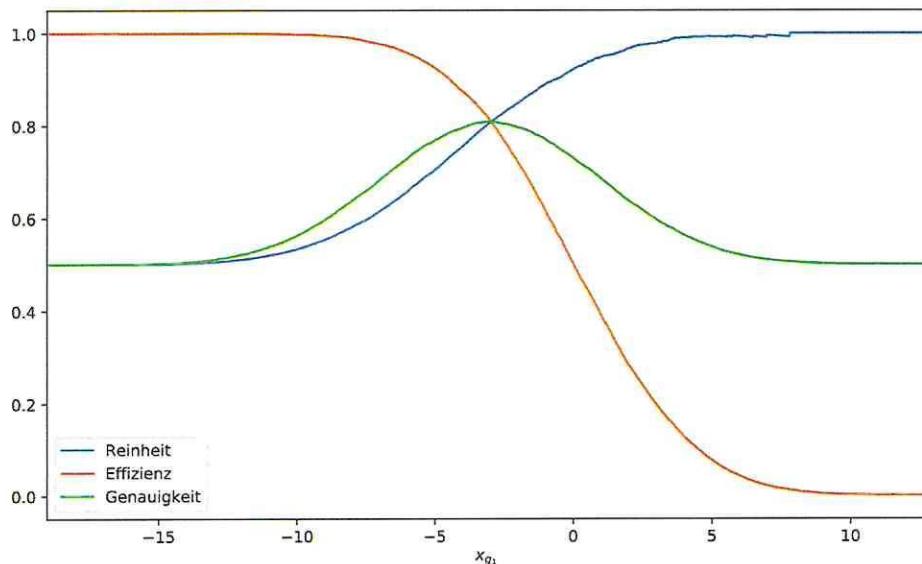


Abbildung 6: Effizienz, Reinheit und Genauigkeit in Abhängigkeit von λ_{cut} für die Projektion der Populationen P_0 und P_1 auf die Gerade g_1 .

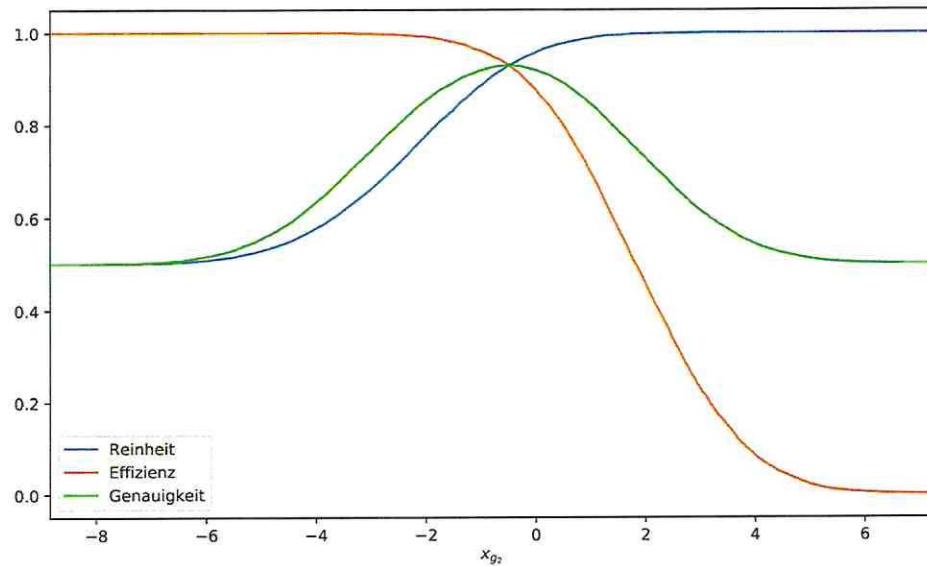


Abbildung 7: Effizienz, Reinheit und Genauigkeit in Abhangigkeit von λ_{cut} fur die Projektion der Populationen P_0 und P_1 auf die Gerade g_2 .

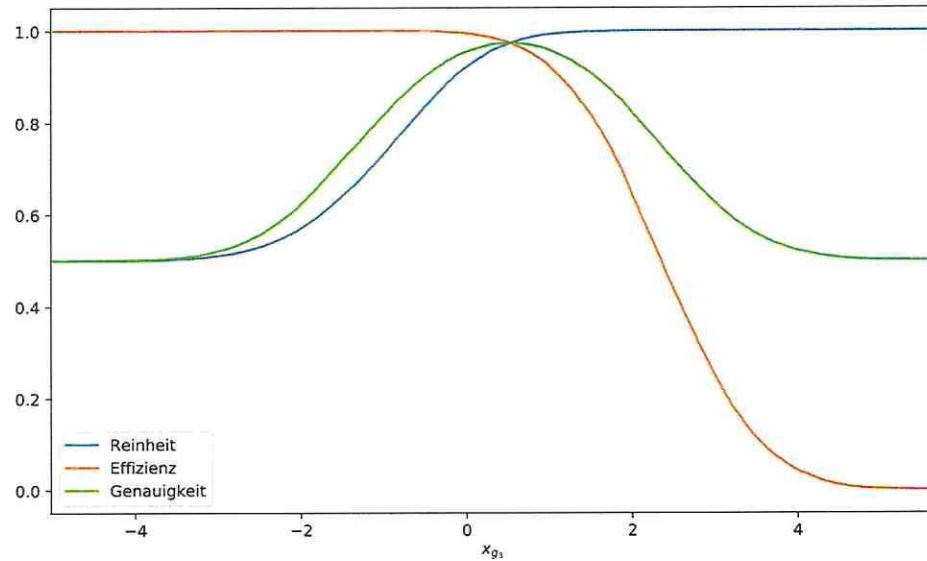


Abbildung 8: Effizienz, Reinheit und Genauigkeit in Abhangigkeit von λ_{cut} fur die Projektion der Populationen P_0 und P_1 auf die Gerade g_3 .

A 14

Population 0: $(2; 2; 1) \quad (2; 3; 2) \quad (2; 1; 2)$
 $(1; 2; 0) \quad (3; 2; 0)$

Population 1: $(2,5; 2,5; 0) \quad (2,5; 1,5; 0) \quad (4; 2; 0)$
 $(5,5; 2,5; 0) \quad (5,5; 1,5; 0)$

a) $\mu_0 = \frac{1}{5} \begin{pmatrix} 10 \\ 10 \\ 5 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}$

$\mu_1 = \frac{1}{5} \begin{pmatrix} 20 \\ 10 \\ 0 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 0 \end{pmatrix}$

$S_w = \sum_j S_j$

$S_j = \sum_{i=0}^4 (\vec{x}_i - \vec{\mu}_j)(\vec{x}_i - \vec{\mu}_j)^T$

$S_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} (0 \ 0 \ 0) + \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} (0 \ 1 \ 1) + \begin{pmatrix} 0 \\ -1 \\ -1 \end{pmatrix} (0 \ -1 \ -1)$

$+ \begin{pmatrix} -1 \\ 0 \\ -1 \end{pmatrix} (-1 \ 0 \ -1) + \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} (1 \ 0 \ -1)$

$= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & -1 & -1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{pmatrix}$

$+ \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}$

$$S_1 = \begin{pmatrix} -1,5 \\ 0,5 \\ 0 \end{pmatrix} (-1,5 \ 0,5 \ 0) + \begin{pmatrix} -1,5 \\ -0,5 \\ 0 \end{pmatrix} (-1,5 \ -0,5 \ 0)$$

$$+ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} (0 \ 0 \ 0) + \begin{pmatrix} 1,5 \\ 0,5 \\ 0 \end{pmatrix} (1,5 \ 0,5 \ 0)$$

$$+ \begin{pmatrix} 1,5 \\ -0,5 \\ 0 \end{pmatrix} (1,5 \ -0,5 \ 0)$$

$$= \begin{pmatrix} 2,25 & -0,75 & 0 \\ -0,75 & 0,25 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 2,25 & 0,75 & 0 \\ 0,75 & 0,25 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$+ \begin{pmatrix} 2,25 & 0,75 & 0 \\ 0,75 & 0,25 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 2,25 & -0,75 & 0 \\ -0,75 & 0,25 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 9 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \checkmark$$

$$-\lambda \left[\left(\frac{4}{m} - \lambda \right) \left(\frac{7}{4} - \lambda \right) - \frac{1}{m} \right] = 0$$

$$\Rightarrow \lambda = 0$$

$$\checkmark \frac{1}{m} - \frac{4}{m} \lambda - \frac{1}{4} \lambda + \lambda^2 - \frac{1}{m} = 0$$

$\Rightarrow \lambda = 0$ ist doppelte NS

$$\checkmark \lambda = \frac{4}{m} + \frac{1}{4} = \frac{16+m}{44} = \frac{23}{99} \checkmark$$

$$\underline{\lambda = 0}$$

$$\begin{pmatrix} \frac{4}{m} & 0 & -\frac{2}{m} \\ 0 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{4} \end{pmatrix} mI + 8 \underline{III}$$

$$\begin{pmatrix} \frac{4}{m} & 0 & -\frac{2}{m} \\ 0 & 0 & 0 \\ 0 & 0 & -2+2 \end{pmatrix}$$

$$\Rightarrow \frac{4}{m} v_1 - \frac{2}{m} v_3 = 0 \Leftrightarrow v_1 = \frac{1}{2} v_3$$

$$\Rightarrow EV: \vec{u}_1 = \frac{1}{\sqrt{5}} \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix} \quad \vec{u}_2 = \frac{1}{\sqrt{6}} \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}$$

$$\text{allgemein: } \left\{ \vec{u}_{1,2} \in \mathbb{R}^3 \mid \vec{u}_{1,2} = \begin{pmatrix} a \\ b \\ 2a \end{pmatrix} \right\}$$

$$D(\bar{u}_{1,2}) = \frac{(a \ b \ 2a) \begin{pmatrix} 4 & 0 & -2 \\ 0 & 0 & 0 \\ -2 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ 2a \end{pmatrix}}{(a \ b \ 2a) \begin{pmatrix} 11 & 3 & 4 \end{pmatrix} \begin{pmatrix} a \\ b \\ 2a \end{pmatrix}}$$

$$= \frac{(4a - 4a \ 0 \ -2a + 2a) \begin{pmatrix} a \\ b \\ 2a \end{pmatrix}}{(11a \ 3b \ 8a) \begin{pmatrix} a \\ b \\ 2a \end{pmatrix}} = 0$$

$$D(\bar{u}_3) = \frac{(a \ 0 \ \frac{11}{8}a) \begin{pmatrix} 4 & 0 & -2 \\ 0 & 0 & 0 \\ -2 & 0 & 1 \end{pmatrix} \begin{pmatrix} -a \\ 0 \\ \frac{11}{8}a \end{pmatrix}}{(a \ 0 \ \frac{11}{8}a) \begin{pmatrix} 11 & 3 & 4 \end{pmatrix} \begin{pmatrix} -a \\ 0 \\ \frac{11}{8}a \end{pmatrix}}$$

$$= \frac{(-4a + \frac{11}{4}a \ 0 \ -2a + \frac{11}{8}a) \begin{pmatrix} -a \\ 0 \\ \frac{11}{8}a \end{pmatrix}}{(-11a \ 0 \ \frac{11}{2}a) \begin{pmatrix} -a \\ 0 \\ \frac{11}{8}a \end{pmatrix}}$$

$$= \frac{4 \cdot \frac{11}{4} - \frac{11}{4} + \frac{121}{64}}{11 + \frac{121}{16}} = \frac{25}{1188} > 0$$

\Rightarrow Die Projektion, die $D(\bar{\lambda})$ maximiert ist (normiert): $\bar{\lambda} = \frac{1}{\sqrt{185}} \begin{pmatrix} -8 \\ 0 \\ 11 \end{pmatrix}$

Verifizierung:

$$\tilde{s}_{\bar{w}}^{\bar{\lambda}} (\bar{\mu}_0 - \bar{\mu}_1) = \begin{pmatrix} \bar{\lambda} \\ \bar{\lambda} \\ \bar{\lambda} \end{pmatrix} \begin{pmatrix} -2 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -\frac{2}{\bar{\lambda}} \\ 0 \\ \frac{1}{\bar{\lambda}} \end{pmatrix} = \sqrt{185} \cdot \frac{1}{88} \bar{\lambda}$$

Aufgabe 15

- a) Siehe unten.
- b) Siehe Code.
- c)

Die mit dem Metropolis-Algorithmus erzeugten Zufallszahlen werden in Abbildung 9 als Histogramm dargestellt und mit einer Gaußverteilung mit $\mu = -3$ und $\sigma = 2$ verglichen. Es zeigt sich eine gute Übereinstimmung. Auffällig ist, dass rechts vom Mittelwert der Verteilung deutlich häufiger Zahlen mit einer sehr geringen Wahrscheinlichkeitsdichte gezogen werden, als links vom Mittelwert. Dies liegt an dem gewählten Startwert $x_0 = 15$ und der Anlaufphase des Algorithmus (vgl. d)).

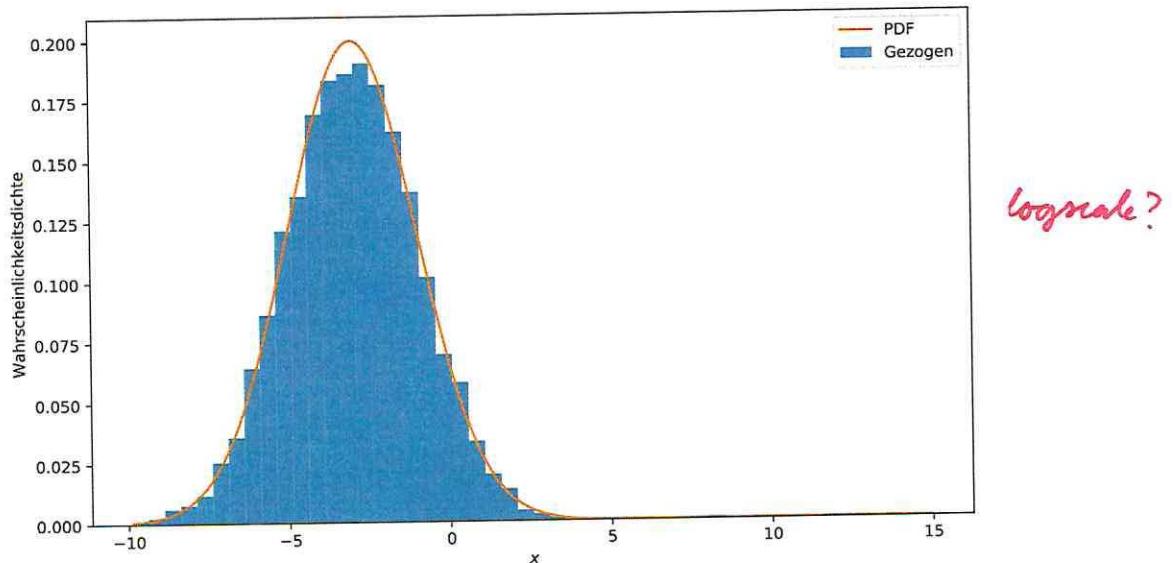


Abbildung 9: Histogramm von 10000 mit dem Metropolis-Algorithmus erzeugten Zufallszahlen ($x_0 = 15$, stepsize = 2, $\mu = -3$ und $\sigma = 2$) und Vergleich mit der analytischen Form der Gaußverteilung.

d)

Die erzeugten Zufallszahlen werden gegen die Iteration, in der sie erzeugt wurden, aufgetragen. Der sogenannte Trace-Plot ist in Abbildung 10 dargestellt. In Abbildung 11 ist der Ausschnitt, der die Anlaufphase enthält, vergrößert dargestellt. In dem Ausschnitt ist deutlich zu erkennen, dass der Algorithmus mit dem gewählten Startparameter von $x_0 = 15$ etwa 50 Iterationen benötigt, um in die Nähe des Mittelwerts zu gelangen (Anlaufphase). Daher sollten die Zufallszahlen, die während der Anlaufphase erzeugt wurden, verworfen werden. Da 10000 Zufallszahlen gezogen wurden, ist es gut möglich, die ersten 500 Iterationen zu verwerfen, um einen gewissen Sicherheitsabstand (Faktor 10) zur Anlaufphase sicherzustellen.

5/5

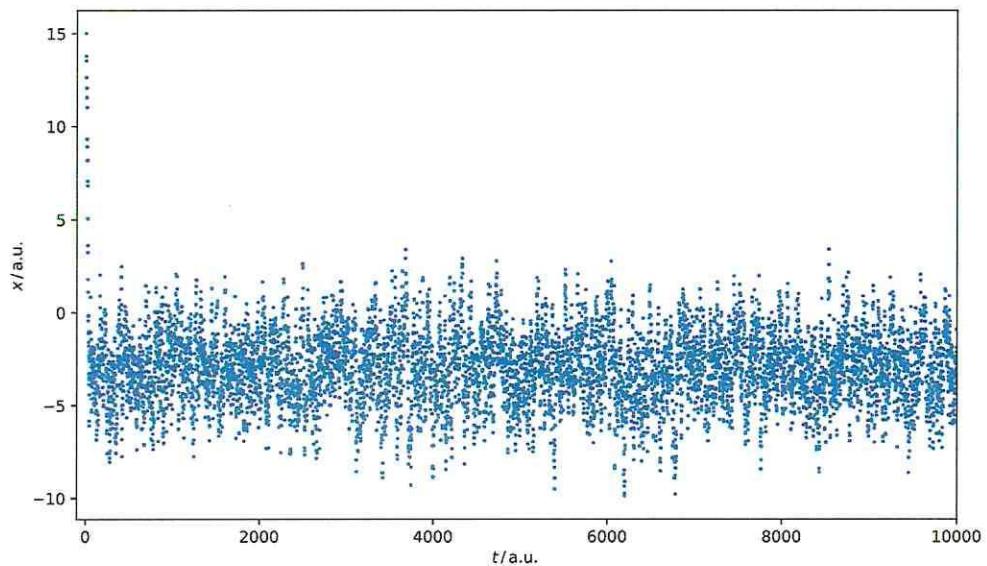


Abbildung 10: Trace-Plot für die erzeugten Zufallszahlen.

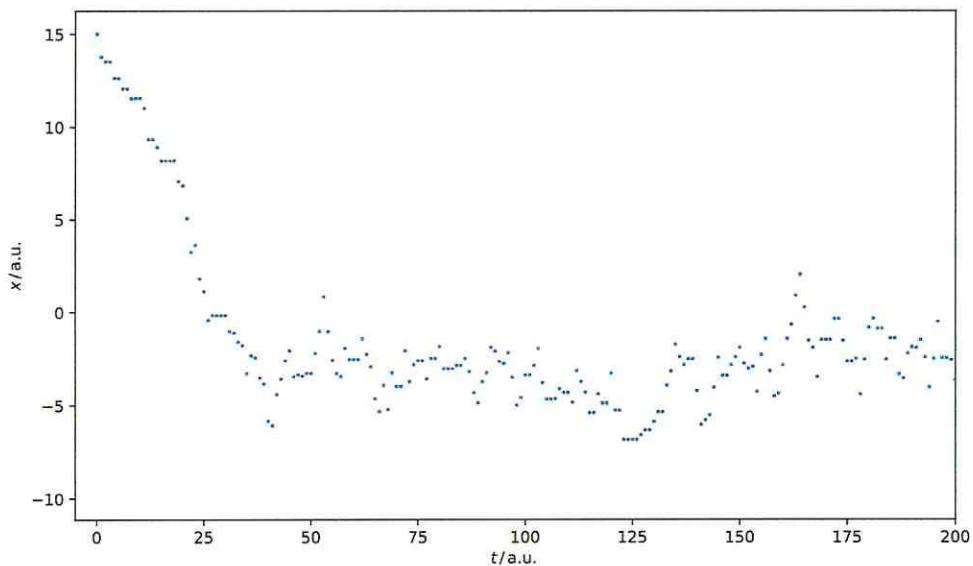


Abbildung 11: Trace-Plot (Ausschnitt der Anlaufphase).

A 15

Metropolis-Hastings:

$$M_{i \rightarrow j} = \min \left(1, \frac{f(x_i)}{f(x_j)} \cdot \frac{g(x_j | x_i)}{g(x_i | x_j)} \right)$$

$\frac{g(x_j | x_i)}{g(x_i | x_j)}$ $\rightarrow 1 \Rightarrow$ Metropolis-Hastings
 \rightarrow Metropolis

g sei gaussförmig (BZL, A7):

$$\frac{g(x|y)}{g(y|x)} = \frac{\sigma_y}{\sigma_x} \exp \left[\frac{1}{2} (u_x^2 - u_y^2) \right] = \frac{\sigma_y}{\sigma_x} \exp \left[\frac{1}{2} \left(\left(\frac{x - \mu_x}{\sigma_x} \right)^2 - \left(\frac{y - \mu_y}{\sigma_y} \right)^2 \right) \right]$$

$$x_j = x_{i+1}, \quad \sigma_{x_j} = \sigma_{x_i}$$

$$\Rightarrow \frac{g(x_j | x_i)}{g(x_i | x_j)} = \frac{\sigma_{x_j}}{\sigma_{x_i}} \exp \left[\frac{1}{2} \left(\left(\frac{x_j - \overset{x_i}{x}_{j+1}}{\sigma_{x_j}} \right)^2 - \left(\frac{x_i - \overset{x_j}{x}_{i+1}}{\sigma_{x_i}} \right)^2 \right) \right]$$
$$= 1 \cdot e^0 = 1$$

Code für blatt04

Bange, Burkowitz, Harnisch

27. November 2017

```
..../blatt04/Bange_Burkowitz_Harnisch/aufgl2.py

1 import h5py
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from pylab import rcParams
5 from uncertainties import ufloat
6
7 rcParams['figure.figsize'] = 10, 5.8
8 rcParams['legend.numpoints'] = 1
9
10
11 # _____ b) _____
12 def aufgl2b(plot=True):
13     p0 = np.random.multivariate_normal([0, 3], [[3.5**2, 0.9*3.5*2.5],
14                                         [0.9*3.5*2.5, 2.5**2]], size=10000).T
15
16     p1 = np.empty((2, 10000))
17     p1[0] = np.random.normal(6, 3.5, size=10000)
18     p1[1] = np.random.normal(-0.5 + 0.6*p1[0], 1)
19
20     if plot:
21         plt.scatter(p0[0], p0[1], s=1, alpha=0.25, label="$P_0$")
22         plt.scatter(p1[0], p1[1], s=1, alpha=0.25, label="$P_1$")
23         plt.legend()
24         plt.xlabel("$x$")
25         plt.ylabel("$y$")
26         # plt.show()
27         plt.savefig("A12b.pdf")
28
29     return (p0, p1)
30
31
32 def aufgl2c(p0, p1):
33     # Vereinigung der beiden Populationen bilden
34     # und alles in eine Liste packen
35     ps = [np.concatenate((p0[0], p1[0])),
36           np.concatenate((p0[1], p1[1]))],
37     p0, p1
38
39     for i, p in enumerate(ps):
40         # Kovarianz
41         cov = np.cov(p)
42
43         # Mittelwerte und Standardabweichungen
44         m = [ufloat(np.average(p[0]), np.sqrt(cov[0][0])),
45               ufloat(np.average(p[1]), np.sqrt(cov[1][1]))]
46
47         # Korrelationskoeffizient
48         r = cov[0][1]/(m[0].s*m[1].s)          np.correl(p)
49         if i == 0:
50             print("Gesamtpopulation:\n-----")
51         else:
52             print("Population %i\n-----" % (i - 1))
53         print("cov =", cov)
54         print("m =", m)
55         print("r =", r)
56         print("\n")
57
```

```

59 def aufg12d(p0, p1):
60     p0_1000 = np.random.multivariate_normal([0, 3], [[3.5**2, 0.9*3.5*2.5],
61                                             [0.9*3.5*2.5, 2.5**2]], size=1000).T
62
63     fh = h5py.File("pops.hdf5", 'w')
64     fh.create_dataset("p0", data=p0)
65     fh.create_dataset("p0_1000", data=p0_1000)
66     fh.create_dataset("p1", data=p1)
67     fh.close()
68
69
70 # ----- Main -----
71 if __name__ == '__main__':
72     np.random.seed(1234)
73     p0, p1 = aufg12b(True)
74     aufg12c(p0, p1)
75     aufg12d(p0, p1)

    ./blatt04/Bange_Burkowitz_Harnisch/aufg13.py

1 import numpy as np
2 import h5py
3 import matplotlib.pyplot as plt
4 from pylab import rcParams
5
6 rcParams['figure.figsize'] = 10, 5.8
7 rcParams['legend.numpoints'] = 1
8
9
10 # ----- a) -----
11 def aufg13ab():
12     fh = h5py.File("../A12/pops.hdf5", "r")
13     p0 = np.copy(fh["p0"])
14     p1 = np.copy(fh["p1"])
15     fh.close()
16
17     x = np.linspace(-10, 20, 1000)
18     plt.scatter(p0[0], p0[1], marker=".", s=2.5, alpha=0.5, label=r"$p_0$")
19     plt.scatter(p1[0], p1[1], marker=".", s=2.5, alpha=0.5, label=r"$p_1$")
20     plt.plot(x, 0*x, label=r"$g_1$")
21     plt.plot(x, -0.75*x, label=r"$g_2$")
22     plt.plot(x, -1.25*x, label=r"$g_3$")
23     plt.xlabel(r"$x$")
24     plt.ylabel(r"$y$")
25     plt.xlim(-10, 20)
26     plt.ylim(-7.5, 12.5)
27     plt.legend()
28     # plt.show()
29     plt.savefig("A13a_scatter.pdf")
30     plt.clf()
31
32     proj = np.zeros((3, 2, len(p0[0])))
33     g = [[-1, 0], [-0.8, 0.6], [-4/np.sqrt(41), 5/np.sqrt(41)]]
34     for i in range(3):
35         proj[i][0] = np.add(np.multiply(p0[0], g[i][0]),
36                             np.multiply(p0[1], g[i][1]))
37         proj[i][1] = np.add(np.multiply(p1[0], g[i][0]),
38                             np.multiply(p1[1], g[i][1]))
39     bins = np.linspace(-20, 10, 100)
40     plt.hist(proj[i][0], bins=bins, histtype="barstacked", label="SP_0$")
41     plt.hist(proj[i][1], bins=bins, histtype="barstacked", label="SP_1$",
42               alpha=0.6)
43     plt.xlabel(r"$x_{g_i}$" % (i + 1))
44     plt.ylabel("$n$")
45     plt.legend()
46     # plt.show()
47     plt.savefig("A13b_%i.pdf" % (i + 1))
48     plt.clf()
49 return proj
50

```

✓

```

51
52 # _____ a)
53 def aufg13c(proj, nl=10000):
54     for i, p in enumerate(proj):
55         p[0][::-1].sort()
56         p[1][::-1].sort()
57         ls = np.linspace(max(p[0][0], p[1][0]),
58                           min(p[0][-1], p[1][-1]), nl)
59         tp = np.zeros(nl)
60         fp = np.zeros(nl)
61         fn = np.zeros(nl)
62         tn = np.zeros(nl)
63         j = 0
64         k = 0
65         for n, l in enumerate(ls):
66             while (j < len(p[0])) and (p[0][j] >= l):
67                 j += 1
68             while k < len(p[1]) and p[1][k] >= l:
69                 k += 1
70             tp[n] = j
71             fn[n] = len(p[0]) - j
72             fp[n] = k
73             tn[n] = len(p[0]) - k
74             # plt.plot(ls, tp)
75             # plt.plot(ls, fp)
76             # plt.plot(ls, fn)
77             plt.plot(ls, tp/(tp + fp), label="Reinheit")
78             plt.plot(ls, tp/(tp + fn), label="Effizienz")
79             plt.plot(ls, (tp + tn)/(tp + tn + fp + fn), label="Genauigkeit")
80             plt.xlim(ls[-1], ls[0])
81             plt.legend()
82             plt.xlabel("$x_{\text{gli}}$ % (i + 1)")
83             # plt.show()
84             plt.savefig("A13c_gli.pdf" % i)
85             plt.clf()
86
87
88 # _____ Main
89 if __name__ == '__main__':
90     proj = aufg13ab()
91     aufg13c(proj)

```

```

.../blatt04/Bange_Burkowitz_Harnisch/aufg15.py

1 from mcmc import MCMC
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from pylab import rcParams
5
6 rcParams['figure.figsize'] = 10, 5.8
7 rcParams['legend.numpoints'] = 1
8
9
10 # _____ c)
11 def aufg15c():
12     normal = MCMC(2, -3, 2)
13     sample = normal.sample(15, 10000)
14     bins = plt.hist(sample, bins=50,
15                      density=True, label="Gezogen")[1]
16     x = np.linspace(bins[0], bins[-1], 1000)
17     plt.plot(x, normal.pdf(x), label="PDF")
18     plt.xlabel("$x$")
19     plt.ylabel("Wahrscheinlichkeitsdichte")
20     plt.legend()
21     # plt.show()
22     plt.savefig("A15c.pdf")
23     plt.clf()
24     return sample
25
26
27 # _____ d)

```

```

28 def aufg15d(sample):
29     plt.plot(np.arange(len(sample)), sample, ".", ms=3)
30     plt.xlim(-100, len(sample))
31     plt.xlabel("$t\backslash, \backslash, \$a.u.$")
32     plt.ylabel("$x\backslash, \backslash, \$a.u.$")
33     plt.savefig("A15d.pdf")
34     plt.xlim(-5, 200)
35     plt.savefig("A15d_zoom.pdf")
36
37 # ----- Main -----
38 if __name__ == '__main__':
39     np.random.seed(1234)
40     sample = aufg15c()
41     aufg15d(sample)

.. ./blatt04/Bangs_Burkowitz_Harnisch/mcmc.py

1 import numpy as np
2 import scipy.stats as scs
3
4
5 class MCMC(object):
6     def __init__(self, step_size=1., loc=0., scale=1.):
7         """
8             Sample from a 1D gaussian PDF with uniform step proposal.
9
10            Parameters
11
12            loc, scale : float
13                Mean and standard deviation for the gaus to sample from.
14            step_size = float
15                Step sized used symmetrically around the current step to propose
16                the next one from a uniform PDF in ``[-step_size, step_size]``.
17        """
18        self.step_size = step_size
19        self.pdf = scs.norm(loc, scale).pdf
20
21    def _propose_step(self, xi):
22        """
23            Calculate the next proposed step from the current one from the
24            step proposal PDF (here: uniform).
25
26            Parameters
27
28            xi : float
29                Current step from which the next positions is calculated.
30
31            Returns
32
33            xj : float
34                Next proposed step.
35        """
36        return xi + np.random.uniform(-1, 1)*self.step_size
37
38    def _accept_step(self, xi, xj):
39        """
40            Decide wether to accept the next step or not using the
41            Metropolis-Hastings detailed balance condition.
42
43            Parameters
44
45            xi : float
46                Current step from which the next positions is calculated.
47            xj : float
48                Next proposed step.
49
50            Returns
51
52            acc : bool
53                ``True`` if the next step is accepted, ``False`` if not.

```

```

54     """
55     return np.random.uniform() <= self.pdf(xj)/self.pdf(xi)
56
57     def sample(self, x0, n=1):
58         """
59         Sample ``n`` points from the gaussian PDF using the MCMC algorithm.
60
61         Parameters
62
63         x0 : float
64             Start value where the Markov chain is started.
65         n : int
66             How many samples to create.
67
68         Returns
69
70         x : array-like
71             Created sample points. Has length ``n``.
72         """
73         x = np.empty(n, dtype=float)
74         x[0] = x0
75         for i in range(1, n):
76             proposed = self._propose_step(x[i-1])
77             if self._accept_step(x[i-1], proposed):
78                 x[i] = proposed
79             else:
80                 x[i] = x[i-1]
81
82         return x

```

Zeit	Raum	Abgabe im Moodle; Mails mit Betreff: [SMD1718]
Di. 10-12	CP-03-150	philipp2.hoffmann@udo.edu und jan.soedingrekso@udo.edu
Di. 16-18	P1-02-110	felix.neubuerger@udo.edu und tobias.hoinka@udo.edu
Di. 16-18	CP-03-150	simone.mender@udo.edu und maximilian.meier@udo.edu

Aufgabe 16: *Fisher-Diskriminante: Implementierung*

10 P.

Gegeben seien die Populationen P_0 und P_1 aus der Aufgabe „Zwei Populationen“. Nutzen Sie das dort erstellt HDF5-File für diese Aufgabe. (Sie finden die Datei ebenfalls im Moodle.)

Hinweis: Es sei Ihnen erlaubt Pakete z.B. für lineare Algebra zu benutzen, jedoch nicht Pakete, die die Diskriminanzanalyse durchführen.

- a) Berechnen Sie die Mittelwerte μ_{P_0} und μ_{P_1} der beiden Populationen.
- b) Berechnen Sie die Kovarianzmatrizen V_{P_0} und V_{P_1} der beiden Populationen, sowie die kombinierte Kovarianzmatrix V_{P_0, P_1} .
- c) Konstruieren Sie eine lineare Fisher-Diskriminante $\vec{\lambda} = \lambda \cdot \vec{e}_{\vec{\lambda}}$. Geben Sie diese Geradengleichung an.
- d) Stellen Sie die Populationen als Projektion auf die Gerade aus c) in einem eindimensionalen Histogramm dar.
- e) Betrachten Sie P_0 als Signal und P_1 als Untergrund. Berechnen Sie die Effizienz und die Reinheit des Signals als Funktion eines Schnittes λ_{cut} in λ und stellen Sie die Ergebnisse in einem Plot dar.
- f) Bei welchem Wert von λ_{cut} wird nach der Trennung das Signal-zu-Untergrundverhältnis S/B maximal? Erstellen Sie auch hierzu einen Plot.
- g) Bei welchem Wert von λ_{cut} wird nach der Trennung die Signifikanz $S/\sqrt{S+B}$ maximal? Erstellen Sie auch hierzu einen Plot.
- h) Wiederholen Sie die Schritte a) bis g) für den Fall, dass P_0 nun die Population P_0 bezeichnet.

Aufgabe 17: *kMeans per Hand*

5 P.

Population: (1;4) (1;5) (1;6) (3;3) (3;2) (4;1) (5;1) (6;2) (6;3) (8;4) (8;5) (8;6)

- a) Führen Sie den kMeans-Algorithmus (euklidisches Abstandsmaß) per Hand durch, um die Punkte der Population in Cluster zu gruppieren. Verwenden Sie als Startwerte die zufällig gewählten Clusterzentren (3;4), (7;4) und (3;7). Berechnen Sie die Abstände nur, wenn die Zugehörigkeit zum Clusterzentrum nicht offensichtlich ist. Skizzieren Sie die neuen Clusterzentren sowie die Grenzen zwischen den Clustern in der vorgefertigten Abbildung 1.
- b) Führen Sie 4 weitere Iterationen von kMeans durch. Fertigen Sie bei jeder Iteration wieder eine Skizze an.
- c) Nach wie vielen Iterationen konvergiert der Algorithmus? Entspricht das Ergebnis Ihren Erwartungen?

Aufgabe 18: *Hauptkomponentenanalyse (PCA)*

5 P.

- a) Erzeugen Sie mit der Funktion `sklearn.datasets.make_blobs` einen Datensatz. Nutzen Sie dabei folgende Einstellungen: `n_samples=1000, centers=2, n_features=4, random_state=0`. Stellen Sie nun zwei beliebige Dimensionen des Datensatzes in einem Scatterplot dar.
- b) Beschreiben Sie kurz die Funktionsweise der Hauptkomponentenanalyse. Geben Sie in Worten und in der richtigen Reihenfolge die notwendigen Berechnungen zur Durchführung der Hauptkomponentenanalyse an.
- c) Wenden Sie nun die Hauptkomponentenanalyse auf den in a) erzeugten Datensatz an. Nutzen Sie dazu das Paket `sklearn.decomposition.PCA`. Wie lauten die Eigenwerte der Kovarianzmatrix? Wie interpretieren Sie die Eigenwerte?
- d) Histogrammieren Sie nun x' in jeder Dimension und stellen sie x'_1 und x'_2 in einem Scatterplot dar.

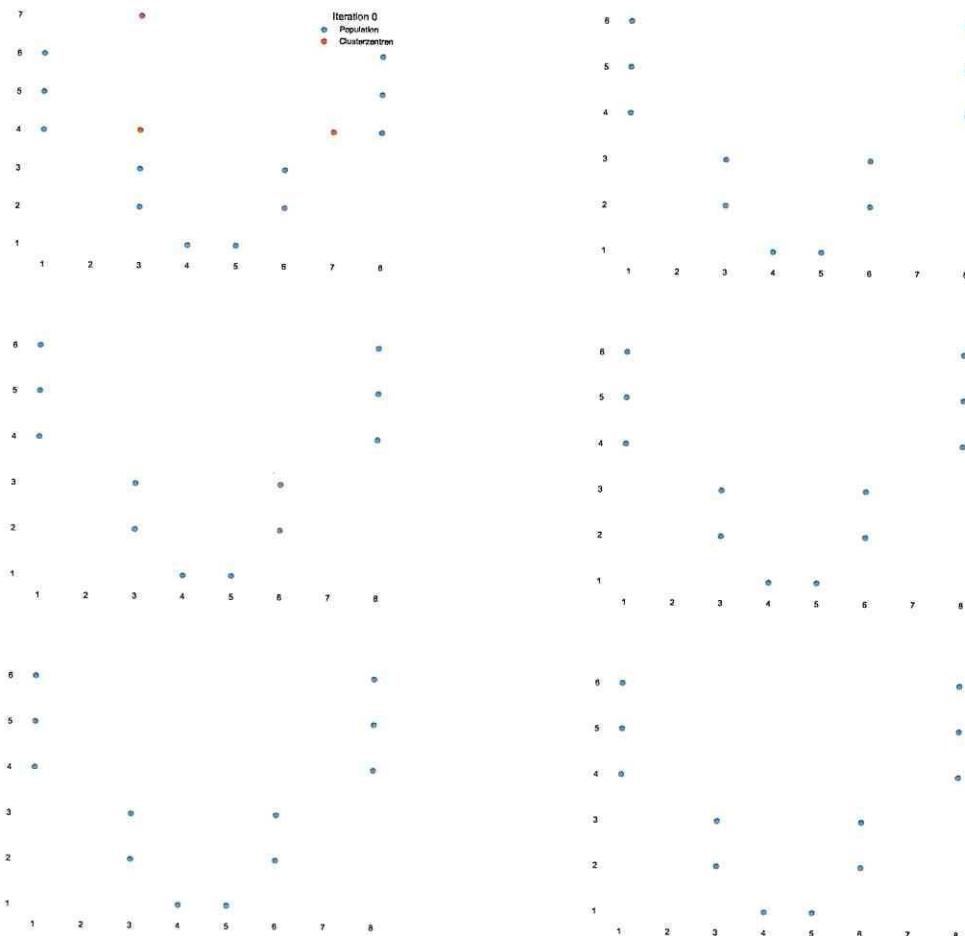


Abbildung 1: Population zum Einzeichnen der Clusterzentren und Clustergrenzen.
Zu Aufgabe 17

Aufgabe 16

a) b)

Siehe Aufgabe 12 c). Wird außerdem nochmal vom Programm ausgegeben.

c)

Mit $P_{0,10000}$ ergibt sich die Geradengleichung

$$g(x) = -1,2743x . \quad (1)$$

Mit $P_{0,1000}$ ergibt sich die Geradengleichung

$$g(x) = -1,2683x . \quad (2)$$

d)

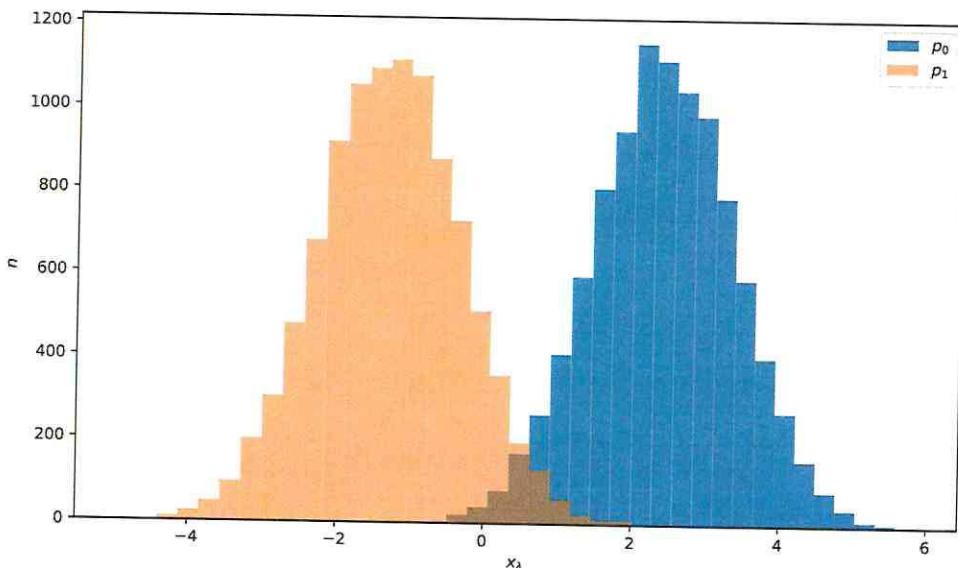


Abbildung 1: Histogramm der Projektion für $P_{0,10000}$.

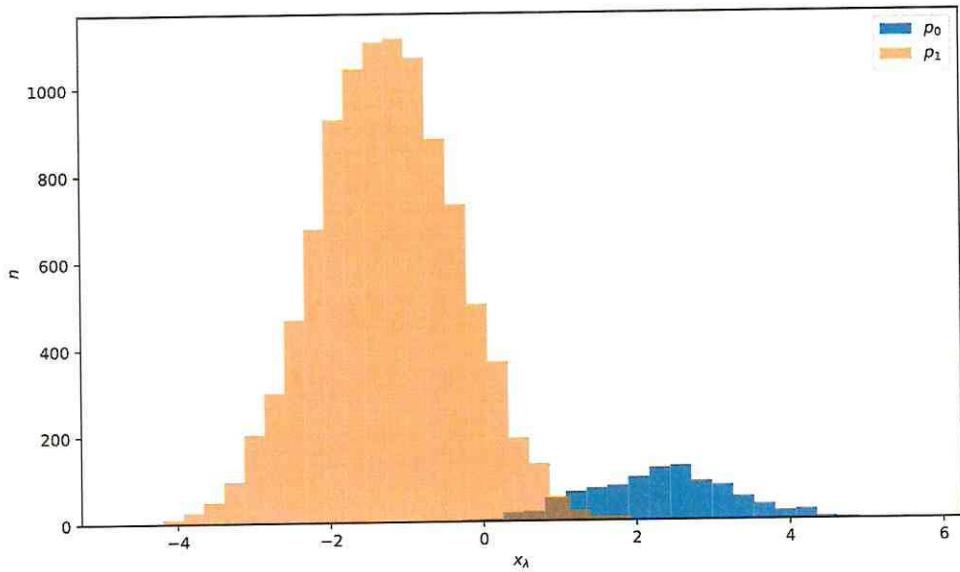


Abbildung 2: Histogramm der Projektion für $P_{0,1000}$.

e)

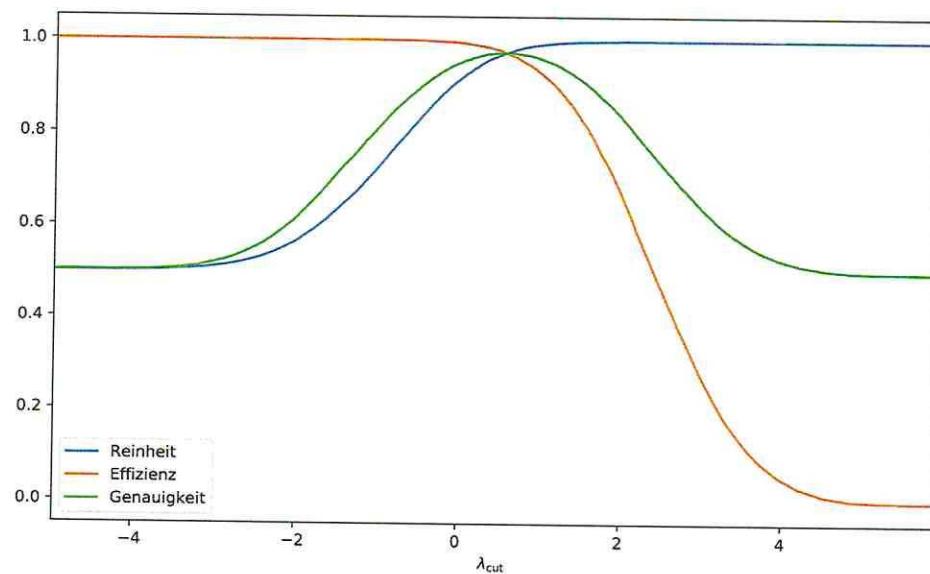


Abbildung 3: Effizienz, Reinheit und Genauigkeit für $P_{0,10000}$.

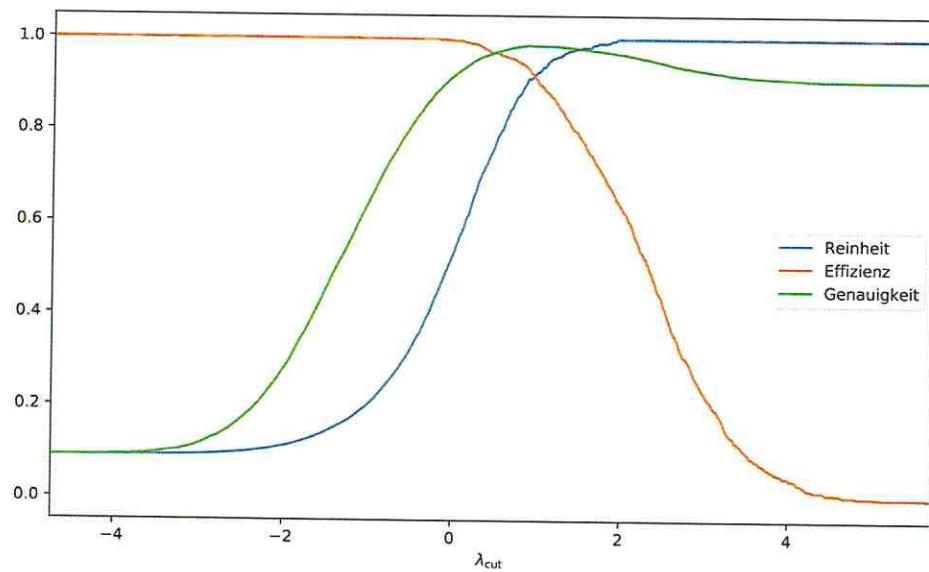


Abbildung 4: Effizienz, Reinheit und Genauigkeit für $P_{0,1000}$.

f)

Das Signal-zu-Untergrundverhältnis $\frac{S}{B}$ wird maximal für $B \rightarrow 0$ also für $\lambda_{\text{cut}} \rightarrow -\infty$.

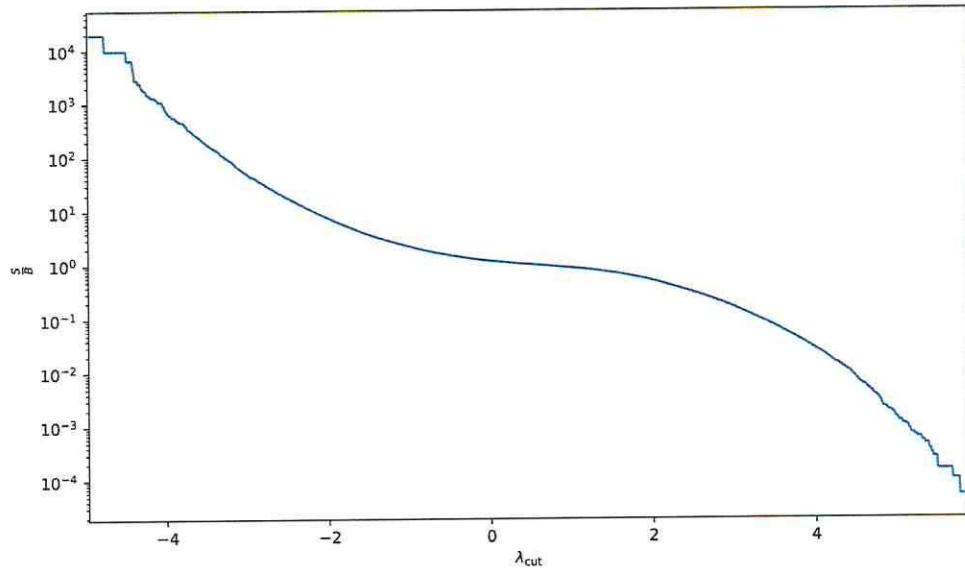


Abbildung 5: Signal-zu-Untergrundverhältnis für $P_{0,10000}$.

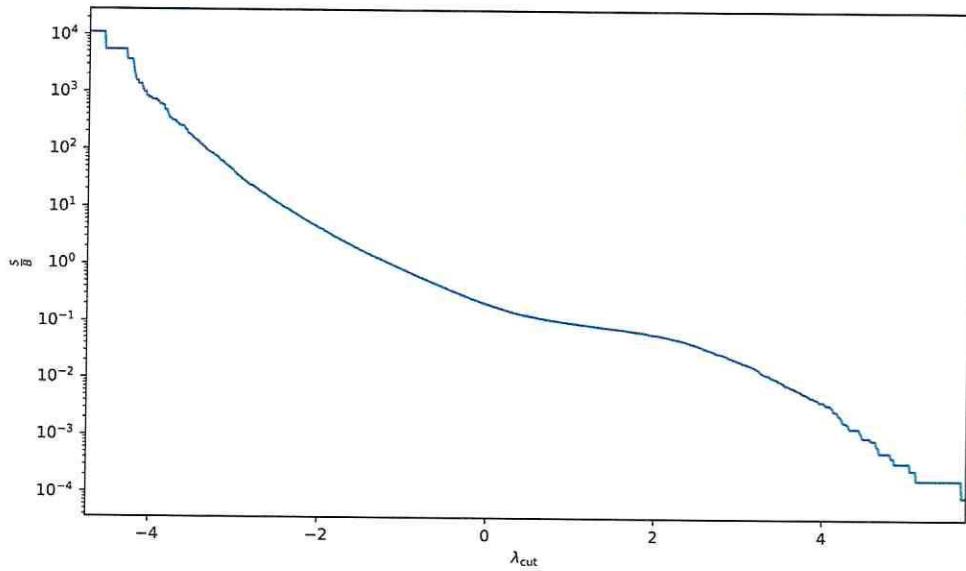


Abbildung 6: Signal-zu-Untergrundverhältnis für $P_{0,1000}$.

g)

Die Signifikanz $\frac{S}{\sqrt{S+B}}$ wird maximal für $S + B \rightarrow 0$ also für $\lambda_{\text{cut}} \rightarrow -\infty$.

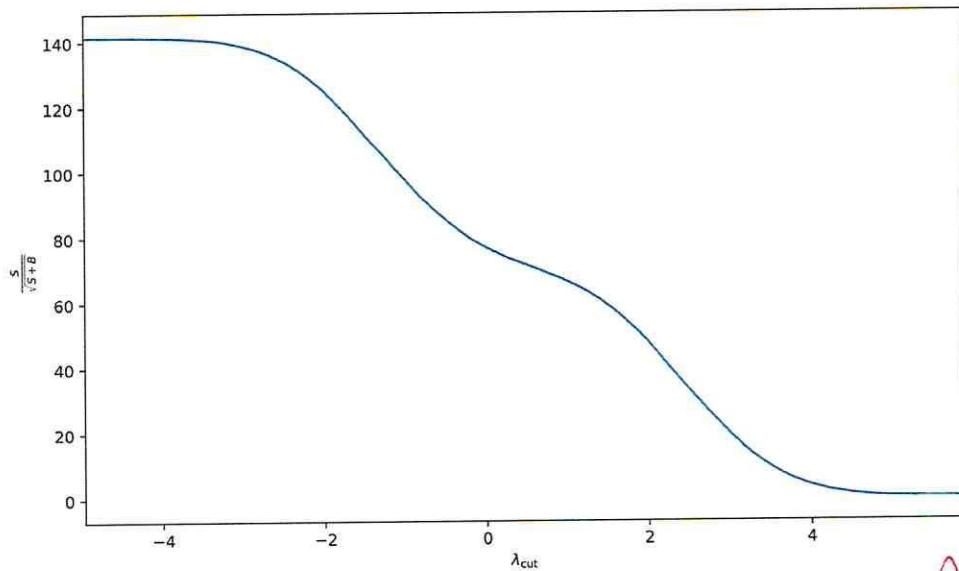


Abbildung 7: Signifikanz für $P_{0,10000}$.

Diese Plots sehen nicht richtig aus, noch cool.

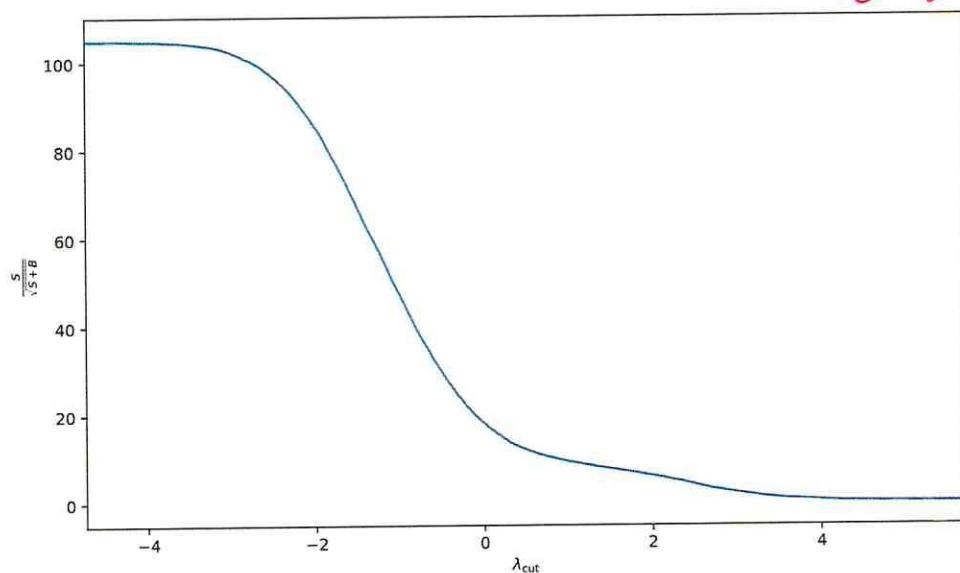


Abbildung 8: Signifikanz für $P_{0,1000}$.

7/10

Aufgabe 17: k-Means per Hand

Population: $\vec{x}_1 = \begin{pmatrix} 1 \\ 4 \end{pmatrix}$, $\vec{x}_2 = \begin{pmatrix} 1 \\ 5 \end{pmatrix}$, $\vec{x}_3 = \begin{pmatrix} 1 \\ 6 \end{pmatrix}$, $\vec{x}_4 = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$, $\vec{x}_5 = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$
 $\vec{x}_6 = \begin{pmatrix} 4 \\ 1 \end{pmatrix}$, $\vec{x}_7 = \begin{pmatrix} 5 \\ 1 \end{pmatrix}$, $\vec{x}_8 = \begin{pmatrix} 6 \\ 2 \end{pmatrix}$, $\vec{x}_9 = \begin{pmatrix} 6 \\ 3 \end{pmatrix}$, $\vec{x}_{10} = \begin{pmatrix} 8 \\ 4 \end{pmatrix}$
 $\vec{x}_{11} = \begin{pmatrix} 5 \\ 5 \end{pmatrix}$, $\vec{x}_{12} = \begin{pmatrix} 8 \\ 6 \end{pmatrix}$

a) Start-Clusterzentren: $S_1^{(0)} = \begin{pmatrix} 3 \\ 7 \end{pmatrix}$, $S_2^{(0)} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$, $S_3^{(0)} = \begin{pmatrix} 7 \\ 4 \end{pmatrix}$

Punkte zuordnen:

Cluster 1: $\{\vec{x}_3\}$

Cluster 2: $\{\vec{x}_1, \vec{x}_2, \vec{x}_4, \vec{x}_5, \vec{x}_6\}$

Cluster 3: $\{\vec{x}_7, \vec{x}_8, \vec{x}_9, \vec{x}_{10}, \vec{x}_{11}, \vec{x}_{12}\}$

Alle eindeutig zuordnbar bis auf \vec{x}_7 :

$$\begin{aligned} \|\vec{x}_7 - S_2^{(0)}\|^2 &= \| \begin{pmatrix} 5 \\ 1 \end{pmatrix} - \begin{pmatrix} 3 \\ 4 \end{pmatrix} \|^2 = 2^2 + (-3)^2 = 13 \\ \|\vec{x}_7 - S_3^{(0)}\|^2 &= \| \begin{pmatrix} 5 \\ 1 \end{pmatrix} - \begin{pmatrix} 7 \\ 4 \end{pmatrix} \|^2 = (-2)^2 + 3^2 = 13 \end{aligned} \quad \left. \begin{array}{l} \text{identisch} \rightarrow \text{scheide einen} \\ \text{Cluster aus} \Rightarrow \text{Cluster 3} \end{array} \right.$$

1. Iteration:

Neue Clusterzentren berechnen

$$S_1^{(1)} = \begin{pmatrix} 1 \\ 6 \end{pmatrix}$$

$$S_2^{(1)} = \frac{1}{5} \left[\begin{pmatrix} 1 \\ 4 \end{pmatrix} + \begin{pmatrix} 1 \\ 5 \end{pmatrix} + \begin{pmatrix} 3 \\ 3 \end{pmatrix} + \begin{pmatrix} 3 \\ 2 \end{pmatrix} + \begin{pmatrix} 4 \\ 1 \end{pmatrix} \right] = \frac{1}{5} \begin{pmatrix} 12 \\ 15 \end{pmatrix} = \begin{pmatrix} 2,4 \\ 3 \end{pmatrix}$$

$$S_3^{(1)} = \frac{1}{6} \left[\begin{pmatrix} 5 \\ 1 \end{pmatrix} + \begin{pmatrix} 6 \\ 2 \end{pmatrix} + \begin{pmatrix} 6 \\ 3 \end{pmatrix} + \begin{pmatrix} 8 \\ 4 \end{pmatrix} + \begin{pmatrix} 8 \\ 5 \end{pmatrix} + \begin{pmatrix} 8 \\ 6 \end{pmatrix} \right] = \frac{1}{6} \begin{pmatrix} 41 \\ 21 \end{pmatrix} \approx \begin{pmatrix} 6,833 \\ 3,5 \end{pmatrix}$$

Punkte neu zuordnen:

Cluster 1: $\{\vec{x}_2, \vec{x}_3\}$

Cluster 2: $\{\vec{x}_1, \vec{x}_4, \vec{x}_5, \vec{x}_6\}$

Cluster 3: $\{\vec{x}_7, \vec{x}_8, \vec{x}_9, \vec{x}_{10}, \vec{x}_{11}, \vec{x}_{12}\}$

\vec{x}_7 nicht offensichtlich zuordnbar:

$$\begin{aligned} \|\vec{x}_7 - S_2^{(1)}\|^2 &= \| \begin{pmatrix} 5 \\ 1 \end{pmatrix} - \begin{pmatrix} 2,4 \\ 3 \end{pmatrix} \|^2 = 2,6^2 + (-2)^2 = 10,76 \\ \|\vec{x}_7 - S_3^{(1)}\|^2 &\approx \| \begin{pmatrix} 5 \\ 1 \end{pmatrix} - \begin{pmatrix} 6,833 \\ 3,5 \end{pmatrix} \|^2 \approx 3,361^2 + 6,25 = 9,611 \end{aligned} \quad \left. \begin{array}{l} \vec{x}_7 \text{ gehört zu} \\ \text{Cluster 3} \end{array} \right.$$

b) 2. Iteration:

$$S_1^{(2)} = \frac{1}{2} \left[\begin{pmatrix} 1 \\ 5 \end{pmatrix} + \begin{pmatrix} 1 \\ 6 \end{pmatrix} \right] = \begin{pmatrix} 1 \\ 5,5 \end{pmatrix}$$

$$S_2^{(2)} = \frac{1}{4} \left[\begin{pmatrix} 1 \\ 4 \end{pmatrix} + \begin{pmatrix} 3 \\ 3 \end{pmatrix} + \begin{pmatrix} 3 \\ 2 \end{pmatrix} + \begin{pmatrix} 4 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} 2,75 \\ 2,5 \end{pmatrix}$$

$$S_3^{(2)} = \cancel{S_3^{(1)}} \quad S_3^{(1)} \approx \begin{pmatrix} 6,833 \\ 3,5 \end{pmatrix}$$

\vec{x}_7 nicht eindeutig zuordnbar:

$$\begin{aligned} \|\vec{x}_7 - S_2^{(2)}\|^2 &= \| \begin{pmatrix} 5 \\ 1 \end{pmatrix} - \begin{pmatrix} 2,75 \\ 2,5 \end{pmatrix} \|^2 = 2,25^2 + (-1,5)^2 = 7,3125 \\ \|\vec{x}_7 - S_3^{(2)}\|^2 &= 9,611 \quad (\text{s.o.b.}) \end{aligned} \quad \left. \begin{array}{l} \vec{x}_7 \text{ gehört zu} \\ \text{Cluster 2} \end{array} \right.$$

- Cluster 1: $\{\vec{x}_1, \vec{x}_2, \vec{x}_3\}$
 Cluster 2: $\{\vec{x}_4, \vec{x}_5, \vec{x}_6, \vec{x}_7\}$
 Cluster 3: $\{\vec{x}_8, \vec{x}_9, \vec{x}_{10}, \vec{x}_{11}, \vec{x}_{12}\}$

3. Iteration:

Neue Clusterzentren berechnen

$$S_1^{(3)} = \frac{1}{3} \left[\begin{pmatrix} 1 \\ 6 \end{pmatrix} + \begin{pmatrix} 1 \\ 5 \end{pmatrix} + \begin{pmatrix} 1 \\ 4 \end{pmatrix} \right] = \begin{pmatrix} 1 \\ 5 \end{pmatrix}$$

$$S_2^{(3)} = \frac{1}{4} \left[\begin{pmatrix} 2 \\ 3 \end{pmatrix} + \begin{pmatrix} 3 \\ 2 \end{pmatrix} + \begin{pmatrix} 4 \\ 1 \end{pmatrix} + \begin{pmatrix} 5 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} 3,75 \\ 1,75 \end{pmatrix}$$

$$S_3^{(3)} = \frac{1}{5} \left[\begin{pmatrix} 6 \\ 2 \end{pmatrix} + \begin{pmatrix} 5 \\ 3 \end{pmatrix} + \begin{pmatrix} 8 \\ 4 \end{pmatrix} + \begin{pmatrix} 8 \\ 5 \end{pmatrix} + \begin{pmatrix} 8 \\ 6 \end{pmatrix} \right] = \begin{pmatrix} 7,2 \\ 4 \end{pmatrix}$$

\vec{x}_8 nicht eindeutig zuordnbar

$$\begin{aligned} \|\vec{x}_8 - S_2^{(3)}\|^2 &= \left\| \begin{pmatrix} 6 \\ 2 \end{pmatrix} - \begin{pmatrix} 3,75 \\ 1,75 \end{pmatrix} \right\|^2 = 2,25 + (-0,25)^2 = 2,5 \\ \|\vec{x}_8 - S_3^{(3)}\|^2 &= \left\| \begin{pmatrix} 6 \\ 2 \end{pmatrix} - \begin{pmatrix} 7,2 \\ 4 \end{pmatrix} \right\|^2 = (-1,2)^2 + (-2)^2 = 5,44 \end{aligned} \quad \left. \begin{array}{l} \vec{x}_8 \text{ gehört zu} \\ \text{Cluster 2} \end{array} \right\}$$

- Cluster 1: $\{\vec{x}_1, \vec{x}_2, \vec{x}_3\}$
 Cluster 2: $\{\vec{x}_4, \vec{x}_5, \vec{x}_6, \vec{x}_7, \vec{x}_8\}$
 Cluster 3: $\{\vec{x}_9, \vec{x}_{10}, \vec{x}_{11}, \vec{x}_{12}\}$

4. Iteration:

Neue Clusterzentren berechnen

$$S_1^{(4)} = S_1^{(3)} = \begin{pmatrix} 1 \\ 5 \end{pmatrix}$$

$$S_2^{(4)} = \frac{1}{5} \left[\begin{pmatrix} 3 \\ 3 \end{pmatrix} + \begin{pmatrix} 3 \\ 2 \end{pmatrix} + \begin{pmatrix} 4 \\ 1 \end{pmatrix} + \begin{pmatrix} 5 \\ 1 \end{pmatrix} + \begin{pmatrix} 6 \\ 2 \end{pmatrix} \right] = \begin{pmatrix} 4,2 \\ 1,8 \end{pmatrix}$$

$$S_3^{(4)} = \frac{1}{4} \left[\begin{pmatrix} 6 \\ 2 \end{pmatrix} + \begin{pmatrix} 8 \\ 4 \end{pmatrix} + \begin{pmatrix} 8 \\ 5 \end{pmatrix} + \begin{pmatrix} 8 \\ 6 \end{pmatrix} \right] = \begin{pmatrix} 7,5 \\ 4,5 \end{pmatrix}$$

\vec{x}_9 nicht eindeutig zuordnbar:

$$\begin{aligned} \|\vec{x}_9 - S_2^{(4)}\|^2 &= \left\| \begin{pmatrix} 6 \\ 2 \end{pmatrix} - \begin{pmatrix} 4,2 \\ 1,8 \end{pmatrix} \right\|^2 = 1,8^2 + 1,2^2 = 4,68 \\ \|\vec{x}_9 - S_3^{(4)}\|^2 &= \left\| \begin{pmatrix} 6 \\ 2 \end{pmatrix} - \begin{pmatrix} 7,5 \\ 4,5 \end{pmatrix} \right\|^2 = (-1,5)^2 + (-1,5)^2 = 4,5 \end{aligned} \quad \left. \begin{array}{l} \vec{x}_9 \text{ gehört zu} \\ \text{Cluster 3} \end{array} \right\}$$

- Cluster 1: $\{\vec{x}_1, \vec{x}_2, \vec{x}_3\}$
 Cluster 2: $\{\vec{x}_4, \vec{x}_5, \vec{x}_6, \vec{x}_7, \vec{x}_8\}$
 Cluster 3: $\{\vec{x}_9, \vec{x}_{10}, \vec{x}_{11}, \vec{x}_{12}\}$

5. Iteration

→ keine Änderung mehr $S_1^{(5)} = S_1^{(4)}, S_2^{(5)} = S_2^{(4)}, S_3^{(5)} = S_3^{(4)}$

c) Der Algorithmus konvergiert in diesem Fall nach der 4. Iteration.
 Das Ergebnis entspricht nicht unseren Erwartungen. Unter der Annahme,
 dass es 3 Cluster gibt, hätten wir folgendes erwartet:

- Cluster 1: $\{\vec{x}_1, \vec{x}_2, \vec{x}_3\}$, Cluster 2: $\{\vec{x}_4, \vec{x}_5, \vec{x}_6, \vec{x}_7, \vec{x}_8, \vec{x}_9\}$, Cluster 3: $\{\vec{x}_{10}, \vec{x}_{11}, \vec{x}_{12}\}$

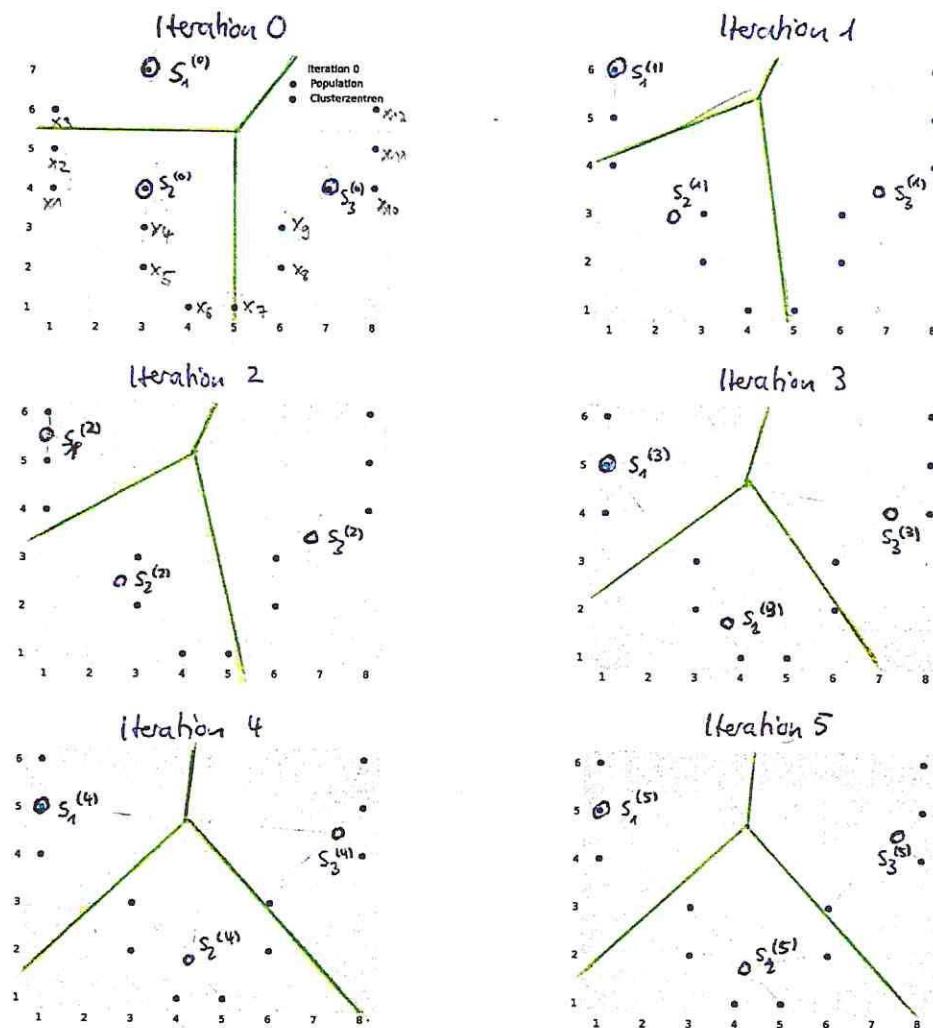


Abbildung 1: Population zum Einzeichnen der Clusterzentren und Clustergrenzen.
 Zu Aufgabe 17

Aufgabe 18

a)

Die Blobs wurden in der Ebene der ersten beiden Dimensionen geplottet.

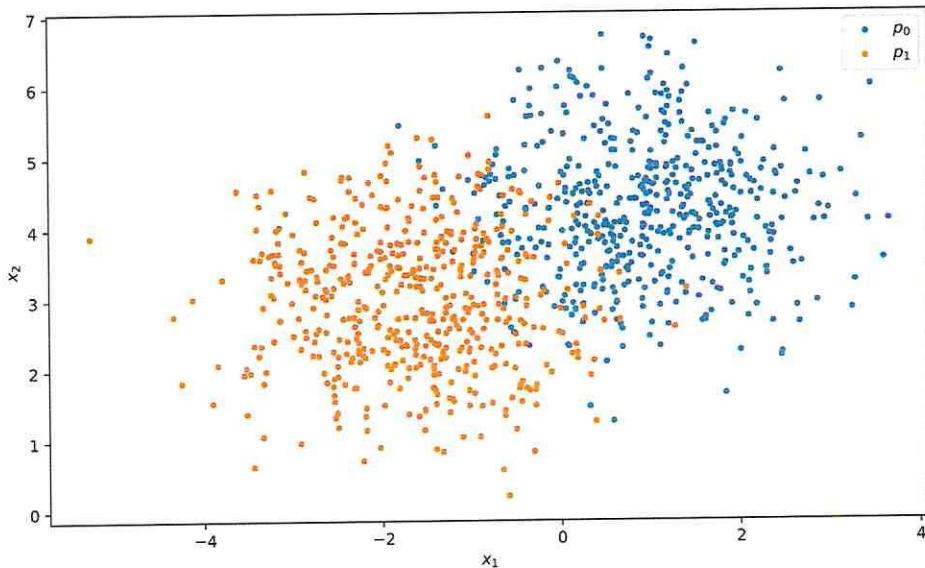


Abbildung 9: Scatterplot in der Ebene der ersten beiden Dimensionen; untransformiert.

b)

In der Hauptkomponentenanalyse sollen die Daten so transformiert werden, dass die Varianz entlang der Achsen extremal wird. Das heißt, die Daten werden verschoben und vor allem gedreht, um am Ende einige Achsen zu haben, bei der die Daten möglichst weit auseinander liegen. Sie haben also eine hohe Varianz. Dadurch wird die Varianz entlang der anderen Achsen zwangsweise kleiner, die Populationen liegen also praktisch übereinander. Diese Daten sind von keiner weiteren Bedeutung.

Der Ablauf:

1. Zentrierung der Daten: Der Mittelwert aller Daten wird von den Daten abgezogen, damit sie danach vernünftig gedreht werden können.
2. Berechnung der Kovarianzmatrix: Die Kovarianzmatrix wird benötigt, um eine Aussage über die Varianz treffen zu können.
3. Berechnung der Eigenwerte und -vektoren: Mit den Eigenwerten kann die Kovarianzmatrix diagonalisiert werden.
4. Die k größten Eigenwerte und zugehörige Eigenvektoren werden ausgewählt: Da die Kovarianzmatrix mit den Eigenwerten diagonalisiert werden kann, kann mit den Eigenvektoren, die zu den größten Eigenwerten gehören, eine Transformationsmatrix aufgestellt werden, die die Varianz maximiert.

5. Bildung einer $d \times k$ Matrix aus den k Eigenvektoren: Dies ist die eben angesprochene Transformationsmatrix.
6. Transformation der Daten mit der Transformationsmatrix: Auf diese Weise wird die Varianz der transformierten Daten x' extremalisiert.

c)

Die Eigenwerte lauten:

$$\begin{aligned}\lambda_1 &\approx 17,5 \\ \lambda_2 &\approx 0,999 \\ \lambda_3 &\approx 0,987 \\ \lambda_4 &\approx 0,898\end{aligned}$$

Daran wird direkt erkennlich, dass die Daten entlang der ersten Achse sehr gut getrennt werden und entlang der anderen Achsen sehr schlecht.

d)

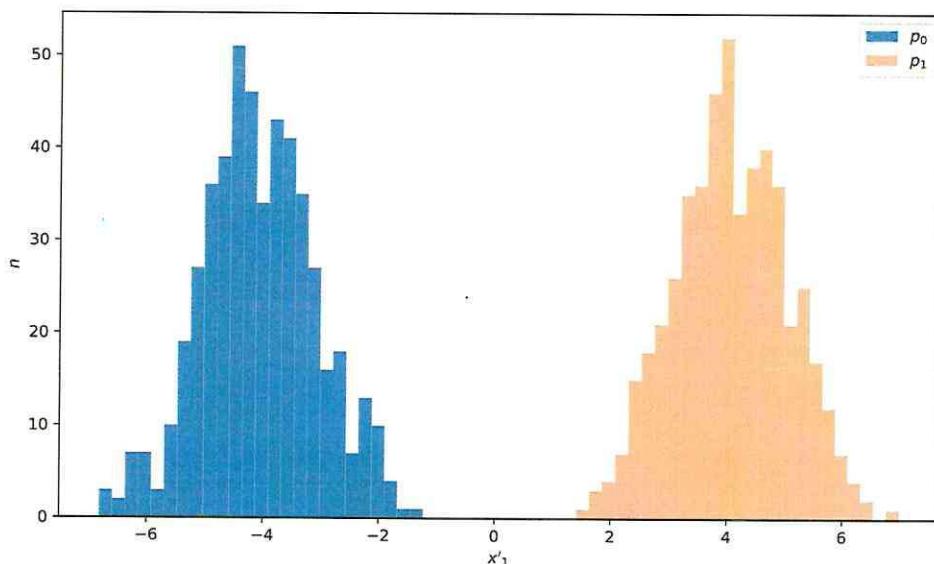


Abbildung 10: Histogramm für die Projektion auf die erste Achse.

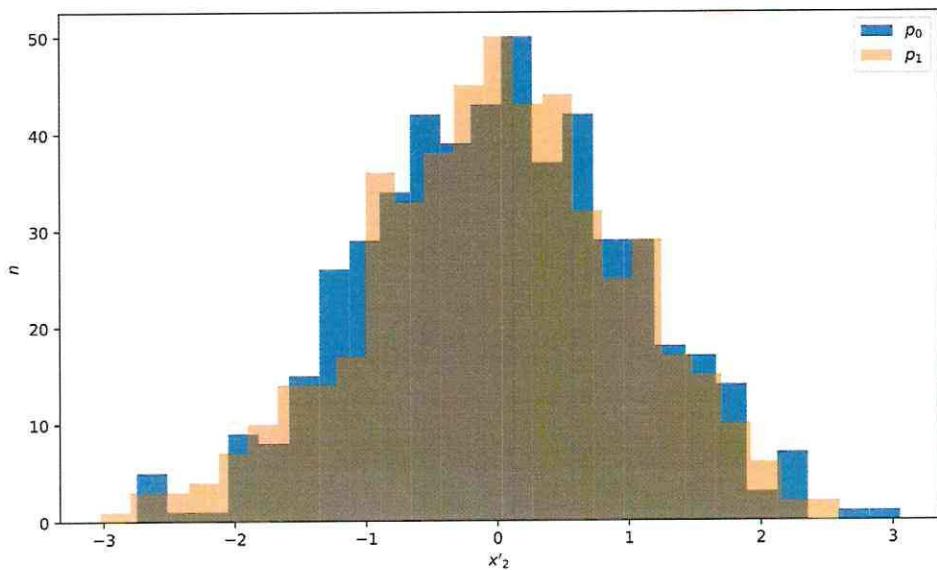


Abbildung 11: Histogramm für die Projektion auf die zweiten Achse.

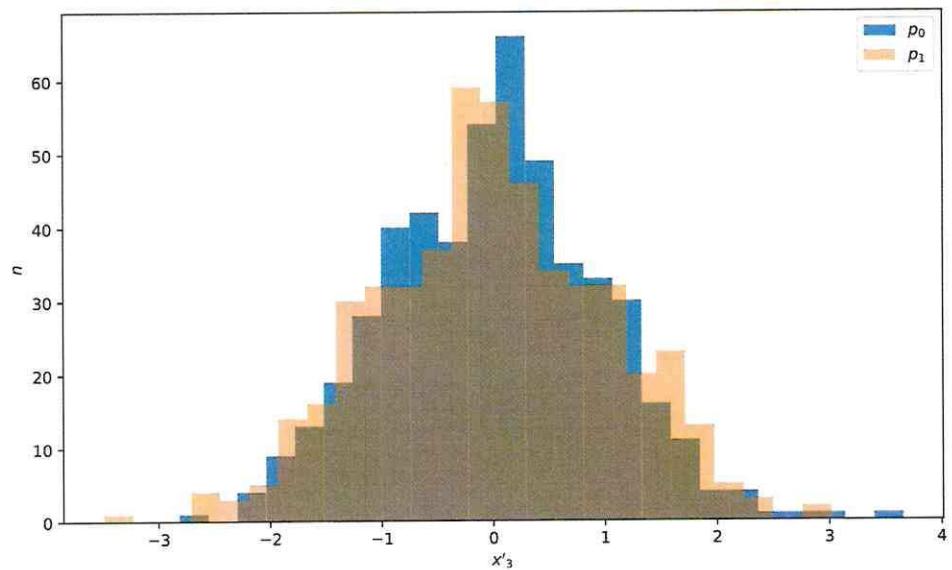


Abbildung 12: Histogramm für die Projektion auf die dritten Achse.

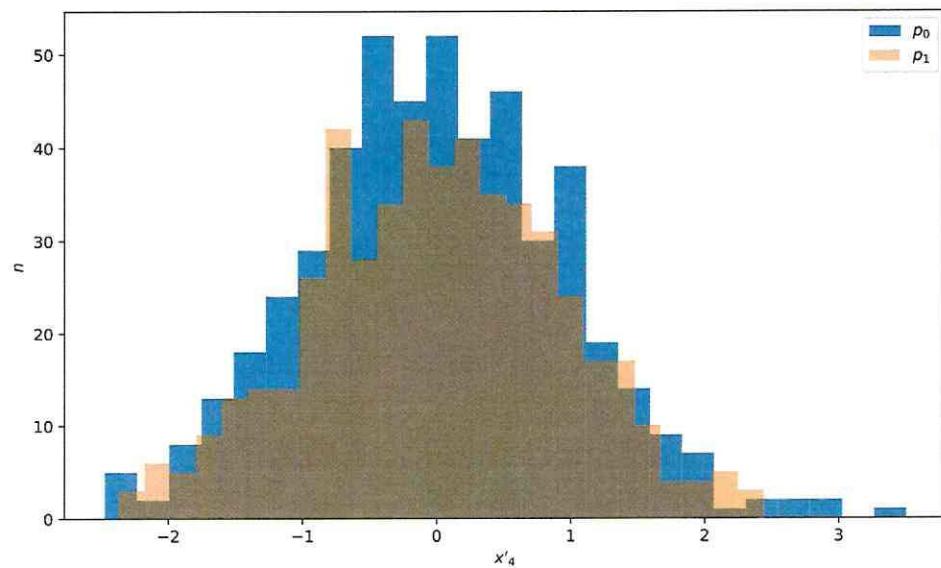


Abbildung 13: Histogramm für die Projektion auf die vierten Achse.

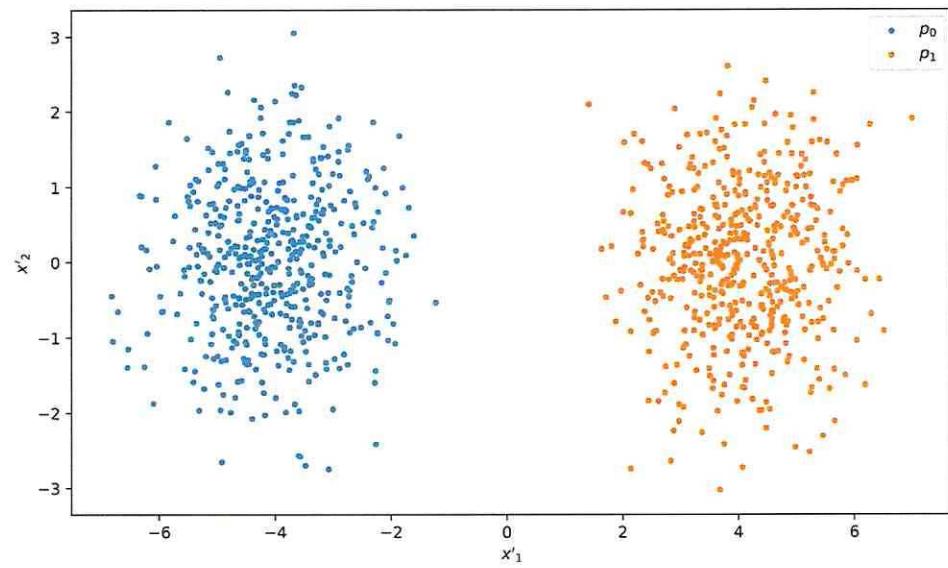


Abbildung 14: Scatterplot in der Ebene der ersten beiden transformierten Dimensionen.

Anhand der Histogramme ist deutlich ersichtlich, dass die Daten lediglich in der ersten transformierten Dimension gut trennbar sind. Die Erwartung, die aus den Eigenwerten folgte, wurde damit bestätigt. Dies ist auch an dem Scatterplot 14 ersichtlich, da die Daten zwar sehr deutlich nach links und rechts, also entlang der x'_1 Dimension, trennbar sind, allerdings nicht nach oben und unten, also entlang der x'_2 Dimension, trennbar sind.

✓

5/5

Code für blatt05

Bange, Burkowitz, Harnisch

4. Dezember 2017

```
..../blatt05/Bange_Burkowitz_Harnisch/aufg16.py
1 from fisher import Fisher
2 import numpy as np
3 import h5py
4 from pylab import rcParams
5
6 rcParams['figure.figsize'] = 10, 5.8
7 rcParams['legend.numpoints'] = 1
8
9
10 # -----
11 if __name__ == '__main__':
12     # Daten laden (Aufgabe 12)
13     fh = h5py.File("pops.hdf5", "r")
14     p0 = np.copy(fh["p0"])
15     p0_1000 = np.copy(fh["p_0_1000"])
16     p1 = np.copy(fh["p1"])
17     fh.close()
18
19     # Instanz unserer Fisher Klasse erstellen
20     # Führt Berechnungen für a) bis c) durch
21     fisher = Fisher(p0, p1)
22     # fisher.show()
23     print("Mittelwerte für p_0_10000:", fisher.m)
24     print("kombinierte Kovarianzmatrix für p_0_10000:", fisher.sw)
25
26     # c)
27     print("Geradengleichung der Projektion für P_0_10000: g(x) = %1.4f*x"
28          % (fisher.lam[1]/fisher.lam[0]))
29
30     # d)
31     fisher.showHist(save="A16d_10000.pdf")
32
33     # e)
34     # Schnitte berechnen
35     fisher.cut()
36     # und anzeigen bzw. speichern
37     fisher.showCuts(save="A16e_10000.pdf")
38
39     # f)
40     fisher.showSignalToBackground(save="A16f_10000.pdf")
41     # g)
42     fisher.showSignificance(save="A16g_10000.pdf")
43
44     # h)
45     fisher.p[0] = p0_1000
46     fisher.calc()
47     print("Mittelwerte für p_0_1000:", fisher.m)
48     print("kombinierte Kovarianzmatrix für p_0_1000:", fisher.sw)
49     print("Geradengleichung der Projektion für P_0_1000: g(x) = %1.4f*x"
50          % (fisher.lam[1]/fisher.lam[0]))
51     fisher.showHist(save="A16d_1000.pdf")
52     fisher.cut()
53     fisher.showCuts(save="A16e_1000.pdf")
54     fisher.showSignalToBackground(save="A16f_1000.pdf")
55     fisher.showSignificance(save="A16g_1000.pdf")
```

```

1 ./blatt05/Bange_Burkowitz_Harnisch/aufg18.py
2
3 from sklearn.datasets import make_blobs
4 from sklearn.decomposition import PCA
5 import matplotlib.pyplot as plt
6 from pylab import rcParams
7
8 rcParams['figure.figsize'] = 10, 5.8
9 rcParams['legend.numpoints'] = 1
10
11
12 # _____ Aufgabe 18 _____
13 def aufg18():
14     samples, mask = make_blobs(n_samples=1000, centers=2,
15                               n_features=4, random_state=0)
16     plt.scatter(samples.T[0][mask == 0],
17                 samples.T[1][mask == 0], marker=".", label="$p_0$")
18     plt.scatter(samples.T[0][mask == 1],
19                 samples.T[1][mask == 1], marker=".", label="$p_1$")
20     plt.xlabel("$x_1$")
21     plt.ylabel("$x_2$")
22     plt.legend()
23     plt.savefig("A18_scatter.pdf")
24     plt.clf()
25     pca = PCA()
26     transformed = pca.fit_transform(samples)
27     print("Eigenwerte der Kovarianzmatrix:", pca.explained_variance_)
28     for i, xp in enumerate(transformed.T):
29         plt.hist(xp[mask == 0], bins=25, label="$p_0$")
30         plt.hist(xp[mask == 1], bins=25, label="$p_1$", alpha=0.5)
31         plt.legend()
32         plt.xlabel("$x_{%i}$" % (i + 1))
33         plt.ylabel("$s_n$")
34         # plt.show()
35         plt.savefig("A18_hist_%i.pdf" % (i + 1))
36         plt.clf()
37     plt.scatter(transformed.T[0][mask == 0],
38                 transformed.T[1][mask == 0], marker=".", label="$p_0$")
39     plt.scatter(transformed.T[0][mask == 1],
40                 transformed.T[1][mask == 1], marker=".", label="$p_1$")
41     plt.xlabel("$x_1$")
42     plt.ylabel("$x_2$")
43     plt.legend()
44     plt.savefig("A18_transformed_scatter.pdf")
45
46 # _____ Main _____
47 if __name__ == '__main__':
48     aufg18()

```

```

1 ./blatt05/Bange_Burkowitz_Harnisch/fisher.py
2
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6
7 class Fisher(object):
8     def __init__(self, signal=None, background=None):
9         """
10             Initialisiert alle verwendeten Klassen Attribute.
11             Falls signal und background gesetzt sind wird die Fisher
12             Diskriminanzanalyse direkt durchgeführt.
13
14             Parameter
15
16             signal: Signalteil der Trainingsdaten 2D Array mit x- und y-Werten der
17             Punkte
18             background: Hintergrundsteil der Trainingsdaten 2D Array mit x- und
19             y-Werten der Punkte
20             """
21             # Zu trennende Populationen

```

```

50     self.p = [signal, background]
51
52     # Vektor mit lambda_cut Werten für die Schnitte
53     self._ls = None
54     # Array mit den projezierten Koordinaten beider Populationen
55     self._proj = None
56
57     # Vektor der Fisher Diskriminante
58     self.lam = None
59     # Mittelwerte der Populationen
60     self.m = None
61     # Kombinierte Kovarianzmatrix
62     self.sw = None
63
64     # Arrays die für alle Schnitte in self._ls die Anzahl der jeweiligen
65     # tp, tn, fp, und fn halten
66     self.tp = None
67     self.tn = None
68     self.fp = None
69     self.fn = None
70
71     # Berechnung durchführen, falls signal und background übergeben
72     if signal is not None and background is not None:
73         self.calc()
74
75     def calc(self):
76         """
77             Führt die Fisher Diskriminanzanalyse durch
78         """
79         if self.p[0] is None or self.p[1] is None:
80             raise ValueError("Es müssen Populationen gegeben sein.")
81         # Die Variablennamen halten sich ans Skript
82         # Streuung
83         self.sw = np.matrix(np.add(np.cov(self.p[0])*len(self.p[0] - 1),
84                               np.cov(self.p[1])*len(self.p[1] - 1)))
85
86         # Mittelwerte
87         self.m = np.matrix([[np.average(self.p[0][0]),
88                             np.average(self.p[0][1])],
89                             [np.average(self.p[1][0]),
90                             np.average(self.p[1][1])]])
91
92         # Fisher Schnitt berechnen
93         self.lam = np.array((self.sw.I*(self.m[0] - self.m[1]).T).T)[0]
94
95         # Projektion durchführen
96         self._proj = []
97         self._proj.append(np.add(np.multiply(self.p[0][0], self.lam[0]),
98                               np.multiply(self.p[0][1], self.lam[1])))
99         self._proj.append(np.add(np.multiply(self.p[1][0], self.lam[0]),
100                               np.multiply(self.p[1][1], self.lam[1])))
101        if np.average(self._proj[0]) < np.average(self._proj[1]):
102            self._proj = -self._proj
103            self.lam = -self.lam
104
105    def show(self, save=False):
106        """
107            Zeigt einen Scatterplot der Populationen zusammen mit der berechneten
108            Fisher Diskriminante an
109
110            Parameter
111
112            save: False zeigt an, ansonsten Dateiname der zu speichernden Datei
113        """
114        if self.lam is None or self.p[0] is None or self.p[1] is None:
115            raise ValueError("Es muss zuerst eine Diskriminante \
116                            berechnet werden.")
117        fig = plt.figure()
118        ax = fig.add_subplot(111)
119        ax.scatter(self.p[0][0], self.p[0][1], marker=".", s=2.5,
120                  alpha=0.5, label=r"$\mathbf{Sp\_0S}$")

```

```

91     ax.scatter(self.p[1][0], self.p[1][1], marker=".", s=2.5,
92                 alpha=0.5, label=r"$p_1$")
93     ax.set_xlim(ax.get_xlim())
94     ax.set_ylim(ax.get_ylim())
95     x = np.linspace(ax.get_xlim()[0], ax.get_xlim()[1], 1000)
96     ax.plot(x, self.lam[1]/self.lam[0]*x)
97     ax.set_xlabel(r"$x_1$")
98     ax.set_ylabel(r"$x_2$")
99     ax.legend()
100    if save:
101        fig.savefig(save)
102    else:
103        plt.show()
104
105    def showHist(self, bins=25, save=False):
106        """
107            Zeigt das Histogramm der Projektion an
108
109            Parameter
110            -----
111            bins: Anzahl der bins
112            save: False zeigt an, ansonsten Dateiname der zu speichernden Datei
113            """
114            if self._proj is None:
115                raise ValueError("Es muss zuerst eine Diskriminante \
116                                berechnet werden.")
117            fig = plt.figure()
118            ax = fig.add_subplot(111)
119
120            ax.hist(self._proj[0], bins=bins, label=r"$p_0$")
121            ax.hist(self._proj[1], bins=bins, label=r"$p_1$",
122                    alpha=0.6)
123            ax.set_xlabel(r"$\lambda$")
124            ax.set_ylabel(r"$n$")
125            ax.legend()
126            if save:
127                fig.savefig(save)
128            else:
129                plt.show()
130
131    def cut(self, nl=1000):
132        """
133            Führt nl gleichverteilte Schnitte durch
134
135            Parameter
136            -----
137            nl: Anzahl der durchzuführenden Schnitte
138            """
139            if self._proj is None:
140                raise ValueError("Es muss zuerst eine Diskriminante \
141                                berechnet werden.")
142            p = [np.sort(self._proj[0])[::-1], np.sort(self._proj[1])[::-1]]
143            self._ls = np.linspace(max(p[0][0], p[1][0]),
144                                  min(p[0][-1], p[1][-1]), nl)
145            self.tp = np.zeros(nl)
146            self.fp = np.zeros(nl)
147            self.fn = np.zeros(nl)
148            self.tn = np.zeros(nl)
149            j = 0
150            k = 0
151            for n, l in enumerate(self._ls):
152                while (j < len(p[0])) and (p[0][j] >= l):
153                    j += 1
154                while k < len(p[1]) and p[1][k] >= l:
155                    k += 1
156                self.tp[n] = j
157                self.fn[n] = len(p[0]) - j
158                self.fp[n] = k
159                self.tn[n] = len(p[1]) - k
160
161    def showCuts(self, save=False):

```

```

162     """
163     Zeigt Reinheit, Effizienz und Genauigkeit für die berechneten Schnitte
164     an bzw. speichert diese wenn save gesetzt ist.
165
166     Parameter
167     -----
168     save: False zeigt an, ansonsten Dateiname der zu speichernden Datei
169     """
170     if self._ls is None:
171         raise ValueError("Die Schnitte müssen zuerst durchgeführt werden.")
172
173     fig = plt.figure()
174     ax = fig.add_subplot(111)
175
176     ax.plot(self._ls, self.tp/(self.tp + self.fp), label="Reinheit")
177     ax.plot(self._ls, self.tp/(self.tp + self.fn), label="Effizienz")
178     ax.plot(self._ls, (self.tp + self.tn) /
179             (self.tp + self.tn + self.fp + self.fn), label="Genauigkeit")
180     ax.set_xlim(self._ls[-1], self._ls[0])
181     ax.legend()
182     ax.set_xlabel("$\lambda_{\mathbf{mathrm{cut}}}$")
183     if save:
184         fig.savefig(save)
185     else:
186         plt.show()
187
188     def showSignificance(self, save=False):
189         """
190         Zeigt die Signifikanz für die berechneten Schnitte an bzw. speichert
191         diese wenn save gesetzt ist.
192
193         Parameter
194         -----
195         save: False zeigt an, ansonsten Dateiname der zu speichernden Datei
196         """
197         if self._ls is None:
198             raise ValueError("Die Schnitte müssen zuerst durchgeführt werden.")
199
200         fig = plt.figure()
201         ax = fig.add_subplot(111)
202
203         ax.plot(self._ls, (self.tp + self.fp) / (tp + fn))
204         np.sqrt(len(self._proj[0]) + len(self._proj[1]))
205         ax.set_xlim(self._ls[-1], self._ls[0])
206         ax.set_xlabel("$\lambda_{\mathbf{mathrm{cut}}}$")
207         ax.set_ylabel(r"$\frac{S}{\sqrt{S+B}}$")
208         if save:
209             fig.savefig(save)
210         else:
211             plt.show()
212
213     def showSignalToBackground(self, save=False):
214         """
215         Zeigt das Signal-Hintergrund-Verhältnis für die berechneten Schnitte an
216         bzw. speichert diese wenn save gesetzt ist.
217
218         Parameter
219         -----
220         save: False zeigt an, ansonsten Dateiname der zu speichernden Datei
221         """
222         if self._ls is None:
223             raise ValueError("Die Schnitte müssen zuerst durchgeführt werden.")
224
225         fig = plt.figure()
226         ax = fig.add_subplot(111)
227
228         with np.errstate(divide='ignore'):
229             ax.plot(self._ls, (self.tp + self.fp)/(self.tn + self.fn))
230
231         ax.set_xlim(self._ls[-1], self._ls[0])
232         ax.set_xlabel("$\lambda_{\mathbf{mathrm{cut}}}$")

```

```
233     ax.set_ylabel(r"\frac{S}{B}")
234     ax.set_yscale("log")
235     if save:
236         fig.savefig(save)
237     else:
238         plt.show()
```

Zeit	Raum	Abgabe im Moodle; Mails mit Betreff: [SMD1718]
Di. 10-12	CP-03-150	philipp2.hoffmann@udo.edu und jan.soedingrekso@udo.edu
Di. 16-18	P1-02-110	felix.neubuerger@udo.edu und tobias.hoinka@udo.edu
Di. 16-18	CP-03-150	simone.mender@udo.edu und maximilian.meier@udo.edu

Aufgabe 19: *k-NN Klassifikation*

7 P.

- Worauf müssen Sie bei einem *k*-NN-Algorithmus achten, wenn die Attribute sich stark in ihren Größenordnungen unterscheiden?
- Warum bezeichnet man den *k*-NN als sogenannten „lazy learner“? Wie sind die Laufzeiten für Lern- und Anwendungs-Phase? Wie sind sie im Vergleich zu anderen Algorithmen wie bspw. einem Random Forest?
- Implementieren Sie einen *k*-NN Algorithmus zur Klassifikation von Ereignissen. Die Funktion soll hierbei das Trainingssample, die Label des Trainingssamples, die zu klassifizierenden Daten, sowie das *k* übergeben bekommen. Die Rückgabe sollen die ermittelten Label für die Datenereignisse sein.

Vorgehen: Für jedes zu klassifizierende Ereignis:

- Berechnung der Abstände zu allen Punkten des Trainingssamples.
 - Bestimmung der *k* Trainingsevents mit dem kleinsten Abstand (Hinweis: Ermitteln Sie nur die Indizes der Ereignisse, statt das Array an sich zu sortieren).
Tipp: Die Python-Funktion `numpy.argsort()` ist hilfreich.
 - Bestimmung des Labels, das in diesen Ereignissen am häufigsten vorkommt.
- d) Wenden Sie Ihren Algorithmus auf das Neutrino Monte-Carlo von Blatt 3 an. Benutzen Sie die im Moodle zur Verfügung gestellte Datei `NeutrinoMC.hdf5`.
- Nutzen Sie die Attribute `AnzahlHits`, `x` und `y`.
 - Setzen Sie *k* = 10.
 - Nutzen Sie je 5000 Ereignisse als Trainingsset.
 - Das Testset soll aus 20 000 Untergrund- und 10 000 Signalevents bestehen.

Bestimmen Sie Reinheit, Effizienz und Signifikanz.

- e) Was ändert sich, wenn Sie `log10(AnzahlHits)` statt `AnzahlHits` nutzen?

- f) Was ändert sich, wenn Sie $k = 20$ statt $k = 10$ verwenden?

Aufgabe 20: *Multivariate Regression*

7 P.

In dieser Aufgabe sollen Sie eine 2 dimensionale multivariate Regression mit *sklearn* durchführen und die Ergebnisse anschaulich darstellen.

- Erstellen sie ein Dataframe mit 10^5 uniform zwischen 0 und 1 verteilten Zufallszahlen x_1, x_2 .
- Berechnen sie aus diesen Attributen ein drittes Attribut x_3 mit der Funktionsvorschrift:

$$x_3 = 15 \sin(4\pi x_1) + 60(x_2 - 0.5)^2$$

Addieren Sie auf diese Zahl eine standardnormalverteilte Zufallszahl, um Rauschen zu simulieren. Das x_3 Attribut ist von nun an Ihr Zielattribut.

- Teilen Sie das Dataframe in einen Trainings- und Test-Datensatz auf.
- Wählen Sie einen Random-Forest-Regressor mit 200 Bäumen und trainieren Sie diesen auf dem Trainingsdatensatz um x_3 zu schätzen.
- Stellen Sie die erstellten Daten und die Vorhersagen des Regressors in einem dreidimensionalen Plot und mehreren 2 dimensionalen Projektionen dar um die Vorhersage mit der Wahrheit zu vergleichen. Geben sie außerdem den *mean-squared-error* der Vorhersage zu den wahren Werten an.
- Erstellen Sie einen weiteren Datensatz, bei dem x_1 und x_2 uniform verteilte Zufallszahlen zwischen 1 und 2 sind. Was für Probleme können hier auftreten und was ist die Vorhersage des Regressors? Stellen Sie das Ergebnis wie in Aufgabe e) dar.

Aufgabe 21: *Binärer Entscheidungsbaum: Die erste Entscheidung*

6 P.

Sie haben einen Datensatz wie er in Tabelle 1 gegeben ist. Hierbei ist

- Temperatur: Temperatur in Grad Celsius.
- Wettervorhersage: Wetterqualität (0: schlecht, 1: normal, 2: gut).
- Luftfeuchtigkeit: Luftfeuchtigkeit in Prozent.
- Wind: Aussage, ob es gerade windig ist.
- Fußball: Lohnt es sich Fußball spielen zu gehen?

Hierbei ist das Zielattribut, welches man bestimmen will, die Entscheidung, ob es sich lohnt Fußball spielen zu gehen. In dieser Aufgabe sollen Sie zu diesem Zweck den ersten Schnitt eines *binären* Entscheidungsbaumes nachvollziehen.

- a) Berechnen Sie per Hand die Entropie der Wurzel (des Baumes).
- b) Berechnen Sie per Hand den Informationsgewinn, falls ein Schnitt auf dem Attribut *Wind* durchgeführt wird.
- c) Berechnen Sie für die verbleibenden Attribute den Informationsgewinn in Abhängigkeit von verschiedenen Schnitten und plotten Sie den Informationsgewinn in Abhängigkeit der jeweiligen Schnitte.
- d) Welches Attribut eignet sich am besten zum Trennen der Daten?

Tabelle 1: Datensatz: „Soll ich Fußballspielen gehen?“

Temperatur / °C	Wettervorhersage	Luftfeuchtigkeit / %	Wind	Fußball
29,4	2	85	False	False
26,7	2	90	True	False
28,3	1	78	False	True
21,1	0	96	False	True
20	0	80	False	True
18,3	0	70	True	False
17,8	1	65	True	True
22,2	2	95	False	False
20,6	2	70	False	True
23,9	0	80	False	True
23,9	2	70	True	True
22,2	1	90	True	True
27,2	1	75	False	True
21,7	0	80	True	False

Aufgabe 19

a)

Wenn eine ungewichtete Norm wie die euklidische Norm für die Abstandsberechnung verwendet wird und sich die Attribute in der Größenordnung unterscheiden, dann haben die größeren Attribute in der Abstandsberechnung ein deutlich höheres Gewicht. Es ist praktisch nur noch der Abstand des größten Attributs relevant, der Rest spielt keine Rolle mehr. Bspw. Abstand der zwei Punkte $a = (1, 10000, 5)^T$ und $b = (3, 10232, 9)^T$:

$$\sqrt{(a - b)^2} \approx b_2 - a_2 = 232. \quad \checkmark \quad (1)$$

b)

Der k -NN-Algorithmus wird als *lazy learner* bezeichnet, weil praktisch der gesamte Rechenaufwand in die Anwendungs-Phase gelegt wird. Anders als bspw. beim Random Forest Verfahren, bei dem in der Lernphase rechenaufwendig Bäume erstellt werden, die sich in der Anwendungsphase mit relativ geringem Rechenaufwand auswerten lassen. \checkmark

d)

Reinheit:	0,8340
Effizienz:	0,9644
Genauigkeit:	0,9242
Signifikanz:	<u>66,76</u> <i>falsche Signifikanz</i> $\frac{t_p}{\sqrt{t_p + f_p}}$ <i>(wie viele wahr)</i>

e)

Reinheit:	0,8712
Effizienz:	0,9813
Genauigkeit:	0,9454
Signifikanz:	<u>65,03</u> <i>aber sonst gut</i>

f)

Reinheit:	0,8590
Effizienz:	0,9859
Genauigkeit:	0,9414
Signifikanz:	<u>66,26</u> <i>7/7</i>

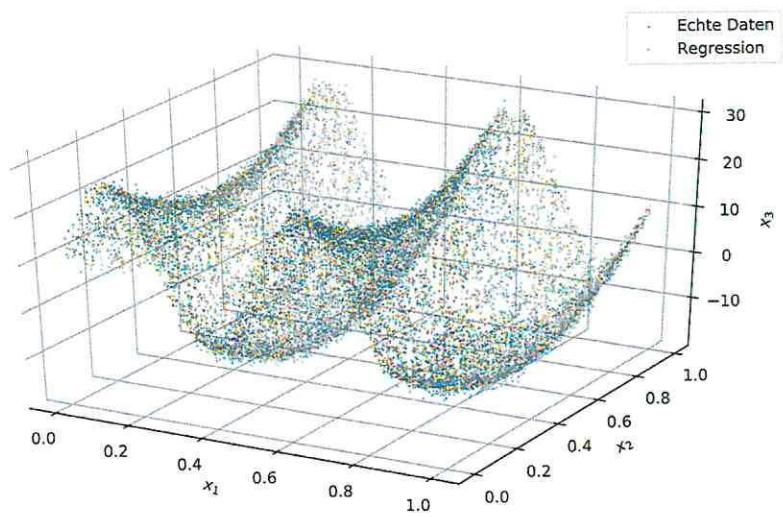


Abbildung 1: 3D-Plot, mit $x_1, x_2 \in \{0, 1\}\right\}$.

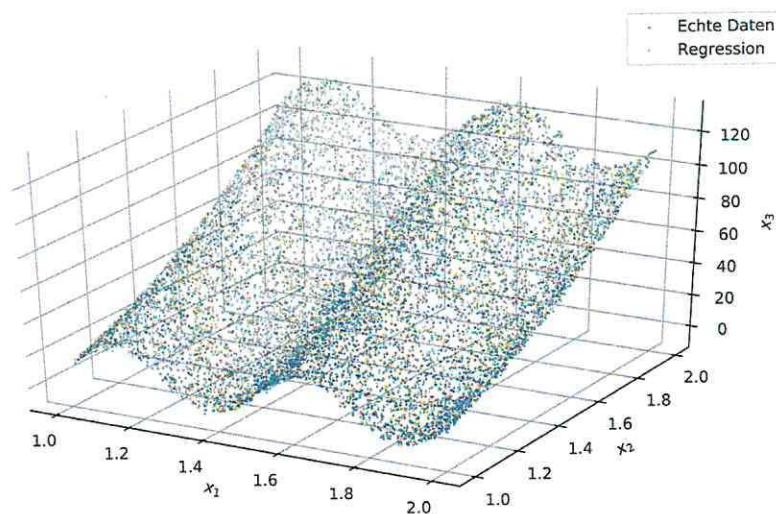


Abbildung 2: 3D-Plot, mit $x_1, x_2 \in \{1, 2\}\right\}$.

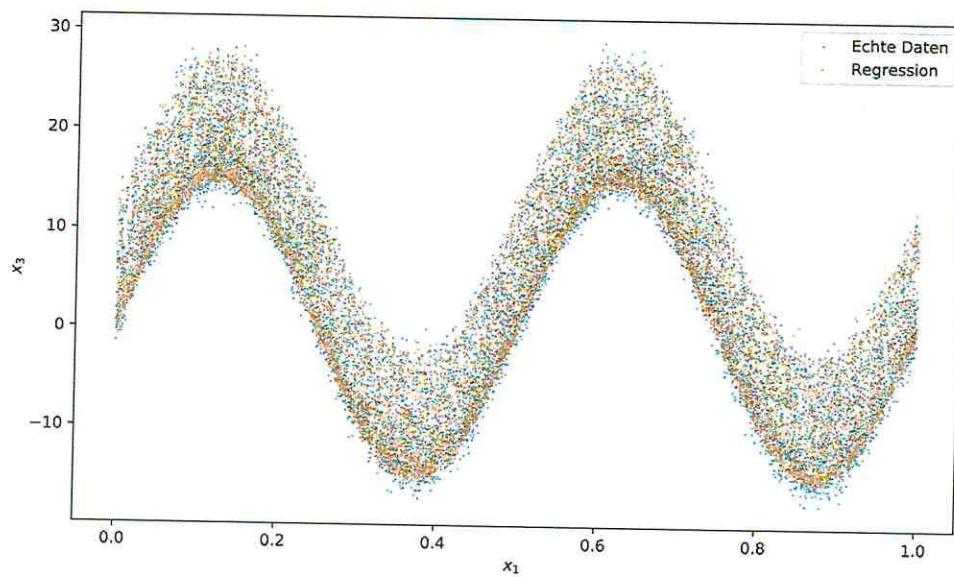


Abbildung 3: Projektion in die x_1, x_3 -Ebene, mit $x_1, x_2 \in \{0, 1\}$.

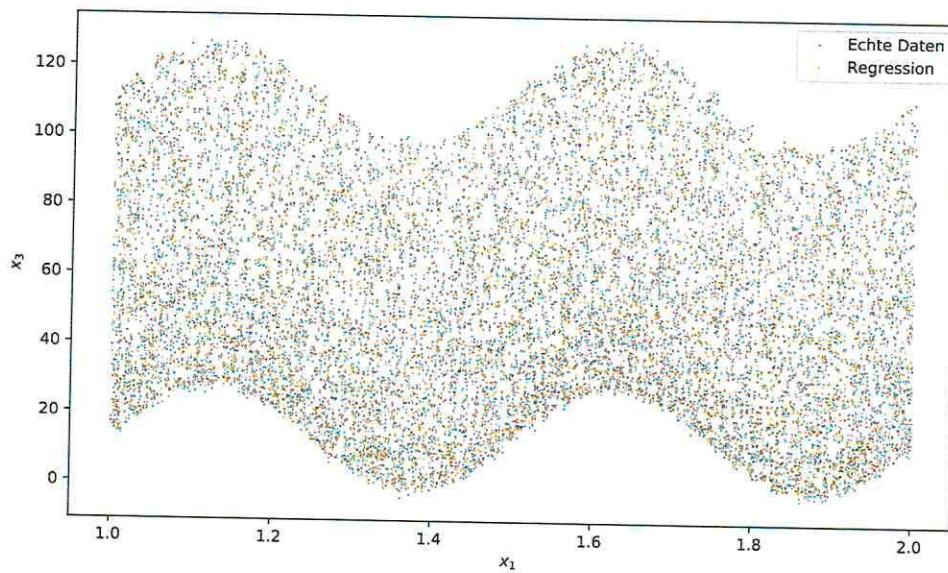


Abbildung 4: Projektion in die x_1, x_3 -Ebene, mit $x_1, x_2 \in \{1, 2\}$.

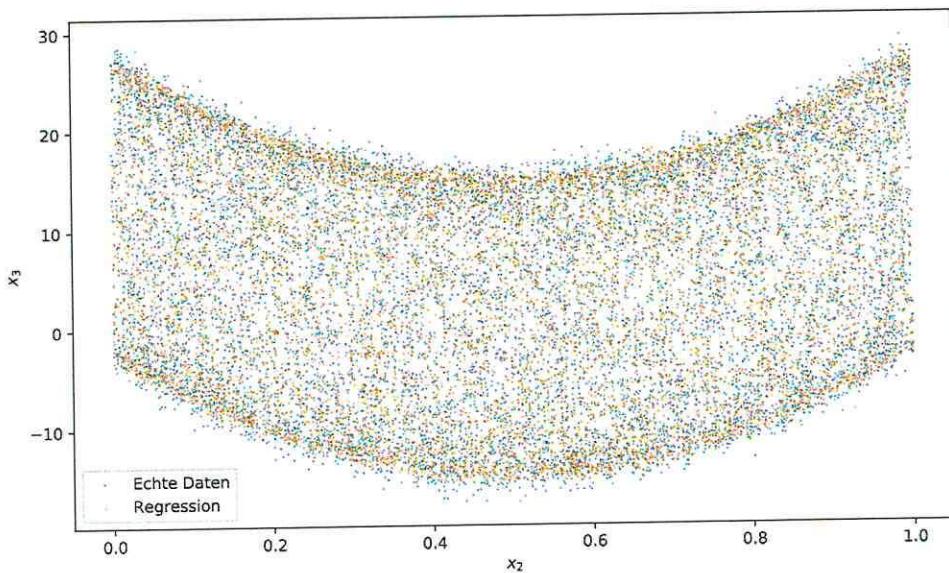


Abbildung 5: Projektion in die x_2, x_3 -Ebene, mit $x_1, x_2 \in \{0, 1\}$.

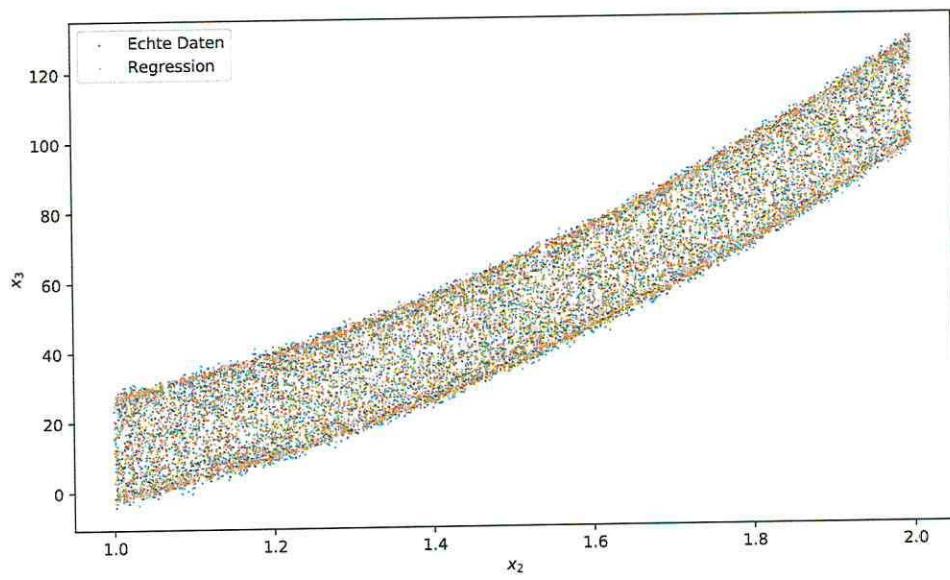


Abbildung 6: Projektion in die x_2, x_3 -Ebene, mit $x_1, x_2 \in \{1, 2\}$.

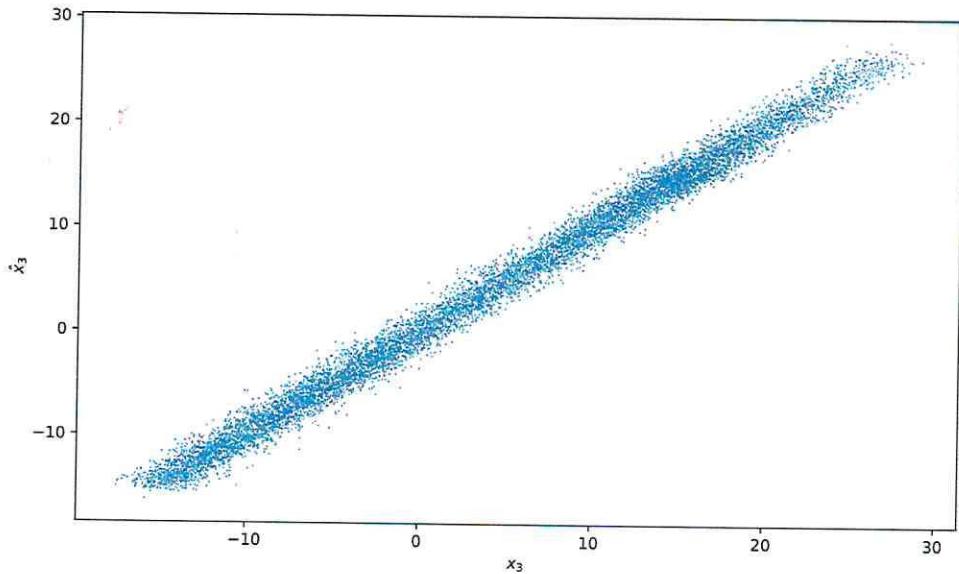


Abbildung 7: Geschätzter x_3 -Wert gegen den echten x_3 -Wert (optimal ist eine Gerade mit Steigung 1), mit $x_1, x_2 \in \{0, 1\}$.

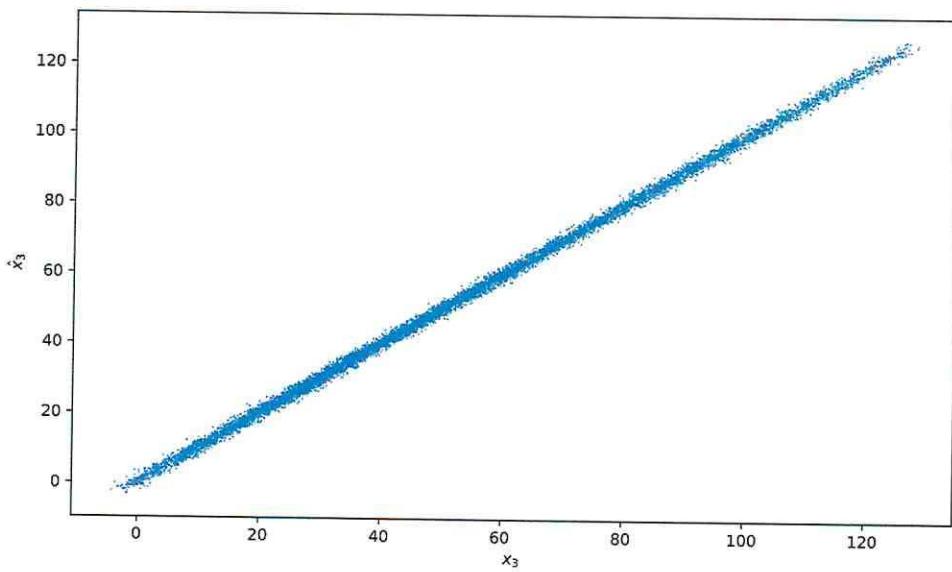


Abbildung 8: Geschätzter x_3 -Wert gegen den echten x_3 -Wert (optimal ist eine Gerade mit Steigung 1), mit $x_1, x_2 \in \{1, 2\}$.

Aufgabe 20

e, f)

$$\begin{array}{ll} \text{MSE, } x_1, x_2 \in \{0, 1\}: & 1,245 \\ \text{MSE, } x_1, x_2 \in \{1, 2\}: & 1,404 \end{array}$$

*Schreibt mir sehr klein
dafür, dass x_1 und x_2 zw. 1 und 2
gezogen werden.*

*Ah, ihr habt einen neuen Forest
euch gebracht; dabei sollte
ihr den gleichen verwenden!*

7/7

21. a)

$$H(Y) = - \sum_{z \in Z} P(Y=z) \log_2(P(Y=z))$$

$$Z = \{ \text{False}, \text{True} \}$$

$$P(Y=\text{False}) = \frac{5}{14}, \quad P(Y=\text{True}) = \frac{9}{14}$$

$$\Rightarrow H(Y) = -\frac{5}{14} \log_2\left(\frac{5}{14}\right) - \frac{9}{14} \log_2\left(\frac{9}{14}\right) \approx 0.9903$$

b) Entropie vorher: $H(Y)$

Entropie nachher: $H(Y|X)$

$$\Rightarrow H(Y|X) = - \sum_{m \in M} P(X=m) \sum_{z \in Z} P(Y=z|X=m) \times \log(P(Y=z|X=m))$$

$$M = \{ \text{False}, \text{True} \}$$

$$P(X=\text{False}) = \frac{8}{14}, \quad P(X=\text{True}) = \frac{6}{14}$$

$$P(Y=\text{False}|X=\text{False}) = \frac{2}{\cancel{10}}$$

$$P(Y=\text{False}|X=\text{True}) = \frac{3}{\cancel{14}6}$$

$$P(Y=\text{True}|X=\text{False}) = \frac{6}{\cancel{14}8}$$

$$P(Y=\text{True}|X=\text{True}) = \frac{3}{\cancel{14}6}$$

$$\Rightarrow H(Y|X) = -\frac{8}{14} \left(\frac{2}{14} \log_2\left(\frac{2}{14}\right) + \frac{6}{14} \log_2\left(\frac{6}{14}\right) \right) - \frac{6}{14} \left(\frac{3}{14} \log_2\left(\frac{3}{14}\right) + \frac{3}{14} \log_2\left(\frac{3}{14}\right) \right) \quad \left. \right\} \text{S.O.}$$

$$\Rightarrow H(Y|X) \approx 0.936 \Rightarrow$$

$$\Rightarrow H(Y) - H(Y|X) \approx 0.00356$$

Aufgabe 21

c)

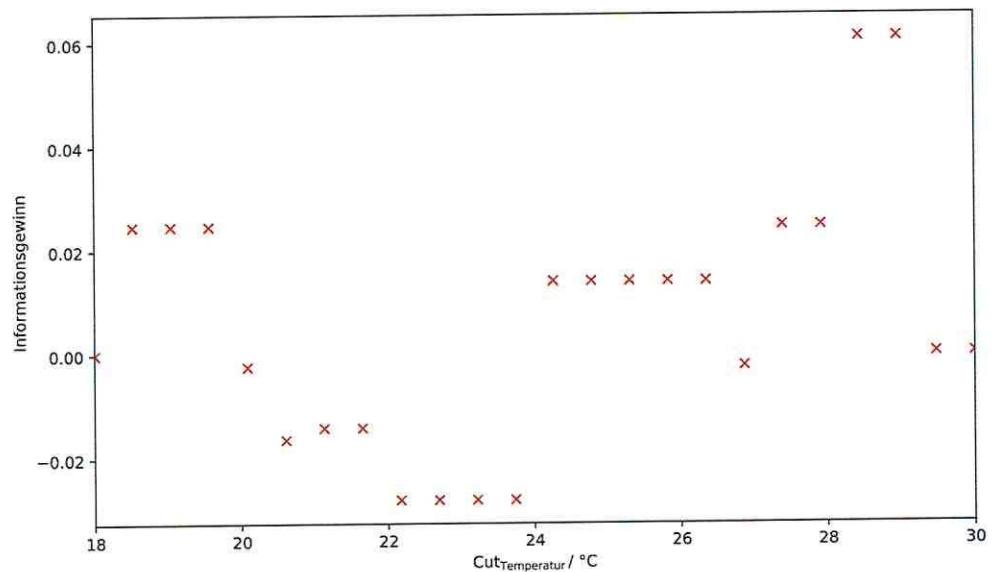


Abbildung 9: Untersuchung des Attributes "Temperatur".

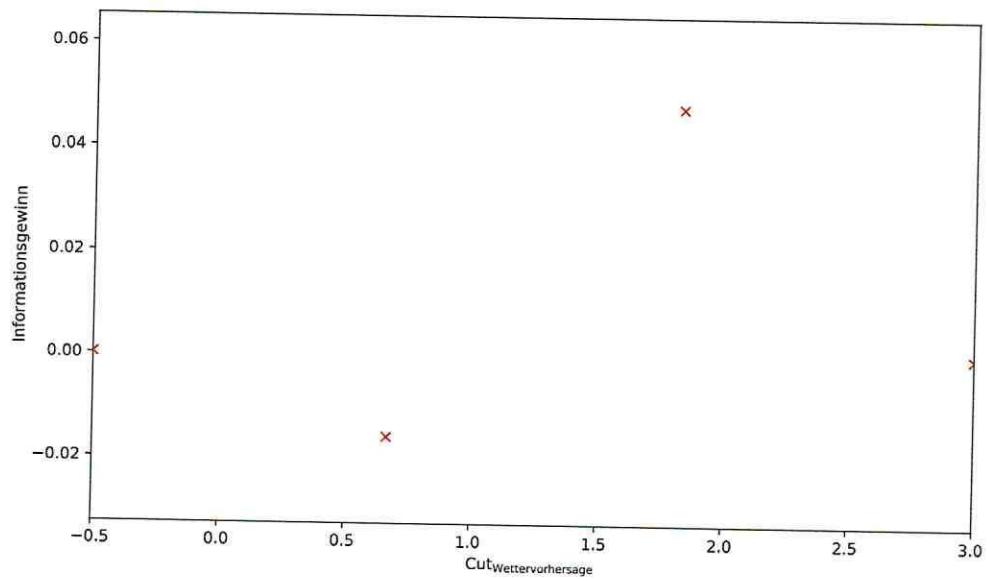


Abbildung 10: Untersuchung des Attributes "Wettervorhersage".

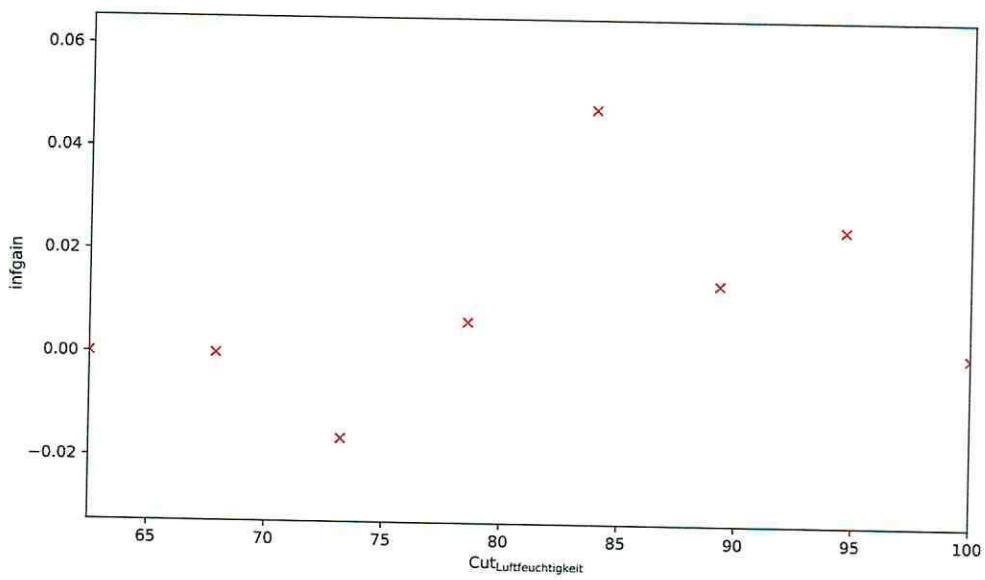


Abbildung 11: Untersuchung des Attributes "Luftfeuchtigkeit".

d)

Da bei der Temperatur der höchste Informationsgewinn vorliegt, ist das das beste Attribut zum Trennen der Daten.

✓
55/6

Code für blatt06

Bange, Burkowitz, Harnisch

11. Dezember 2017

```
.../blatt06/Bange_Burkowitz_Harnisch/aufg19.py

1 import numpy as np
2 import pandas as pd
3 import os
4 from scipy.spatial.distance import cdist
5 # _____ Funktionen _____
6
7
8 def kNN(x_train, y_train, x_test, k):
9     """
10     Implementierung des kNN Lerners.
11
12     Parameters
13
14     x_train: Trainingsdatensample
15     y_train: Labels der Trainingsdaten
16     x_test: zu klassifizierende Daten
17
18     Returns
19
20     Die Labels der zu klassifizierenden Daten
21     """
22     if k > len(x_train):
23         raise ValueError("k kann nicht so groß sein")
24
25     idx = np.argsort(cdist(x_test, x_train), axis=1)
26     y_test = np.empty(len(x_test))
27
28     for i in range(len(x_test)):
29         (values, counts) = np.unique(y_train[idx[i]][:k], return_counts=True)
30         y_test[i] = values[np.argmax(counts)]
31     return y_test
32
33
34 def evaluate(x_train, y_train, x_test, y_test, k, fsave=None):
35     if fsave is not None:
36         if not os.path.isfile(fsave):
37             y_test_knn = kNN(x_train, y_train, x_test, k)
38             np.save(fsave, y_test_knn)
39         else:
40             y_test_knn = np.load(fsave)
41     else:
42         y_test_knn = kNN(x_train, y_train, x_test, k)
43
44     tp = np.count_nonzero(y_test_knn[:len(signal.index) - 5000])
45     fn = len(signal.index) - 5000 - tp
46     fp = np.count_nonzero(y_test_knn[len(signal.index) - 5000:])
47     tn = len(background.index) - 5000 - fp
48     print("Reinheit =", tp/(tp + fp))
49     print("Effizienz =", tp/(tp + fn))
50     print("Genauigkeit =", (tp + tn)/len(y_test))
51     print("Signifikanz =", (tp + fp)/np.sqrt(len(y_test)))
52     return y_test_knn
53
54
55 # _____ Main _____
56 if __name__ == '__main__':
57     signal = pd.read_hdf("NeutrinoMC.hdf5", "Signal")
```

```

58 background = pd.read_hdf("NeutrinoMC.hdf5", "Background")
59
60 signal.drop(columns=["Energy"], inplace=True)
61 x_train = signal.iloc[:5000].append(background.iloc[:5000])
62 y_train = np.concatenate((np.ones(5000), np.zeros(5000)))
63 x_test = signal.iloc[5000:].append(background.iloc[5000:])
64 y_test = np.concatenate((np.ones(len(signal.index) - 5000),
65 np.zeros(len(background.index) - 5000)))
66
67 print("Teilaufgabe d)")
68 evaluate(x_train, y_train, x_test, y_test, 10, "A19d.npy")
69
70 print("Teilaufgabe e)")
71 with np.errstate(divide='ignore'):
72     x_train["NumberOfHits"] = np.log10(x_train["NumberOfHits"])
73     x_test["NumberOfHits"] = np.log10(x_test["NumberOfHits"])
74
75 evaluate(x_train, y_train, x_test, y_test, 10, "A19e.npy")
76
77 print("Teilaufgabe f)")
78 evaluate(x_train, y_train, x_test, y_test, 20, "A19f.npy")

```

..../blatt06/Bange_Burkowitz_Harnisch/aufg20.py

```

1 import os
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 from sklearn.externals import joblib
6 from mpl_toolkits.mplot3d import Axes3D
7 from sklearn.metrics import mean_squared_error
8 from sklearn.model_selection import train_test_split
9 from sklearn.ensemble import RandomForestRegressor
10 from pylab import rcParams
11
12 rcParams['figure.figsize'] = 10, 5.8
13 rcParams['legend.numpoints'] = 1
14
15
16 def doEverything(df, fname):
17     train, test = train_test_split(df, test_size=0.1)
18
19     if os.path.isfile("{} .pkl".format(fname)):
20         rf = joblib.load("{} .pkl".format(fname))
21     else:
22         rf = RandomForestRegressor(n_estimators=200)
23         rf.fit(train.drop(columns=["x3"]), train["x3"])
24         joblib.dump(rf, "{} .pkl".format(fname))
25     x3_predict = rf.predict(test.drop(columns=["x3"]))
26     print("MSE =", mean_squared_error(test["x3"], x3_predict))
27
28     fig = plt.figure()
29     ax = fig.add_subplot(111, projection="3d")
30     ax.scatter(test["x1"], test["x2"], test["x3"], marker=".", s=0.5,
31                label="Echte Daten")
32     ax.scatter(test["x1"], test["x2"], x3_predict, marker=".", s=0.5,
33                label="Regression")
34     ax.set_xlabel("$x_1$")
35     ax.set_ylabel("$x_2$")
36     ax.set_zlabel("$x_3$")
37     plt.legend()
38     fig.savefig("A20_3D_{}.pdf".format(fname))
39     # plt.show()
40     plt.clf()
41
42     plt.scatter(test["x1"], test["x3"], marker=".", s=0.5, label="Echte Daten")
43     plt.scatter(test["x1"], x3_predict, marker=".", s=0.5, label="Regression")
44     plt.xlabel("$x_1$")
45     plt.ylabel("$x_3$")
46     plt.legend()
47     # plt.show()
48     plt.savefig("A20_x1_x3_{}.pdf".format(fname))

```

Ihr setzt in f)
den Forest aus d)e)
nur anwenden,
keinen neuen bauen.

```

48     plt.clf()
49
50
51     plt.scatter(test["x2"], test["x3"], marker=".", s=0.5, label="Echte Daten")
52     plt.scatter(test["x2"], x3_predict, marker=".", s=0.5, label="Regression")
53     plt.xlabel("$x_2$")
54     plt.ylabel("$x_3$")
55     plt.legend()
56     # plt.show()
57     plt.savefig("A20_x2_x3_{}.pdf".format(fname))
58     plt.clf()
59
60
61     plt.scatter(test["x3"], x3_predict, marker=".", s=0.5)
62     plt.xlabel("$x_3$")
63     plt.ylabel("$\hat{x}_3$")
64     # plt.show()
65     plt.savefig("A20_x3_x3_{}.pdf".format(fname))
66     plt.clf()
67
68 # ----- Main -----
69 if __name__ == '__main__':
70     np.random.seed(1234)
71
72     print("Teilaufgabe e)")
73     df = pd.DataFrame({"x1": np.random.uniform(size=100000,
74                         "x2": np.random.uniform(size=100000)})
75     df["x3"] = 15*np.sin(4*np.pi*df["x1"]) + 50*(df["x2"] - 0.5)**2 + \
76         np.random.normal(size=100000)
77     doEverything(df, "e")
78
79     print("Teilaufgabe f)")
80     df = pd.DataFrame({"x1": np.random.uniform(1, 2, size=100000),
81                         "x2": np.random.uniform(1, 2, size=100000)})
82     df["x3"] = 15*np.sin(4*np.pi*df["x1"]) + 50*(df["x2"] - 0.5)**2 + \
83         np.random.normal(size=100000)
84     doEverything(df, "f")

```

```

.../blatt06/Bange_Burkowitz_Harnisch/aufg21.py
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from pylab import rcParams
5
6 rcParams['figure.figsize'] = 10, 5.8
7 rcParams['legend.numpoints'] = 1
8
9
10 def infgain(data, target, cut):
11     cut = np.atleast_1d(cut)
12     p_ytrue = np.count_nonzero(target) / len(target)
13     p_yfalse = 1 - p_ytrue
14     H_y = -p_ytrue * np.log2(p_ytrue) - p_yfalse * np.log2(p_yfalse)
15
16     p_xhigh = np.zeros(len(cut))
17     p_xlow = np.zeros(len(cut))
18     p_xhigh_ytrue = np.zeros(len(cut))
19     p_xhigh_yfalse = np.zeros(len(cut))
20     p_xlow_ytrue = np.zeros(len(cut))
21     p_xlow_yfalse = np.zeros(len(cut))
22     H_y_x = np.zeros(len(cut))
23     for i in range(len(cut)):
24         p_xhigh[i] = np.count_nonzero(data > cut[i]) / len(data)
25         p_xlow[i] = 1 - p_xhigh[i]
26         p_xhigh_ytrue[i] = np.count_nonzero((data >= cut[i]) & (target is True)) / (len(data) * p_xhigh[i])
27         p_xhigh_yfalse[i] = np.count_nonzero((data >= cut[i]) & (target is False)) / (len(data) * p_xhigh[i])
28         p_xlow_ytrue[i] = np.count_nonzero((data < cut[i]) & (target is True)) / (len(data) * p_xlow[i])
29         p_xlow_yfalse[i] = np.count_nonzero((data < cut[i]) & (target is False)) / (len(data) * p_xlow[i])

```

Zeit	Raum	Abgabe im Moodle; Mails mit Betreff: [SMD1718]
Di. 10-12	CP-03-150	philipp2.hoffmann@udo.edu und jan.soedingrekso@udo.edu
Di. 16-18	P1-02-110	felix.neubuerger@udo.edu und tobias.hoinka@udo.edu
Di. 16-18	CP-03-150	simone.mender@udo.edu und maximilian.meier@udo.edu

Aufgabe 22: *Lineare Klassifikation mit Softmax*

10 P.

Für eine Parameteranpassung bei der Klassifikation mittels der Softmax-Funktion muss der Gradient der Kostenfunktion für alle anzupassenden Parameter bestimmt werden. Die Kostenfunktion C ist wie aus der Vorlesung bekannt, gegeben durch:

$$C(f) = \frac{1}{m} \sum_{i=1}^m \hat{C}(f_i) = \frac{1}{m} \sum_{i=1}^m \left[-\sum_{k=1}^K \mathbf{1}(y_i = k) \log \frac{\exp(f_{k,i})}{\sum_j \exp(f_{j,i})} \right]. \quad (1)$$

Zur Ableitung der Kostenfunktion wird die Kettenregel verwendet:

$$\nabla_W \hat{C} = \sum_{k=1}^K \frac{\partial \hat{C}}{\partial f_{k,i}} \cdot \frac{\partial f_{k,i}}{\partial W} \quad (2)$$

$$\nabla_b \hat{C} = \sum_{k=1}^K \frac{\partial \hat{C}}{\partial f_{k,i}} \cdot \frac{\partial f_{k,i}}{\partial b}. \quad (3)$$

- a) Gegeben seien K Klassen und m Beispiele x_i jeweils mit M Komponenten. Welche Dimension haben die einzelnen Komponenten x_i , C , W , b , $\nabla_W \hat{C}$, $\nabla_{f_i} \hat{C}$, $\frac{\partial f_{k,i}}{\partial W}$, $\frac{\partial f_{k,i}}{\partial b}$? Unterscheiden Sie dabei auch zwischen Zeilen- und Spaltenvektoren.
- b) Zeigen Sie, dass sich für die Ableitung der Kostenfunktion nach den Scores für die Klasse a folgendes ergibt:

$$\nabla_{f_a} C(f) = \frac{1}{m} \sum_{i=1}^m \left[\frac{\exp(f_{a,i})}{\sum_j \exp(f_{j,i})} - \mathbf{1}(y_i = a) \right] \cdot \vec{e}_a \times \vec{z} \quad (4)$$

- c) Bestimmen Sie als zweiten Schritt der Kettenregel die Ableitungen von $f_{k,i}$ nach W und b mit $f_{k,i} = W_k x_i + b_k$.
- d) Implementieren Sie die lineare Klassifikation mit Softmax für die zwei Populationen P0 und P1¹. Verfahren Sie dabei wie folgt:
 - Lesen Sie die Populationen mit $PX = np.load('PX.npy')$ ein.

¹Blatt04 Aufgabe1, es befinden sich numpy-Dateien im Moodle

Aufgabe 24: *DeepLearning Kurzfragen*

5 P.

- a) Was beschreibt die Lossfunktion und wofür wird sie benötigt?
- b) Wie kann die Lossfunktion minimiert werden?
- c) Welche Funktion haben die Aktivierungsfunktionen bzw. welches Problem wird durch diese gelöst? Nennen Sie drei gängige Aktivierungsfunktionen.
- d) Was ist ein Neuron?
- e) Nennen Sie drei Anwendungsbeispiele für Neuronale Netze und beschreiben Sie kurz warum sie für diese Beispiele besonders geeignet sind.

Aufgabe 24

a)

Die Lossfunktion gibt an, wie schlecht die berechnete Vorhersage im Vergleich zu dem korrekten Ergebnis ist. Um die bestmögliche Vorhersage zu treffen, muss die Lossfunktion minimiert werden.

✓

b)

Wenn die Lossfunktion abgeleitet werden kann, kann sie minimiert werden, indem die Ableitung der Lossfunktion gleich Null gesetzt wird.

*Das Prinzip richtig, aber in Bezug
zu NN die der Aufgabe vorbei.*

c)

Aktivierungsfunktionen stellen einen Zusammenhang zwischen dem Input an einem Knoten in einem Neuronalen Netz und dem Output des Knotens her. Beispielsweise realisieren sie einen Schwellwert für den Input, ab dem eine Aktivierung des Neurons stattfindet und somit ein Output generiert wird. Aktivierungsfunktionen sind nicht-linear und ermöglichen damit durch nicht-lineare Kombination des (gewichteten) Inputs die Erzeugung nicht-linearer Entscheidungsgrenzen.

Beispiele

1. Sigmoid:

$$f(x) = \frac{1}{1 + e^{-x}}$$

2. Tangens hyperbolicus:

$$f(x) = \tanh(x)$$

3. Rectified Linear Unit (ReLU) / Softplus:

$$\begin{aligned} f(x)_{\text{ReLU}} &= \max(0, x) \\ f(x)_{\text{Softplus}} &= \ln(1 + e^x) \end{aligned}$$

4. Softmax:

$$q_k(x) = \frac{e^{f_k(x)}}{\sum_j e^{f_j(x)}}$$

✓

d)

Künstliche Neuronen sind Bestandteile eines Neuronalen Netzes. Jedes Neuron erhält Input, der zunächst (mit individuellen Gewichtungen für jedes Neuron) gewichtet wird und dann durch die Übertragungsfunktion aufsummiert wird. Das Ergebnis ist die sogenannte Netzeingabe. Zusätzlich kann für jedes Neuron ein Schwellwert festgelegt werden. Die

Netzeingabe muss dann diesen Schwellwert überschreiten, damit die Aktivierungsfunktion die Eingabe modulieren kann und somit die Ausgabe festlegen kann.

✓

e)

Allgemein sind neuronale Netze dann besonders gut geeignet, wenn man selber die Lösung nicht vernünftig mathematisch beschreiben kann. Anwendungsbeispiele für Neuronale Netze:

1. Bilderkennung: Wie definiert man einen Stuhl? Lösung: Das neuronale Netz soll sich das selber benennen.
2. Spiele: Sehr viele mögliche Parameter, Problem wird zu hochdimensional.
3. Sinnerkennung von Sprache (Siri, Cortana, Alexa,...): Ähnlich wie Bilderkennung zur Erkennung der Worte und viele mögliche Parameter, weil jeder leicht anders formuliert.

4/5

22) Andere Notation als in der Vorlesung (Transponiert)

M : Anzahl der Attribute

m : Anzahl der Beispiele bzw. Datenpunkte

K : Anzahl der Klassen

a) $x_i \in \mathbb{R}^{M \times 1}$ Beispiele, Trainingsdatenpunkte

$$C(f): \mathbb{R}^{K \times m} \rightarrow \mathbb{R}^{1 \times 1}$$

$$\hat{C}(f_i): \mathbb{R}^{K \times 1} \rightarrow \mathbb{R}^{1 \times 1}$$

$$w \in \mathbb{R}^{K \times M}$$

$$b \in \mathbb{R}^{K \times 1}$$

$$\nabla_w \hat{C} \in \mathbb{R}^{K \times M}$$

$$\nabla_b \hat{C} \in \mathbb{R}^{K \times 1}$$

$$\nabla_{f_i} \hat{C} \in \mathbb{R}^{K \times 1}$$

$$\frac{\partial f_{k,i}}{\partial w} \in \mathbb{R}^{K \times M}$$

$$\frac{\partial A}{\partial w} = \begin{pmatrix} \frac{\partial A}{\partial w_{11}} & \frac{\partial A}{\partial w_{12}} & \dots \\ \frac{\partial A}{\partial w_{21}} & \ddots & \ddots \\ \vdots & & \ddots \end{pmatrix}$$

b) $\nabla_w C(f) = \sum_{k,i} \frac{\partial C}{\partial f_{k,i}} \nabla_w f_{k,i}$

$$\nabla_{f_{ab}} C(f) = -\frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K \mathbb{1}(y_i=k) \underbrace{\nabla_{f_{ab}} \log \frac{\exp(f_{k,i})}{\sum_{j=1}^K \exp(f_{j,i})}}_{\text{--- --- --- ---}}$$

$$\underbrace{\nabla_{f_{ab}} \log \frac{\exp(f_{k,i})}{\sum_{j=1}^K \exp(f_{j,i})}}_{\text{--- --- --- ---}} = \frac{\partial}{\partial (\exp(f_{k,i}))} \log \frac{\exp(f_{k,i})}{\sum_{j=1}^K \exp(f_{j,i})} \frac{\partial \exp(f_{k,i})}{\partial f_{ab}}$$

$$= \frac{\sum_{j=1}^K \exp(f_{j,i})}{\exp(f_{k,i})} \left(\frac{1}{\sum_{j=1}^K \exp(f_{j,i})} - \frac{\exp(f_{k,i})}{(\sum_{j=1}^K \exp(f_{j,i}))^2} \right) \frac{\partial \exp(f_{k,i})}{\partial f_{ab}}$$

$$= \cancel{\frac{1}{\exp(f_{k,i})}} \left(1 - \frac{\exp(f_{k,i})}{\sum_{j=1}^K \exp(f_{j,i})} \right) \cancel{\exp(f_{k,i})} S_{ka} \delta_{ib}$$

$$\Rightarrow \nabla_{f_{ab}} C(f) = -\frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K \mathbb{1}(y_i=k) \left(1 - \frac{\exp(f_{k,i})}{\sum_{j=1}^K \exp(f_{j,i})} \right) \delta_{ka} \delta_{ib}$$

$$= \frac{1}{m} \left(\frac{\exp(f_{a,b})}{\sum_{j=1}^K \exp(f_{j,b})} - \mathbb{1}(y_b=a) \right)$$

c)

$$W_k x_i = (w_{k1} \ w_{k2} \ w_{k3} \dots) \begin{pmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ \vdots \end{pmatrix} = w_{k1} x_{i1} + w_{k2} x_{i2} + w_{k3} x_{i3} + \dots$$

$$\Rightarrow \nabla_w f_{k,i} = \begin{pmatrix} \frac{\partial}{\partial w_{11}} & \frac{\partial}{\partial w_{12}} & \dots \\ \frac{\partial}{\partial w_{21}} & \frac{\partial}{\partial w_{22}} & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} W_k x_i = \begin{pmatrix} 0 \\ \vdots \\ x_i^\top \\ \vdots \\ 0 \end{pmatrix} \xrightarrow{k\text{te Zeile}}$$

$$\Rightarrow (\nabla_w f_{k,i})_a = \delta_{ka} x_i^\top$$

$$\nabla_b f_{k,i} = \frac{\partial b_k}{\partial b} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \xrightarrow{k\text{te Zeile}}$$

$$\Rightarrow (\nabla_b f_{k,i})_a = \delta_{ka}$$

e) $f_1 > f_2$: Datenpunkt gehört zur ersten Klasse

$f_1 < f_2$: Datenpunkt gehört zur zweiten Klasse

$f_1 = f_2$: Grenzfall \Rightarrow Trenngerade

Herleitung:

$$f_1 = f_2$$

$$\Leftrightarrow w_1 \vec{x} + b_1 = w_2 \vec{x} + b_2$$

$$\Leftrightarrow (w_1 - w_2) \vec{x} + b_1 - b_2 = 0$$

$$\Leftrightarrow \begin{pmatrix} w_{11} - w_{21} & w_{12} - w_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + b_1 - b_2 = 0$$

$$\Leftrightarrow x(w_{11} - w_{21}) + y(w_{12} - w_{22}) + b_1 - b_2 = 0$$

$$\Leftrightarrow y = \frac{b_2 - b_1 - x(w_{11} - w_{21})}{w_{12} - w_{22}}$$

Achtung im Code: $w_{i,j}$ in der Vorlesung genau transponiert definiert:

$$w_{12} \leftrightarrow w_{21}, \quad b_1 \leftrightarrow b_2$$

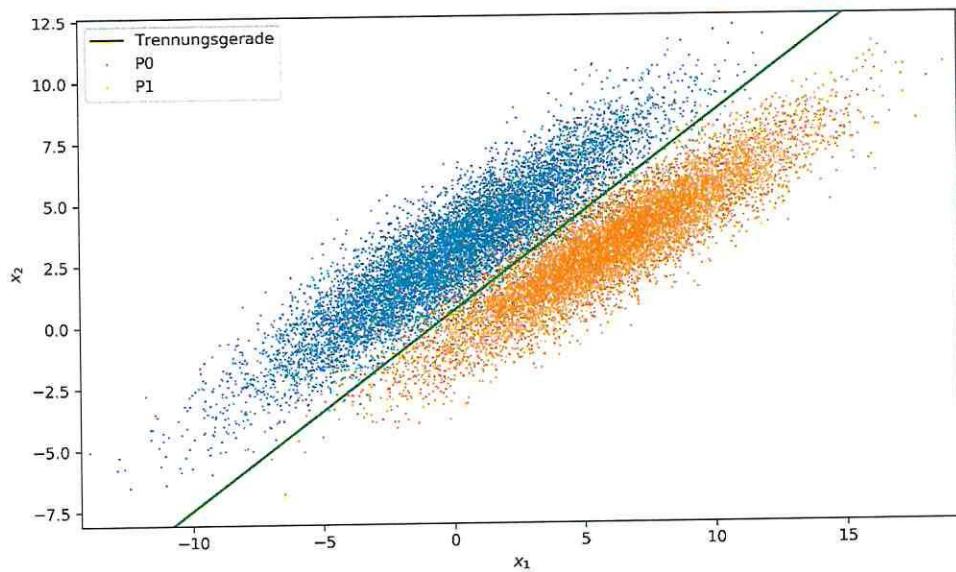


Abbildung 1: Datenpunkte und Trenngrade.

9/10

$$73) \quad y = a_0 + a_1 x$$

$$a_0 = 1,0 \pm 0,2$$

$$\rho = -0,8 = \frac{\text{cov}(a_0, a_1)}{\sigma_{a_0} \sigma_{a_1}}$$

$$a_1 = 1,0 \pm 0,2$$

$$\sigma_{a_0} = \sigma_{a_1} = 0,2$$

$$\text{allg.: } \sigma_y = \sqrt{\sum_{i=1}^m \left(\frac{\partial y}{\partial x_i} \sigma_{x_i} \right)^2 + 2 \sum_{i=1}^{m-1} \sum_{k=i+1}^m \left(\frac{\partial y}{\partial x_i} \right) \left(\frac{\partial y}{\partial x_k} \right) \text{cov}(x_i, x_k)}$$

$$\begin{aligned} \text{a) } \sigma_y &= \sqrt{\left(\frac{\partial y}{\partial a_0} \sigma_{a_0} \right)^2 + \left(\frac{\partial y}{\partial a_1} \sigma_{a_1} \right)^2 + 2 \left(\frac{\partial y}{\partial a_0} \right) \left(\frac{\partial y}{\partial a_1} \right) \text{cov}(a_0, a_1)} \\ &= \sqrt{0,2^2 + 0,2^2 x^2 + 2 \cdot 0,2^2 \cdot (-0,8)} \\ &= 0,2 \sqrt{1+x^2 - 1,6x} \end{aligned}$$

mit $\rho = 0$:

$$\sigma_y = 0,2 \sqrt{1+x^2}$$

c) analytisch

$$y(-3) = -2,0 \pm 0,8$$

$$y(0) = 1,0 \pm 0,2$$

$$y(3) = 4,0 \pm 0,5$$

numerisch

$$y(-3) = -2,0 \pm 0,8$$

$$y(0) = 1,0 \pm 0,2$$

$$y(3) = 4,0 \pm 0,5$$

Funktioniert gut!

Wappt hält besser, wenn man mehr Werte für a_0 und a_1 zieht.

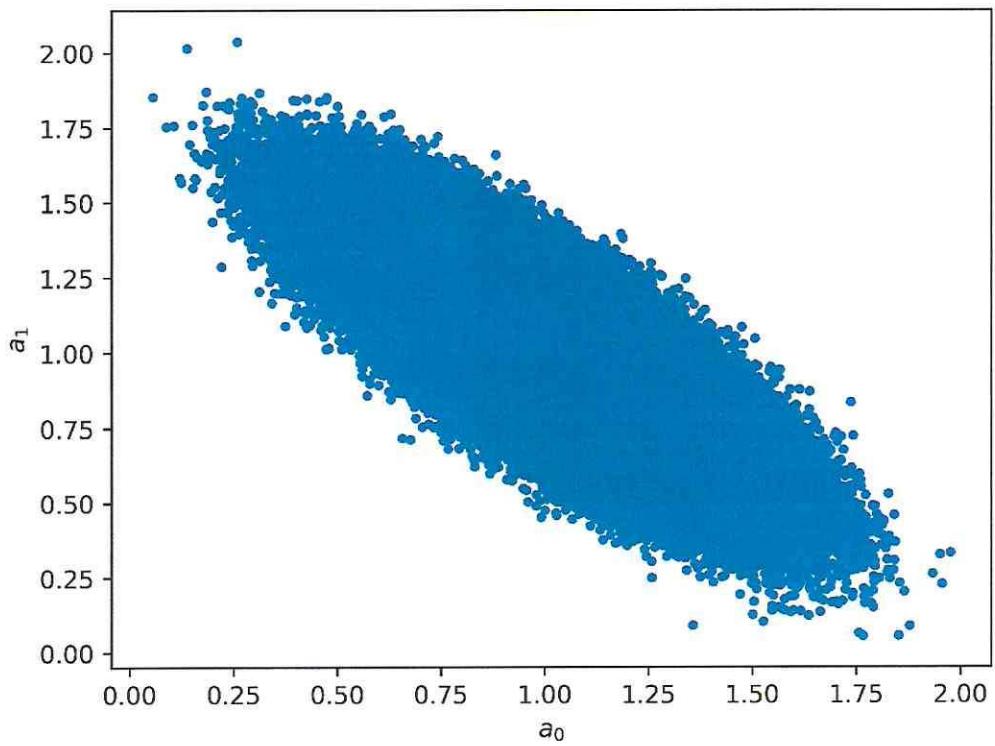


Abbildung 2: Korrelierte Parameter a_0 und a_1 .

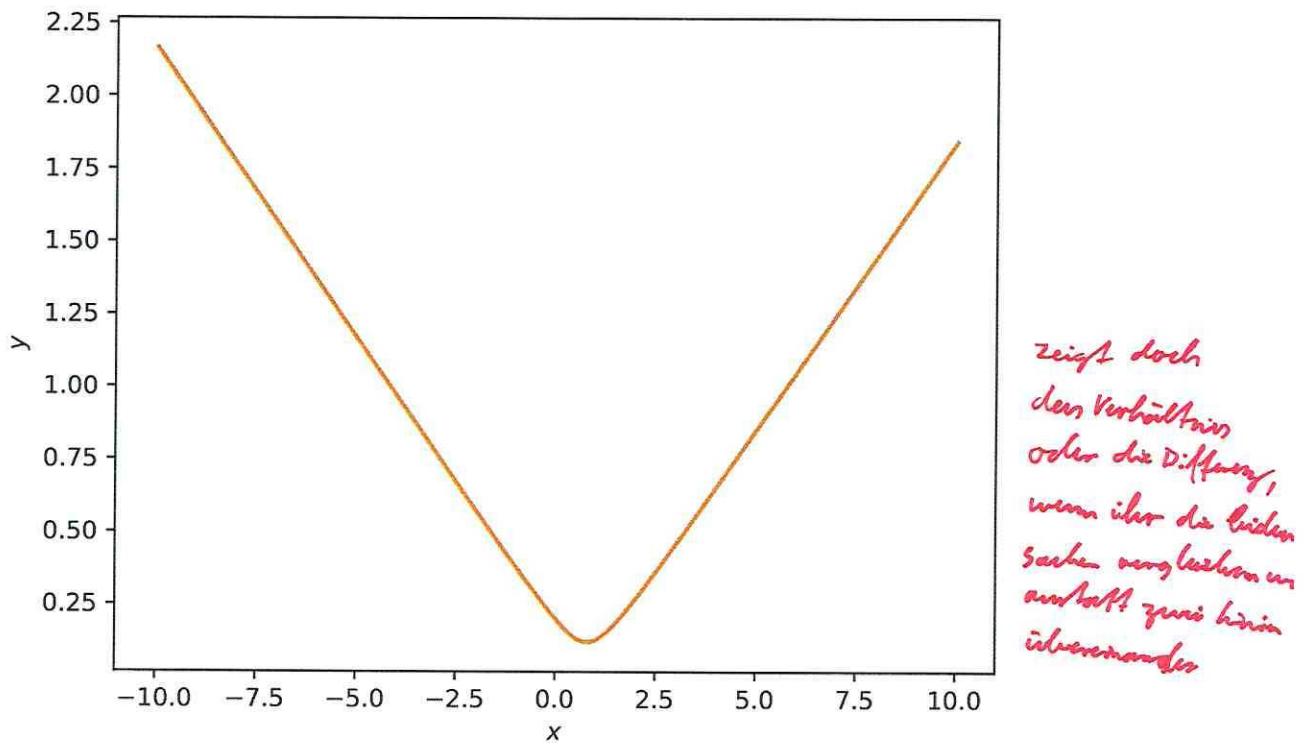


Abbildung 3: Vergleich des numerisch bestimmten Fehlers mit dem analytisch bestimmten Fehlers.

Code für blatt07

Bange, Burkowitz, Harnisch

18. Dezember 2017

```
..../blatt07/Bange_Burkowitz_Harnisch/aufg22.py
1 from sklearn.datasets import make_blobs
2 from sklearn.preprocessing import label_binarize
3 from matplotlib.colors import LinearSegmentedColormap
4 import matplotlib.pyplot as plt
5 import numpy as np
6 from tqdm import tqdm
7 from pylab import rcParams
8
9 rcParams['figure.figsize'] = 10, 5.8
10 rcParams['legend.numpoints'] = 1
11
12
13 # _____ Functions _____
14 # Aus der Vorlesung kopiert, aber besser kommentiert
15 #
16
17 def softmax(f):
18     """
19         Verschobene Softmax für bessere numerische Stabilität
20     """
21     f_shifted = f - f.max()
22     p = np.exp(f_shifted).T/np.sum(np.exp(f_shifted), axis=1)
23     return p.T
24
25
26 def loss_cross_ent(X, y, W):
27     """
28         Berechnet die Cross-Entropy des gesamten Datensatzes mit dem gegebenen W
29     """
30     # Falls die erste Spalte von X Einsen sind, ist die erste Zeile von W
31     # der Biasvektor
32     f = X@W
33     q = softmax(f)
34     # Mittelwert über alle Datenpunkte, wie nach (1), definitiv ein Skalar...
35     return -np.sum(y*np.log2(q), axis=1).mean()
36
37
38 def gradient(W, X, y):
39     """
40         Berechnet den Gradienten der Cross-Entropy für das aktuelle W
41     """
42     # Falls die erste Spalte von X Einsen sind, ist die erste Zeile von W
43     # der Biasvektor
44     # würde reichen f und softmax nur einmal zu berechnen für loss_cross_end
45     # und gradient, ist aber hier von der Performance kein Problem und so
46     # besser lesbar
47     f = X@W
48     p = softmax(f)
49     # Herleitung war aufgabe b). Hier sieht man auch ganz klar, dass es
50     # definitiv ein Vektor ist was in (4) falsch ist...
51     dh = (p - y) # Eigentlich noch /m aber das kommt in der nächsten Zeile
52     dW = X.T@dh/dh.shape[0]
53     return dW
54
55
56 def gradient_descent(X, y, max_iter=10000, step_size=0.01):
57     """
```

```

58 Führt den gradient descent auf der Cross-Entropy durch.
59 Es macht absolut keinen Sinn hier, wie in der VL, die Lossfunktion
60 als Parameter zu übergeben, weil die Gradienten Funktion bereits analytisch
61 aus dieser hergeleitet ist. Wenn dann muss man die auch übergeben
62 oder alles numerisch ableiten.
63 """
64 K = y.shape[1]
65 p = X.shape[1]
66
67 # leere liste um loss historie zu speichern
68 losses = []
69 # mit zufällig initialisierten Gewichten (und Biasen) anfangen
70 W = np.random.normal(size=(p, K))*step_size
71 for i in tqdm(range(max_iter)):
72     W = W - gradient(W, X, y)*step_size
73     loss = loss_cross_ent(X, y, W)
74     losses.append(loss)
75
76 return losses, W
77
78 if __name__ == "__main__":
    np.random.seed(723)
79     p0 = np.load("p0.npy").T
80     p1 = np.load("p1.npy").T
81
82     X = np.concatenate((p0, p1))
83     y = np.concatenate((np.ones(len(p0)), np.zeros(len(p1)))).T
84
85 """
86     X, y = make_blobs(n_samples=1500, n_features=2, center_box=(-5, 5),
87                         centers=2, cluster_std=0.3, random_state=1)
88 """
89
90     # Einserspalte an stelle 0 einfügen, sodass die erste Zeile
91     # der W Matrix zum Biasvektor wird
92     ones = np.ones(shape=(len(X), 1))
93     X_b = np.hstack([ones, X])
94     y_b = label_binarize(y, range(0, 3))[:, :-1]
95
96     losses, W = gradient_descent(X_b, y_b, max_iter=150, step_size=0.5)
97
98 """
99     prediction = np.argmax(softmax(X_b@W), axis=1)
100     discrete_cmap = LinearSegmentedColormap.from_list('discrete',
101                                                       colors=[(0.8, 0.2, 0.3),
102                                                               (0, 0.4, 0.8)],
103                                                       N=2)
104     plt.scatter(X[:, 0], X[:, 1], c=discrete_cmap(y), marker=".", s=0.5,
105                 alpha=1)
106     plt.scatter(X[:, 0], X[:, 1], facecolor='', marker=".", s=5, alpha=0.25,
107                 edgecolors=discrete_cmap(prediction))
108 """
109
110     plt.scatter(p0[:, 0], p0[:, 1], marker=".", s=0.5, alpha=1, label="p0")
111     plt.scatter(p1[:, 0], p1[:, 1], marker=".", s=0.5, alpha=1, label="p1")
112
113     x = np.array([min(X[:, 0]), max(X[:, 0])])
114     plt.plot(x, (x*(W[1][0] - W[1][1]) + W[0][0] - W[0][1])/(W[2][1] -
115                                                               W[2][0]), "g-",
116                 label="Trennungsgerade")
117
118     plt.legend()
119     plt.xlim(x)
120     plt.ylim((min(X[:, 1]), max(X[:, 1])))
121     plt.xlabel("$x_1$")
122     plt.ylabel("$x_2$")
123     # plt.show()
124     plt.savefig("A22.png", dpi=300)

```

..blatt07/Bange_Burkowitz_Harnisch/aufg23.py

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from uncertainties import ufloat
4 from tqdm import tqdm
5
6
7 # _____ Funktionen _____
8 def line(x, n):
9     a0 = np.random.normal(size=n)
10    a1 = np.random.normal(size=n)
11    # einfacher damit es lesbar ist alles hier ausrechnen
12    a0 = np.sqrt(1 - 0.8**2)*0.2*a0 - 0.8*0.2*a1 + 1
13    a1 = 0.2*a1 + 1
14
15    return a0 + a1*x
16
17
18 # _____ Main _____
19 if __name__ == "__main__":
20     np.random.seed(1234)
21     x = np.linspace(-10, 10, 1000)
22
23     # wir nehmen jetzt einfach für alle x die selben a0 und a1
24     a0 = np.random.normal(size=1000000)
25     a1 = np.random.normal(size=1000000)
26     a0 = np.sqrt(1 - 0.8**2)*0.2*a0 - 0.8*0.2*a1 + 1
27     a1 = 0.2*a1 + 1
28
29     y_err = np.empty(len(x))
30     y_m = np.empty(len(x))
31     for idx, xi in tqdm(enumerate(x), total=len(x)):
32         y = a0 + a1*xi
33         y_m[idx] = np.mean(y)
34         y_err[idx] = np.std(y)
35
36     # Plotten
37     plt.scatter(a0, a1, marker=".")
38     plt.xlabel("$a_0$")
39     plt.ylabel("$a_1$")
40     # plt.show()
41     plt.savefig("A23_scatter.png", dpi=600)
42     plt.clf()
43
44     plt.plot(x, y_err, label="Numerisch")
45     plt.plot(x, 0.2*np.sqrt(1 + x**2 - 1.6*x), label="Analytisch")
46     plt.xlabel("$x$")
47     plt.ylabel("$y$")
48     # plt.show()
49     plt.savefig("A23_resultat.pdf")
50
51     print("y(-3) =", ufloat(y_m[350], y_err[350]))
52     print("y(0) =", ufloat(y_m[500], y_err[500]))
53     print("y(3) =", ufloat(y_m[650], y_err[650]))

```

Zeit	Raum	Abgabe im Moodle; Mails mit Betreff: [SMD1718]
Di. 10-12	CP-03-150	philipp2.hoffmann@udo.edu und jan.soeddingrekso@udo.edu
Di. 16-18	P1-02-110	felix.neubuerger@udo.edu und tobias.hoinka@udo.edu
Di. 16-18	CP-03-150	simone.mender@udo.edu und maximilian.meier@udo.edu

Aufgabe 25: γ -Astronomie

7 P.

Bei einer typischen Messung in der γ -Astronomie wird das Teleskop auf eine Position (*on*-Position) gerichtet, an der eine γ -Strahlungs-Quelle vermutet wird. In der anschließenden Messung werden N_{on} Ereignisse über einen Zeitraum t_{on} aufgezeichnet. In den gemessenen Ereignissen N_{on} befinden sich sowohl Untergrund- als auch Signalphotonen. Um zu ermitteln, wie viel Untergrund vorhanden ist, wird ebenfalls an einer anderen Position ohne Quelle (*off*-Position) gemessen. Bei dieser Messung werden N_{off} Photonen-Ereignisse in einer Zeit t_{off} gemessen.

Um zu entscheiden, ob sich an der *on*-Position eine Quelle befindet, soll mit einem Likelihood-Quotienten-Test getestet werden, ob ein signifikanter Überschuss an Photonen über der Untergrundunterwartung für die *on*-Position gemessen wurde (hier noch nicht, erst in Kapitel *Testen*).

Ziel dieser Aufgabe ist es, vorbereitend die richtige Likelihood-Funktion für den späteren Likelihood-Quotienten-Test aufzustellen.

Nutzen Sie für die Bearbeitung der Aufgabe die Ausdrücke:

- $\alpha = \frac{t_{\text{on}}}{t_{\text{off}}}$: Quotient der unterschiedlichen Messzeiten
- $b = \langle N_{\text{off}} \rangle$: Unbekannter Erwartungswert für die Zahl der Untergrundphotonen während der Messzeit t_{off}
- s : Unbekannter Erwartungswert für die Zahl der Signal Photonen während der Messzeit t_{on} aus der γ -Strahlungs-Quelle. Nicht zu verwechseln mit der gesamten Erwartung für die *on*-Position.

a) Wie groß ist der Erwartungswert $\langle N_{\text{on}} \rangle$, ausgedrückt durch s , b und α ?

b) Welchen Wahrscheinlichkeitsverteilungen folgen N_{on} und N_{off} ?

Tipp: Die gezählten Ereignisse kommen unabhängig voneinander im Detektor an.

c) Wie sieht die Likelihoodfunktion $\mathcal{L}(b, s)$ für die Parameter b und s aus?

- d) Welche Werte \hat{b} und \hat{s} maximieren \mathcal{L} ?
Tipp: Nutzen Sie die negative Likelihood-Funktion, dann wird die Rechnung einfacher.
- e) Berechnen Sie die Kovarianzmatrix von \hat{b} und \hat{s} . Wie hängt die Kovarianzmatrix mit der Likelihood zusammen? Ist diese Art der Fehlerberechnung exakt?

Aufgabe 26: Stichprobenvarianz

7 P.

Für alle Berechnungen sind x_1, \dots, x_n die Ausprägungen der quadratisch integrierbaren, paarweise unkorrelierten, reellwertige Zufallsvariablen X_1, \dots, X_n mit der Varianz σ^2 und dem Mittelwert μ .

- a) Testen Sie, ob die Formel (arithmetisches Mittel)

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i \quad (1)$$

eine erwartungstreue Schätzfunktion für den Mittelwert μ der Grundgesamtheit ist. Falls die Schätzfunktion nicht erwartungstreu ist, suchen Sie nach einer geeigneten Korrektur.

- b) Der Standardfehler des arithmetischen Mittels (1) ist definiert als die Wurzel aus der Varianz von \bar{X} . Zeigen Sie, dass

$$E((\bar{X} - \mu)^2) = \text{Var}(\bar{X}) = \frac{\sigma^2}{n} \quad (2)$$

gilt.

Tipp: Schauen Sie sich Rechenregeln für das Rechnen mit Varianzen an.

- c) Testen Sie, ob die Formel

$$S_0^2 = \frac{1}{n} \sum_{i=1}^n (X_i - \mu)^2 \quad (3)$$

eine erwartungstreue Schätzfunktion für die Varianz σ^2 der Grundgesamtheit ist. Falls die Schätzfunktion nicht erwartungstreu ist, suchen Sie nach einer geeigneten Korrektur.

- d) Meist ist die Varianz σ^2 der Grundgesamtheit unbekannt und es wird der Schätzer (1) für μ genutzt und (3) wird zu:

$$S_1'^2 = \frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2. \quad (4)$$

Testen Sie, ob (4) eine erwartungstreue Schätzfunktion für die Varianz σ^2 der Grundgesamtheit ist. Falls die Schätzfunktion nicht erwartungstreu ist, suchen Sie nach einer geeigneten Korrektur.

Tipp: Erweitern Sie den Summanden mit $-\mu + \mu$ und nutzen Sie die gebene Relation (2).

Aufgabe 27: Maximum-Likelihood

6 P.

Eine Zufallsvariable x soll einer Gleichverteilung

$$f(x) = \begin{cases} 1/b & 0 \leq x \leq b \\ 0 & x < 0 \quad \text{oder} \quad x > b \end{cases}$$

folgen.

- a) Bestimmen Sie einen Schätzer für den Parameter b mit der Maximum Likelihood Methode aus einer Stichprobe x_1, x_2, \dots, x_n .
- b) Ist diese Schätzung erwartungstreu? Wenn nein, wie kann das in diesem Fall korrigiert werden?

A 25:

a) $\langle N_{on} \rangle = s + \alpha b$

b) Poisson ($p = \frac{\lambda^k e^{-\lambda}}{k!}$)

$$p_{on} = \frac{\langle N_{on} \rangle^{N_{on}} e^{-\langle N_{on} \rangle}}{N_{on}!} = \frac{(s + \alpha b)^{N_{on}} e^{-(s + \alpha b)}}{N_{on}!}$$

$$p_{off} = \frac{\langle N_{off} \rangle^{N_{off}} e^{-\langle N_{off} \rangle}}{N_{off}!} = \frac{b^{N_{off}} e^{-b}}{N_{off}!}$$

c) $L(b, s) = p_{on} p_{off}$

$$= \frac{(s + \alpha b)^{N_{on}} e^{-(s + \alpha b)} b^{N_{off}} e^{-b}}{N_{on}! N_{off}!}$$

$$= \frac{(s + \alpha b)^{N_{on}} b^{N_{off}} e^{-(s + b(\alpha + 1))}}{N_{on}! N_{off}!}$$

d) Nenner egal zum Maximieren

$$\Rightarrow \text{Maximiere } Z = N_{on}! N_{off}! L$$

Z und $\ln(Z)$ haben das Maximum an der gleichen Stelle

$$\Rightarrow \text{Maximiere } \ln(Z)$$

$$\ln(Z) = N_{on} \ln(s + \alpha b) + N_{off} \ln(b) - s - b(\alpha + 1)$$

$$\frac{\partial \ln(Z)}{\partial s} = \frac{N_{on}}{s + \alpha b} - 1 \stackrel{!}{=} 0$$

$$\Rightarrow N_{on} = \tilde{s} + \alpha b \quad (1)$$

$$\frac{\partial \ln(Z)}{\partial b} = \frac{N_{on}}{s + \alpha b} \alpha + \frac{N_{off}}{b} - \alpha - 1 \stackrel{!}{=} 0 \quad (2)$$

(1) in (2)

$$\Rightarrow \frac{\tilde{s} + \alpha b}{\tilde{s} + \alpha b} \alpha + \frac{N_{off}}{b} - \alpha - 1 = 0$$

$$\Leftrightarrow \frac{N_{off}}{b} = 1 \quad \Rightarrow \boxed{b = N_{off}}$$

\tilde{b} in (1)

$$\Rightarrow \boxed{\tilde{s} = N_{on} - \alpha N_{off}}$$

$$e) \sigma(\alpha) = \left(\frac{d^2 F}{d \alpha^2} \Big|_{\alpha} \right)^{-1/2}$$

$$F = -\ln(\zeta) = -N_{on} \ln(s+\alpha b) - N_{off} \ln(b) + s + b(\alpha+1) + \ln(N_{on}! N_{off}!)$$

$$\frac{\partial F}{\partial s} = -\frac{N_{on}}{s+\alpha b} + 1$$

$$\frac{\partial F}{\partial b} = -\frac{N_{on}}{s+\alpha b} \alpha - \frac{N_{off}}{b} + \alpha + 1$$

$$\frac{\partial^2 F}{\partial s^2} = \frac{N_{on}}{(s+\alpha b)^2}$$

$$\frac{\partial^2 F}{\partial s^2} \Big|_{s=\tilde{s}, b=\tilde{b}} = \frac{1}{N_{on}}$$

$$\frac{\partial^2 F}{\partial s \partial b} = \frac{N_{on}}{(s+\alpha b)^2} \alpha = \frac{\partial^2 F}{\partial b \partial s} \Big|_{s=\tilde{s}, b=\tilde{b}} = \frac{1}{N_{on}} \alpha$$

$$\frac{\partial^2 F}{\partial b^2} = \frac{N_{on}}{(s+\alpha b)^2} \alpha^2 + \frac{N_{off}}{b^2}$$

$$\frac{\partial^2 F}{\partial b^2} \Big|_{s=\tilde{s}, b=\tilde{b}} = \frac{1}{N_{on}} \alpha^2 + \frac{1}{N_{off}}$$

$$\Rightarrow \sigma^2 = \left(\frac{1}{N_{on}} \begin{pmatrix} 1 & \alpha \\ \alpha & \alpha^2 + \frac{N_{on}}{N_{off}} \end{pmatrix} \right)^{-1} = \frac{N_{on}}{\alpha^2 + \frac{N_{on}}{N_{off}} - \alpha^2} \begin{pmatrix} \alpha^2 + \frac{N_{on}}{N_{off}} & -\alpha \\ -\alpha & 1 \end{pmatrix}$$

$$= N_{off} \begin{pmatrix} \alpha^2 + \frac{N_{on}}{N_{off}} & -\alpha \\ -\alpha & 1 \end{pmatrix}$$

✓

7/7

A 26

$$a) E(\bar{\mu}) = E(\bar{x}) = E\left(\frac{1}{n} \sum_{i=1}^n x_i\right) = \frac{1}{n} \sum_{i=1}^n E(x_i)$$
$$= \frac{1}{n} \sum_{i=1}^n \mu = \frac{1}{n} n \mu = \mu$$

\Rightarrow ist erwartungstreu

$$b) \text{Var}(\bar{x}) = \text{Var}\left(\frac{1}{n} \sum_{i=1}^n x_i\right) = \frac{1}{n^2} \text{Var}\left(\sum_{i=1}^n x_i\right)$$
$$= \frac{1}{n^2} \left[\text{Var}(x_1) + \text{Var}\left(\sum_{i=2}^n x_i\right) + 2 \underbrace{\text{Cov}(x_1, \sum_{i=2}^n x_i)}_{=0, \text{ unkorreliert}} \right]$$
$$= \dots = \frac{1}{n^2} \sum_{i=1}^n \text{Var}(x_i) = \frac{1}{n^2} \sum_{i=1}^n \sigma^2 = \frac{1}{n^2} n \sigma^2$$
$$= \frac{\sigma^2}{n}$$

$$c) E(\hat{\sigma}^2) = E\left(\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2\right) = \frac{1}{n} \sum_{i=1}^n E((x_i - \mu)^2)$$
$$= \frac{1}{n} \sum_{i=1}^n \text{Var}(x_i) = \frac{1}{n} n \sigma^2 = \sigma^2$$

$\Rightarrow s^2$ ist für die Varianz erwartungstreu

$$d) E(\hat{\sigma}'^2) = E\left(\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2\right) = \frac{1}{n} \sum_{i=1}^n E((x_i - \bar{x})^2)$$
$$= \frac{1}{n} \sum_{i=1}^n E((x_i - \mu) + (\mu - \bar{x}))^2$$
$$= \frac{1}{n} \sum_{i=1}^n E((x_i - \mu)^2 + 2(x_i - \mu)(\mu - \bar{x}) + (\mu - \bar{x})^2)$$
$$= \frac{1}{n} \left(\sum_{i=1}^n E(x_i - \mu)^2 + E\left(\sum_{i=1}^n 2(x_i - \mu)(\mu - \bar{x}) + (\mu - \bar{x})^2 \right) \right)$$
$$= \frac{1}{n} \left(\sum_{i=1}^n E(x_i - \mu)^2 + E(2n(\bar{x} - \mu)(\mu - \bar{x}) + n(\mu - \bar{x})^2) \right)$$
$$= \frac{1}{n} \left(\sum_{i=1}^n E(x_i - \mu)^2 + E(-n(\mu - \bar{x})^2) \right)$$
$$= \frac{1}{n} \left(n \sigma^2 - n \frac{\sigma^2}{n} \right)$$
$$= \sigma^2 - \frac{\sigma^2}{n} = \frac{n-1}{n} \sigma^2$$

\Rightarrow nicht erwartungstreu

$$\Rightarrow \text{Korrektur: } \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \rightarrow \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

✓ 7/7

A27:

a) $L(b) = \prod_{i=1}^n f(x_i | b) = \prod_{i=1}^n \frac{1}{b} \Theta(b - x_i)$ (untere Grenze ist fest auf $M_n(1)$)

$$= \frac{1}{b^n} \Theta(b - \max(X))$$

I $\Theta(b - \max(X)) \Rightarrow L \text{ maximal für } b \geq \max(X)$

II $\frac{1}{b^n} \Rightarrow \text{stetig fallend} \rightarrow b \text{ muss minimal sein}$

I & II \Rightarrow Schätzer: $\hat{b} = \max(X)$

b) $E(\hat{b}) = E(\max(X))$

Wahrscheinlichkeitsdichte:

$$g(x) = n \left(\int_0^x \frac{1}{b} dx \right)^{n-1} \frac{1}{b} \rightarrow \text{PDF, um genau den Wert } x \text{ zu ziehen.}$$

WSK, $n-1$ mal einen Wert kleiner als x zu ziehen

Das Maximum kann an n verschiedenen Stellen stehen.

$$g(x) = n \left(\frac{x}{b} \right)^{n-1} \frac{1}{b}$$

$$\Rightarrow E(\max(X)) = \int_0^b x g(x) dx = \int_0^b \frac{n}{b^n} x^n dx = \frac{n}{b^n} \frac{b^{n+1}}{n+1}$$

$$= \frac{n}{n+1} b$$

$$\Rightarrow E(\hat{b}) = \frac{n}{n+1} b \neq b$$

\Rightarrow nicht erwartungstreu

$$\Rightarrow \text{Korrektur: } \hat{b}' = \frac{n+1}{n} \max(X)$$

$$E(\hat{b}') = \frac{n+1}{n} E(\max(X)) = \frac{n+1}{n} \cdot \frac{n}{n+1} b = b$$

✓

6/6

a)

$$\langle N_{on} \rangle = s + jb$$

b) Poisson:

$$P_{on}(N_{on}) = \frac{\langle N_{on} \rangle^{N_{on}} e^{-\langle N_{on} \rangle}}{N_{on}!} = \frac{(s+jb)^{N_{on}} e^{-(s+jb)}}{N_{on}!}$$

$$P_{off}(N_{off}) = \frac{b^{N_{off}} e^{-b}}{N_{off}!}$$

$$c) \mathcal{L}(b, s) = P_{on}(N_{on}) P_{off}(N_{off})$$

$$= \frac{(s+jb)^{N_{on}} e^{-(s+jb)} b^{N_{off}} e^{-b}}{N_{on}! N_{off}!}$$

$$= \frac{(s+jb)^{N_{on}} b^{N_{off}} \exp(-(s+b(\lambda+\gamma)))}{N_{on}! N_{off}!}$$

d) Meiner egal für mich

$$\text{Grad}_{(b,s)}(\tilde{\mathcal{L}}) \stackrel{!}{=} \vec{0}$$

$$\tilde{\mathcal{L}} := N_{on}! N_{off}! f$$

$$\frac{\partial}{\partial s} \tilde{\mathcal{L}} = -b^{N_{off}} \exp(-(s+b(\lambda+\gamma))) (2b+s)^{N_{on}-1} (jb+s-N_{on})$$

$$\frac{\partial}{\partial b} \tilde{\mathcal{L}} =$$

Viel zu leicht / (iob)

($\partial_b(\tilde{\mathcal{L}})$) nehmen!

$$\log(\tilde{L}) = N_{on} \log(s+b) + N_{off} \log(s) \\ \oplus - (s+b(s+1))$$

$$\frac{\partial}{\partial s} \log(\tilde{L}) = \frac{N_{on}}{s+b} - 1 \stackrel{!}{=} 0$$

$$\Leftrightarrow \tilde{s} + \tilde{b} = N_{on} \quad (1)$$

$$\frac{\partial}{\partial b} \log(\tilde{L}) = \frac{N_{on}}{s+b} + \frac{N_{off}}{b} - 1 - 1 \stackrel{!}{=} 0$$

$$\left(\Leftrightarrow \frac{s N_{on} \tilde{b} + N_{off} (\tilde{s} + \tilde{b})}{b (\tilde{s} + \tilde{b})} = 1 \right)$$

(1) einsetzen:

$$1 \cdot \tilde{b} + \frac{N_{off}}{\tilde{b}} - 1 - 1 = 0$$

$$\Rightarrow \tilde{b} = N_{off} \quad \text{mache ausdrücklich} \\ \stackrel{(1)}{\Rightarrow} \tilde{s} = N_{on} - \tilde{b} N_{off} \quad \text{ein}$$

Eigentlich müsste man noch überprüfen, ob wir NC (Maxim.)

e)

Zeit	Raum	Abgabe im Moodle; Mails mit Betreff: [SMD1718]
Di. 10-12	CP-03-150	philipp2.hoffmann@udo.edu und jan.soedingrekso@udo.edu
Di. 16-18	P1-02-110	felix.neubuerger@udo.edu und tobias.hoinka@udo.edu
Di. 16-18	CP-03-150	simone.mender@udo.edu und maximilian.meier@udo.edu

Aufgabe 28: *Data-Mining Challenge*

20 P.

In dieser Aufgabe soll eine reale Problemstellung aus dem Gebiet des Immobiliengeschäfts behandelt werden. Es handelt sich dabei um die Schätzung von Haus-Verkaufspreisen. Es steht Ihnen wie immer frei, wie viel Zeit und Anstrengung Sie für diese Aufgabe verwenden, es wird jedoch **Amazon-Gutscheine** für die besten Regressionen zu gewinnen geben.

Problemstellung

Die Problemstellung ist ausführlich hier beschrieben:

<https://www.kaggle.com/c/house-prices-advanced-regression-techniques>
Kaggle ist eine Plattform, auf der Datensätze und Problemstellungen hochgeladen werden können und Benutzer versuchen die besten Modelle für die Probleme zu finden.

Datensatz

Die Daten finden Sie im Moodle, sie sind bereits vorverarbeitet. Wie der Problemstellung zu entnehmen ist, kann es sinnvoll sein, eigene Attribute zu erzeugen und ggfs. zusätzliche Informationen in den Datensatz mit einzubringen.

Auswertung

Die Qualität der Regression wird über das *Kaggle* Leaderboard bestimmt. Dafür muss eine CSV-Datei mit der ID der Häuser und dem erwarteten Verkaufspreis erstellt werden. Die Datei `sample_submission.csv` zeigt die korrekte Formatierung. Sie reichen die CSV-Datei und alle Skripte/Prozesse wie gewohnt über das Moodle ein. Wir werden dann Ihre finalen Regressionen auf *Kaggle* hochladen. Die drei Regressionen mit den höchsten Rängen im Leaderboard gewinnen Preise. Beachten Sie, dass für die Aufgabe ebenfalls Punkte für den Übungsbetrieb verteilt werden, die Gutscheine sind nur ein Bonus.

Hinweise

Benutzen Sie Python für die Lösung. Die Bibliotheken `pandas` und `sklearn` können dabei hilfreich sein. Sollte `sklearn` benutzt werden, achten Sie darauf die Daten geeignet zu formatieren (`numpy.array, float32`).

Sie müssen sich keine *Kaggle* Account für diese Aufgabe erstellen. Alle erforderlichen Daten sind im Moodle bereitgestellt.

Zur Vorbereitung gibt es zusätzlich kleine Aufgaben, um sich mit dem Datensatz vertraut zu machen:

- a) Lesen Sie die Datei `train.csv` ein. Bestimmen Sie die drei Attribute mit der höchsten Korrelation zum Attribute `SalePrice`. Stellen Sie die Korrelationen jeweils in einem Scatter-Plot dar.
- b) Führen Sie eine lineare Regression zwischen dem Attribut mit der höchsten Korrelation und dem Attribut `SalePrice` durch. Stellen Sie die Regressionsgerade zusammen mit den Datenpunkten dar.
- c) Stellen Sie den relativen Abstand zwischen den geschätzten Verkaufspreisen aus der linearen Regression in Teil b) und den wahren Verkaufspreisen in einem Histogramm dar.
- d) Anschließend die Aufgabe der Data-Mining Challenge: Führen Sie eine Regression mit Methoden Ihrer Wahl im Rahmen einer Validierung auf den Trainingsdaten (`train.csv`) aus und wenden das Model auf den Testdatensatz (`test.csv`) an. Speichern Sie die regressierten Verkaufspreise zusammen mit den IDs der Häuser in einer csv-Datei gemäß der Beispieldatei `sample_submission.csv` von *Kaggle* ab.

Geben Sie jeglichen Code, der zur Lösung genutzt wurde mit ab. Zusätzlich muss eine korrekt formatierte csv-Datei mit den finalen Regressionsergebnissen abgegeben werden.

Aufgabe 28

a) b)

Die drei Attribute, die die höchste Korrelation zum Zielattribut, also dem Verkaufspreis, aufweisen sind *OverallQual*, *GrLivArea* und *GarageCars*. Also die Gesamt Qualität, die Fläche des bewohnbaren Bereiches im Erdgeschoss und die Anzahl der Autos, die insgesamt in die Garage bzw. die Garagen passen. Die Korrelationen sind als Scatter-Plots zusammen mit Regressionsgeraden in den Abbildungen 1 bis 3 dargestellt.

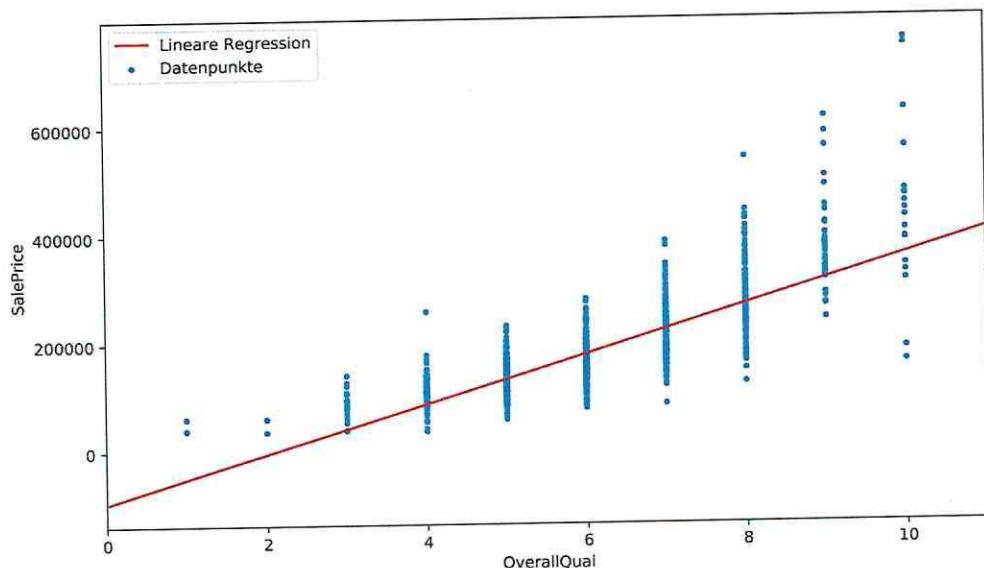


Abbildung 1: Korrelation zwischen *OverallQual* und *SalePrice*.

20/20

0.1 c)

Die relativen Abstände zwischen den geschätzten Verkaufspreisen aus den linearen Regressionsen und den wahren Verkaufspreisen sind in den Histogrammen 4 bis 6 dargestellt.

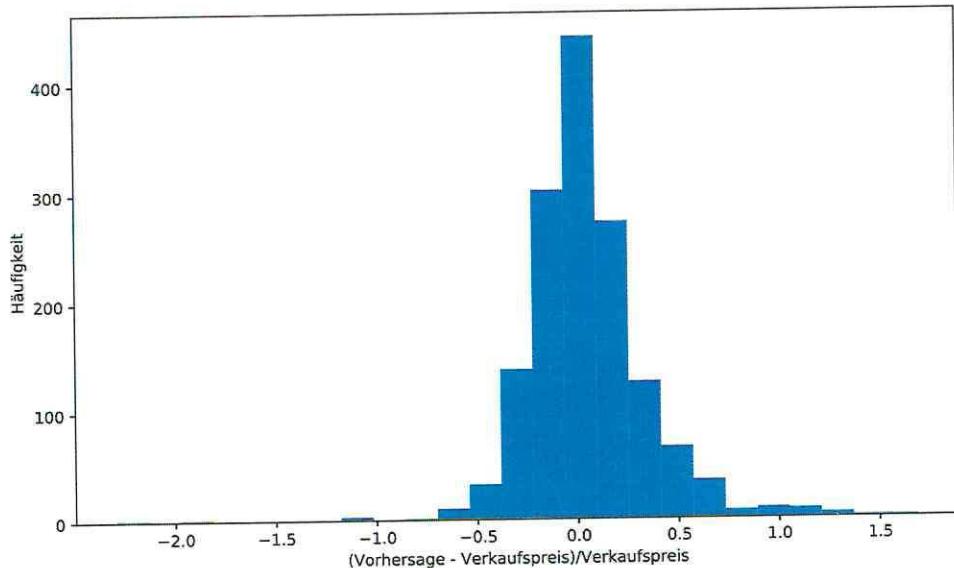


Abbildung 4: Histogramm der relativen Abstände zwischen dem vorhergesagten und dem tatsächlichen Verkaufspreis für das Attribut *OverallQual*.

Aufgabe 28 d) - Data-Mining Challenge

Insgesamt hatten wir zu wenig Zeit, um die Challenge zu unserer eigenen Zufriedenheit zu bearbeiten. Darum war die Idee erst einmal von Kernels auf Kaggle zu lernen und als Basis damit selbst zu versuchen weiterzumachen. Leider haben wir es nicht geschafft, den Score des Kernels [1] zu verbessern. Unser bester public Score von 0,11968 ist der mit XGBoost [2] ohne Änderungen aus [1], daher keine Eigenleistung. Wir waren allerdings wirklich interessiert daran, ob wir es schaffen würden mit neuronalen Netzen, die ja nach allgemeiner aktueller Meinung für tabellarische Probleme eher weniger geeignet sind, als geboostete Entscheidungsbäume, diesen Score zu übertreffen oder zumindest ähnlich gute Ergebnisse zu erzielen. Übertreffen konnten wir ihn leider nicht, allerdings sind wir zumindest nach unserer eigenen Crossvalidation auf den Trainingsdaten zu ähnlich guten Ergebnissen gekommen. Der beste public Score viel allerdings mit 0,13084 schlechter aus, als unser CV-Score. Mit einem neuronalen Netz ähnliche gute Ergebnisse zu erzielen war jedoch deutlich schwerer und zeitaufwändiger als anfänglich gedacht. Und auch wenn dies leider trotzdem ein Misserfolg war, wissen wir so nun zumindest, dass Entscheidungsbäume hier wirklich das bessere Verfahren zu sein scheinen. Wir haben es immerhin geschafft, den public Score von neuronalen Netzen durch Anpassen der Architektur und Featureselection von 0,20233 (minimalistisches neuronales Netz ohne Featureselection) auf 0,13092 (komplexere Architektur und Featureselection) zu verbessern. Unser bestes Ergebnis von 0,13084 nach dem Publicscore hat jedoch das selbe Netz ohne Featureselection erzielt, was unserer Meinung nach aber einfach nur unglücklicher Zufall sein sollte, da unser Netz mit Featureselection in CV auf den Trainingsdaten etwas besser abgeschnitten hat, als ohne Featureselection.

Wir haben es nicht geschafft durch eigenes Feature Engineering irgendetwas gegenüber [1] zu verbessern. Trotzdem haben wir natürlich nachvollzogen und verstanden, wobei es beim Feature Engineering auf diesen Daten ankommt. Daher beschreiben wir dies hier auch nochmal. Unsere Eigenleistung hier besteht darin eine relativ gute Netzarchitektur zu finden und für diese Architektur eine Featureselection durchzuführen, auch wenn man hier mit mehr Zeit sicherlich noch sehr viel mehr rausholen könnte.

Das endgültige Ergebnis sollte mit unserem Code replizierbar sein, die Optimierung des Modells und der Featureselection allerdings nicht, da viel von dem dafür verwendeten Code nicht mehr in der exakt auszuführenden Form übrig ist, da er relativ unsauber dauernd manuell angepasst wurde. Um hier streng sauber wissenschaftlich zu arbeiten fehlte die Zeit. Außerdem haben wir allein für die Featureselection insgesamt etwa 10 Stunden Rechenzeit auf einer GeForce GTX 1060 investiert.

Die Kommentare in der Abgabe sind auf Englisch, um konsistent zu sein und den Code eventuell online teilen zu können.

Vorbereiten der Daten und Feature Engineering

Das in der Abgabe stehende Feature Engineering stammt in dieser Form nicht von uns, sondern von [1]. Eigentlich sollte alles durch die Kommentare, die auch von von [1] stammen, erklärt sein. Aus den ursprünglich 81 Features werden insgesamt 407 Features generiert. Hier nochmal kurz das Wichtigste:

- Aussortieren von Ausreißern
- Fehlende Werte ersetzen, beispielsweise durch Median (LotFrontage)

sein könnte, gleichverteilt zufällige Thresholds aus den vorkommenden Scores gezogen. Das Ergebnis ist in Abbildung 7 dargestellt.

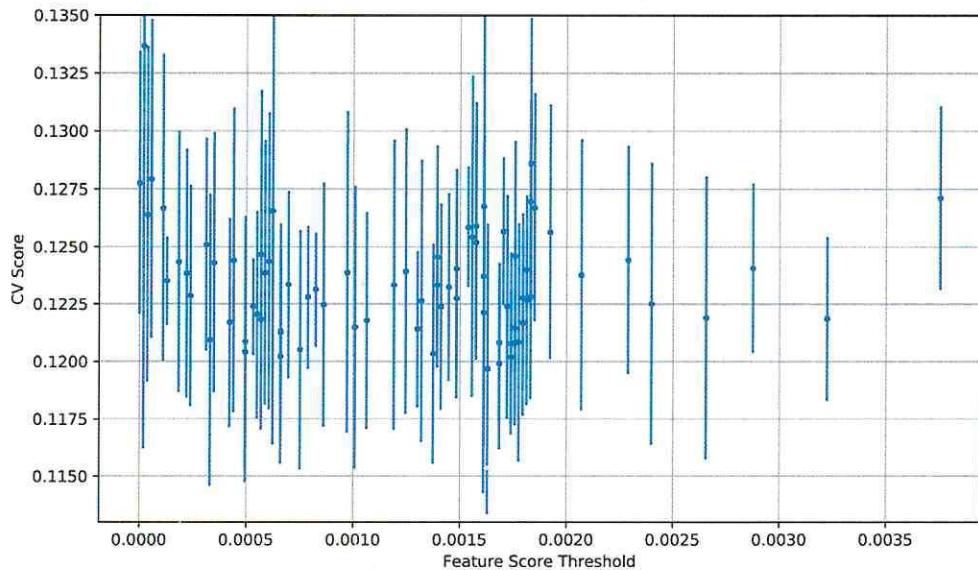


Abbildung 7: Crossvalidation Score für verschiedene Feature Score Thresholds.

Es ist ein Abwärtstrend für das Aussortieren der schlechtesten Features erkennbar. Insgesamt scheint die Featureselection allerdings keine sehr großen Verbesserungen zu erzielen, es lässt sich allerdings ein Minimum bei 0,0016 erahnen. Wir wählen als Threshold 0,00162974, mit diesem sind noch 156 der 407 Features übrig, was zumindest den Vorteil hat, dass das Training so sehr viel weniger aufwändig ist, was in der Praxis allerdings nur bedingt relevant ist.

Die gewählte Vorgehensweise ist kritisch zu sehen, da in der Featureselection bereits Information aus den gesamten Trainingsdaten enthalten ist, und nicht nur aus denen, die in den jeweiligen Splitts der Crossvalidation zum Training verwendet werden. Streng genommen müsste die Featureselection für jede Crossvalidation Split neu durchgeführt werden um ein wirklich absolut sauberes Ergebnis zu erhalten.

Ergebnisse

Wenn ihr wirklich nur nach dem Publicscore gehen wollt, dann beinhaltet *submission_xgb.csv* unser bestes Ergebnis von 0,11968, das allein beinhaltet aber praktisch keine Eigenleistung. Unser Ergebnis mit neuronalem Netz und Featureselection aus *submission_0.csv* ist mit 0,13092 deutlich schlechter und fällt sogar etwas schlechter aus als der Score von 0,13084 ohne Featureselection, was jedoch einfach Zufall sein sollte. Nach unserer Crossvalidation sind die Ergebnisse mit Featureselection etwas besser, als ohne Featureselection. Also letztlich haben wir auf Grund von Zeitmangel eigentlich nur einen Vergleich zwischen XGBoost und neuronalen Netzen durchgeführt und gezeigt, dass man durchaus neuronale Netze verwenden kann, XGBoost mit einem CV Score von $0,11 \pm 0,01$ auf den selben 10 Splitts jedoch in diesem Fall wirklich besser ist.

Literatur

- [1] Human Analog *XGBoost + Lasso - Kaggle Kernel* <https://www.kaggle.com/humananalog/xgboost-lasso/>
- [2] *XGBoost* <https://github.com/dmlc/xgboost/tree/master/python-package>
- [3] *Keras* <https://github.com/keras-team/keras>
- [4] Google *Tensorflow* <https://www.tensorflow.org>

Code für blatt09

Bange, Burkowitz, Harnisch

15. Januar 2018

```
..../blatt09/Bange_Burkowitz_Harnisch/aufg2Babc.py

1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from scipy.stats import linregress
5 from pylab import rcParams
6
7 rcParams['figure.figsize'] = 10, 5.8
8 rcParams['legend.numpoints'] = 1
9
10
11 df_train = pd.read_csv("train.csv")
12 df_test = pd.read_csv("test.csv")
13 cols = df_train.corr().nlargest(4, "SalePrice")["SalePrice"].index[1:]
14 print("Attribute mit der höchsten Korrelation in absteigender Reihenfolge:")
15 print(cols.values)
16
17 for feature in cols:
18     a, b, _, _, _ = linregress(df_train[feature], df_train["SalePrice"])
19     x = np.array([np.min(df_train[feature]) - 1, np.max(df_train[feature]) + 1])
20     plt.scatter(df_train[feature], df_train["SalePrice"], marker=".", 
21                 label="Datenpunkte")
22     plt.plot(x, a*x + b, "r-", label="Lineare Regression")
23     plt.xlabel(feature)
24     plt.xlim(x)
25     plt.ylabel("SalePrice")
26     plt.legend()
27     # plt.show()
28     plt.savefig("Scatter_{}.pdf".format((feature)))
29     plt.clf()
30
31     y_pred = a*df_train[feature] + b
32     plt.hist((y_pred - df_train["SalePrice"])/df_train["SalePrice"], bins=25)
33     plt.xlabel("(Vorhersage - Verkaufspreis)/Verkaufspreis")
34     plt.ylabel("Häufigkeit")
35     # plt.show()
36     plt.savefig("Hist_{}.pdf".format((feature)))
37     plt.clf()
```

```
..../blatt09/Bange_Burkowitz_Harnisch/challenge.py

1 import numpy as np
2 import pandas as pd
3 from scipy.stats import skew
4 from sklearn.metrics import mean_squared_error
5 from sklearn.preprocessing import LabelEncoder
6 from sklearn.preprocessing import StandardScaler
7 from sklearn.feature_selection import SelectFromModel
8 from keras.models import Sequential
9 from keras.layers import Dense, Activation, BatchNormalization, Dropout
10 # from keras.layers.advanced_activations import LeakyReLU, PReLU
11 from keras.wrappers.scikit_learn import KerasRegressor
12 from sklearn.model_selection import cross_val_score
13 from sklearn.model_selection import KFold
14 # from keras.optimizers import SGD
15 # from sklearn.pipeline import Pipeline
16 import xgboost as xgb
```

```

17 import matplotlib.pyplot as plt
18 import os
19 os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
20
21
22 # -----
23 # Functions
24 # All functions to prepare the data and evaluating the models
25 #
26 def rmse(y_true, y_pred):
27     """
28     The error metric. Rooted mean squared error of the logarithm of the sale
29     prices. Assumes the input is already the logarithm.
30
31     Parameters
32
33     y_true: logarithm of the true sale prices
34     y_pred: logarithm of the predicted sale prices
35
36     Returns
37
38     Rooted mean squared error
39     """
40
41     return np.sqrt(mean_squared_error(y_true, y_pred))
42
43
44 def prepare(train_df, test_df):
45     """
46     Prepares the data for training. This also includes feature engineering.
47     All of this has been copied from Eigen habe rollt den dritten Teil der eigenen Forschung raus
48     https://www.kaggle.com/humananalog/xgboost-lasso/
49
50     Parameters
51
52     train_df: dataframe with training data
53     test_df: dataframe with test data
54
55     Returns
56
57     Train dataframe, labels of the training data and test dataframe
58     """
59
60     # There are a few houses with more than 4000 sq ft living area that are
61     # outliers, so we drop them from the training data. (There is also one in
62     # the test set but we obviously can't drop that one.)
63     train_df.drop(train_df[train_df["GrLivArea"] > 4000].index, inplace=True)
64
65     # The test example with ID 666 has GarageArea, GarageCars, and GarageType
66     # but none of the other fields, so use the mode and median to fill them in.
67     test_df.loc[666, "GarageQual"] = "TA"
68     test_df.loc[666, "GarageCond"] = "TA"
69     test_df.loc[666, "GarageFinish"] = "Unf"
70     test_df.loc[666, "GarageYrBlt"] = "1980"
71
72     # The test example 1116 only has GarageType but no other information. We'll
73     # assume it does not have a garage.
74     test_df.loc[1116, "GarageType"] = np.nan
75
76     # For imputing missing values: fill in missing LotFrontage values by the
77     # median LotFrontage of the neighborhood.
78     lot_frontage_by_neighborhood = \
79         train_df["LotFrontage"].groupby(train_df["Neighborhood"])
80
81     # Used to convert categorical features into ordinal numbers.
82     # (There's probably an easier way to do this, but it works.)
83     le = LabelEncoder()
84
85     def factorize(df, factor_df, column, fill_na=None):
86         factor_df[column] = df[column]
87         if fill_na is not None:
88             factor_df[column].fillna(fill_na, inplace=True)
89         le.fit(factor_df[column].unique())
90         factor_df[column] = le.transform(factor_df[column])

```

```

58     return factor_df
59
60 # Combine all the (numerical) features into one big DataFrame. We don't add
61 # the one-hot encoded variables here yet, that happens later on.
62 def munge(df):
63     all_df = pd.DataFrame(index=df.index)
64
65     all_df["LotFrontage"] = df["LotFrontage"]
66     for key, group in lot_frontage_by_neighborhood:
67         idx = (df["Neighborhood"] == key) & (df["LotFrontage"].isnull())
68         all_df.loc[idx, "LotFrontage"] = group.median()
69
70     all_df["LotArea"] = df["LotArea"]
71
72     all_df["MasVnrArea"] = df["MasVnrArea"]
73     all_df["MasVnrArea"].fillna(0, inplace=True)
74
75     all_df["BsmtFinSF1"] = df["BsmtFinSF1"]
76     all_df["BsmtFinSF1"].fillna(0, inplace=True)
77
78     all_df["BsmtFinSF2"] = df["BsmtFinSF2"]
79     all_df["BsmtFinSF2"].fillna(0, inplace=True)
80
81     all_df["BsmtUnfSF"] = df["BsmtUnfSF"]
82     all_df["BsmtUnfSF"].fillna(0, inplace=True)
83
84     all_df["TotalBsmtSF"] = df["TotalBsmtSF"]
85     all_df["TotalBsmtSF"].fillna(0, inplace=True)
86
87     all_df["1stFlrSF"] = df["1stFlrSF"]
88     all_df["2ndFlrSF"] = df["2ndFlrSF"]
89     all_df["GrLivArea"] = df["GrLivArea"]
90
91     all_df["GarageArea"] = df["GarageArea"]
92     all_df["GarageArea"].fillna(0, inplace=True)
93
94     all_df["WoodDeckSF"] = df["WoodDeckSF"]
95     all_df["OpenPorchSF"] = df["OpenPorchSF"]
96     all_df["EnclosedPorch"] = df["EnclosedPorch"]
97     all_df["3SsnPorch"] = df["3SsnPorch"]
98     all_df["ScreenPorch"] = df["ScreenPorch"]
99
100    all_df["BsmtFullBath"] = df["BsmtFullBath"]
101    all_df["BsmtFullBath"].fillna(0, inplace=True)
102
103    all_df["BsmtHalfBath"] = df["BsmtHalfBath"]
104    all_df["BsmtHalfBath"].fillna(0, inplace=True)
105
106    all_df["FullBath"] = df["FullBath"]
107    all_df["HalfBath"] = df["HalfBath"]
108    all_df["BedroomAbvGr"] = df["BedroomAbvGr"]
109    all_df["KitchenAbvGr"] = df["KitchenAbvGr"]
110    all_df["TotRmsAbvGrd"] = df["TotRmsAbvGrd"]
111    all_df["Fireplaces"] = df["Fireplaces"]
112
113    all_df["GarageCars"] = df["GarageCars"]
114    all_df["GarageCars"].fillna(0, inplace=True)
115
116    all_df["CentralAir"] = (df["CentralAir"] == "Y") * 1.0
117
118    all_df["OverallQual"] = df["OverallQual"]
119    all_df["OverallCond"] = df["OverallCond"]
120
121    # Quality measurements are stored as text but we can convert them to
122    # numbers where a higher number means higher quality.
123
124    qual_dict = {None: 0, "Po": 1, "Fa": 2, "TA": 3, "Gd": 4, "Ex": 5}
125    all_df["ExterQual"] = df["ExterQual"].map(qual_dict).astype(int)
126    all_df["ExterCond"] = df["ExterCond"].map(qual_dict).astype(int)
127    all_df["BsmtQual"] = df["BsmtQual"].map(qual_dict).astype(int)
128    all_df["BsmtCond"] = df["BsmtCond"].map(qual_dict).astype(int)

```

```

230     # Most properties use standard circuit breakers.
231     all_df["IsElectricalSBrkr"] = (df["Electrical"] == "SBrkr") * 1
232
233     # About 2/3rd have an attached garage.
234     all_df["IsGarageDetached"] = (df["GarageType"] == "Detchd") * 1
235
236     # Most have a paved drive. Treat dirt/gravel and partial pavement
237     # as "not paved".
238     all_df["IsPavedDrive"] = (df["PavedDrive"] == "Y") * 1
239
240     # The only interesting "misc. feature" is the presence of a shed.
241     all_df["HasShed"] = (df["MiscFeature"] == "Shed") * 1.
242
243     # If YearRemodAdd != YearBuilt, then a remodeling took place at some
244     # point.
245     all_df["Remodeled"] = \
246         (all_df["YearRemodAdd"] != all_df["YearBuilt"]) * 1
247
248     # Did a remodeling happen in the year the house was sold?
249     all_df["RecentRemodel"] = \
250         (all_df["YearRemodAdd"] == all_df["YrSold"]) * 1
251
252     # Was this house sold in the year it was built?
253     all_df["VeryNewHouse"] = (all_df["YearBuilt"] == all_df["YrSold"]) * 1
254
255     all_df["Has2ndFloor"] = (all_df["2ndFlrSF"] == 0) * 1
256     all_df["HasMasVnr"] = (all_df["MasVnrArea"] == 0) * 1
257     all_df["HasWoodDeck"] = (all_df["WoodDeckSF"] == 0) * 1
258     all_df["HasOpenPorch"] = (all_df["OpenPorchSF"] == 0) * 1
259     all_df["HasEnclosedPorch"] = (all_df["EnclosedPorch"] == 0) * 1
260     all_df["Has3SsnPorch"] = (all_df["3SsnPorch"] == 0) * 1
261     all_df["HasScreenPorch"] = (all_df["ScreenPorch"] == 0) * 1
262
263     # These features actually lower the score a little.
264     # we still use them for later selecting the best features
265     all_df["HasBasement"] = df["BsmtQual"].isnull() * 1
266     all_df["HasGarage"] = df["GarageQual"].isnull() * 1
267     all_df["HasFireplace"] = df["FireplaceQu"].isnull() * 1
268     all_df["HasFence"] = df["Fence"].isnull() * 1
269
270     # Months with the largest number of deals may be significant.
271     all_df["HighSeason"] = df["MoSold"].replace(
272         {1: 0, 2: 0, 3: 0, 4: 1, 5: 1, 6: 1, 7: 1, 8: 0, 9: 0, 10: 0, 11:
273             0, 12: 0})
274
275     all_df["NewerDwelling"] = df["MSSubClass"].replace(
276         {20: 1, 30: 0, 40: 0, 45: 0, 50: 0, 60: 1, 70: 0, 75: 0, 80: 0, 85:
277             0, 90: 0, 120: 1, 150: 0, 160: 0, 180: 0, 190: 0})
278
279     all_df.loc[df.Neighborhood == 'NridgHt', "Neighborhood_Good"] = 1
280     all_df.loc[df.Neighborhood == 'Crawfor', "Neighborhood_Good"] = 1
281     all_df.loc[df.Neighborhood == 'StoneBr', "Neighborhood_Good"] = 1
282     all_df.loc[df.Neighborhood == 'Somerst', "Neighborhood_Good"] = 1
283     all_df.loc[df.Neighborhood == 'NoRidge', "Neighborhood_Good"] = 1
284     all_df["Neighborhood_Good"].fillna(0, inplace=True)
285
286     all_df["SaleCondition_PriceDown"] = df.SaleCondition.replace(
287         {'Abnormal': 1, 'Alloca': 1, 'AdjLand': 1, 'Family': 1, 'Normal': 0,
288             'Partial': 0})
289
290     # House completed before sale or not
291     all_df["BoughtOffPlan"] = df.SaleCondition.replace(
292         {"Abnorml": 0, "Alloca": 0, "AdjLand": 0, "Family": 0,
293             "Normal": 0, "Partial": 1})
294
295     all_df["BadHeating"] = df.HeatingQC.replace(
296         {'Ex': 0, 'Gd': 0, 'TA': 0, 'Fa': 1, 'Po': 1})
297
298     area_cols = ['LotFrontage', 'LotArea', 'MasVnrArea', 'BsmtFinSF1',
299                 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF',
300                 '2ndFlrSF', 'GrLivArea', 'GarageArea', 'WoodDeckSF',

```

```

372     "ClearCr": 3,  # 200250
373     "Crawfor": 3,  # 200624
374     "Veenker": 3,  # 218000
375     "Somerst": 3,  # 225500
376     "Timber": 3,   # 228475
377     "StoneBr": 4,  # 278000
378     "NoRidge": 4,  # 290000
379     "NridgHt": 4,  # 315000
380 }
381
382     all_df["NeighborhoodBin"] = df["Neighborhood"].map(neighborhood_map)
383
384
385     return all_df
386
387
388     train_df_munged = munge(train_df)
389     test_df_munged = munge(test_df)
390
391     # print(train_df_munged.shape)
392     # print(test_df_munged.shape)
393
394
395     # Copy NeighborhoodBin into a temporary DataFrame because we want to use
396     # the unscaled version later on (to one-hot encode it).
397     neighborhood_bin_train = pd.DataFrame(index=train_df.index)
398     neighborhood_bin_train["NeighborhoodBin"] = \
399         train_df_munged["NeighborhoodBin"]
400     neighborhood_bin_test = \
401         pd.DataFrame(index=test_df.index)
402     neighborhood_bin_test["NeighborhoodBin"] = \
403         test_df_munged["NeighborhoodBin"]
404
405 #####
406
407     numeric_features = train_df_munged.dtypes[train_df_munged.dtypes !=
408                                                 "object"].index
409
410
411     # Transform the skewed numeric features by taking log(feature + 1).
412     # This will make the features more normal.
413
414     skewed = train_df_munged[numeric_features].apply(lambda x:
415                                                       skew(x.dropna(),
416                                                       astype(float)))
417
418     skewed = skewed[skewed > 0.75]
419     skewed = skewed.index
420
421
422     train_df_munged[skewed] = np.log1p(train_df_munged[skewed])
423     test_df_munged[skewed] = np.log1p(test_df_munged[skewed])
424
425
426     # Additional processing: scale the data.
427     scaler = StandardScaler()
428     scaler.fit(train_df_munged[numeric_features])
429
430
431     scaled = scaler.transform(train_df_munged[numeric_features])
432     for i, col in enumerate(numeric_features):
433         train_df_munged[col] = scaled[:, i]
434
435
436     scaled = scaler.transform(test_df_munged[numeric_features])
437     for i, col in enumerate(numeric_features):
438         test_df_munged[col] = scaled[:, i]
439
440 #####
441
442     # Convert categorical features using one-hot encoding.
443     def onehot(onehot_df, df, column_name, fill_na, drop_name):
444         onehot_df[column_name] = df[column_name]
445         if fill_na is not None:
446             onehot_df[column_name].fillna(fill_na, inplace=True)
447
448         dummies = pd.get_dummies(onehot_df[column_name], prefix="_" +
449                                 column_name)
450
451         onehot_df = onehot_df.join(dummies)
452         onehot_df = onehot_df.drop([column_name], axis=1)

```

```

114     yearbin_df["GarageYrBltBin"].fillna("NoGarage", inplace=True)
115
116     yearbin_df["YearBuiltBin"] = df.YearBuilt.map(year_map)
117     yearbin_df["YearRemodAddBin"] = df.YearRemodAdd.map(year_map)
118
119     onehot_df = onehot(onehot_df, yearbin_df, "GarageYrBltBin", None, None)
120     onehot_df = onehot(onehot_df, yearbin_df, "YearBuiltBin", None, None)
121     onehot_df = onehot(onehot_df, yearbin_df, "YearRemodAddBin", None,
122                         None)
123
124     return onehot_df
125
126
127     # Add the one-hot encoded categorical features.
128     onehot_df = munge_onehot(train_df)
129     onehot_df = onehot(onehot_df, neighborhood_bin_train, "NeighborhoodBin",
130                         None, None)
131     train_df_munged = train_df_munged.join(onehot_df)
132
133
134     # These onehot columns are missing in the test data, so drop them from the
135     # training data or we might overfit on them.
136     drop_cols = [
137         "_Exterior1st_ImStucco", "_Exterior1st_Stone",
138         "_Exterior2nd_Other", "_HouseStyle_2.5Fin",
139
140         "_RoofMatl_Membran", "_RoofMatl_Metal", "_RoofMatl_Roll",
141         "_Condition2_RRAe", "_Condition2_RRAn", "_Condition2_RRNn",
142         "_Heating_Floor", "_Heating_OthW",
143
144         "_Electrical_Mix",
145         "_MiscFeature_TenC",
146         "_GarageQual_Ex", "_PoolQC_Fa"
147     ]
148     train_df_munged.drop(drop_cols, axis=1, inplace=True)
149
150
151     onehot_df = munge_onehot(test_df)
152     onehot_df = onehot(onehot_df, neighborhood_bin_test, "NeighborhoodBin",
153                         None, None)
154     test_df_munged = test_df_munged.join(onehot_df)
155
156
157     # This column is missing in the training data. There is only one example
158     # with this value in the test set. So just drop it.
159     test_df_munged.drop(["_MSSubClass_150"], axis=1, inplace=True)
160
161
162     # Drop these columns. They are either not very helpful or they cause
163     # overfitting.
164     drop_cols = [
165         "_Condition2_PosN",      # only two are not zero
166         "_MSZoning_C (all)",   ,
167         "_MSSubClass_160",
168     ]
169     train_df_munged.drop(drop_cols, axis=1, inplace=True)
170     test_df_munged.drop(drop_cols, axis=1, inplace=True)
171
172 #####
173
174     # We take the log here because the error metric is between the log of the
175     # SalePrice and the log of the predicted price. That does mean we need to
176     # exp() the prediction to get an actual sale price.
177     label_df = pd.DataFrame(index=train_df_munged.index, columns=["SalePrice"])
178     label_df["SalePrice"] = np.log(train_df["SalePrice"])
179
180
181     # print("Training set size:", train_df_munged.shape)
182     # print("Test set size:", test_df_munged.shape)
183     return (train_df_munged, label_df, test_df_munged)
184
185
186 def evaluate_model(X, y, model, seed, n_splits, epochs, batch_size):
187     """
188         Evaluates the given model using the Keras wrapper for sklearn for k-fold
189         cross validation.
190
191
```

```

656     not fully connected layers with half the amount of neurons of the previous
657     layer
658     """
659
660     model = Sequential()
661
662     # Input Dropout and Normalization
663     model.add(Dropout(dropout_ratio, input_shape=(input_dim,)))
664     model.add(BatchNormalization())
665
666     # Fully connected layers
667     for i in range(0, 3):
668         model.add(Dense(input_dim, kernel_initializer="normal",
669                         activation="relu"))
670         model.add(Dropout(dropout_ratio))
671         model.add(BatchNormalization())
672
673     # More fully connected layers with less neurons than the previous ones
674     if input_dim >= 200:
675         model.add(Dense(200, kernel_initializer="normal", activation="relu"))
676         model.add(Dropout(dropout_ratio))
677         model.add(BatchNormalization())
678     if input_dim >= 100:
679         model.add(Dense(100, kernel_initializer="normal", activation="relu"))
680         model.add(Dropout(dropout_ratio))
681         model.add(BatchNormalization())
682     model.add(Dense(50, kernel_initializer="normal", activation="relu"))
683     model.add(Dropout(dropout_ratio))
684     model.add(BatchNormalization())
685     # for some reason two layers with 50 neurons improved performance
686     model.add(Dense(50, kernel_initializer="normal", activation="relu"))
687     model.add(Dropout(dropout_ratio))
688     model.add(BatchNormalization())
689     model.add(Dense(25, kernel_initializer="normal", activation="relu"))
690     model.add(Dropout(dropout_ratio))
691     model.add(BatchNormalization())
692     model.add(Dense(12, kernel_initializer="normal", activation="relu"))
693     model.add(Dropout(dropout_ratio))
694     model.add(BatchNormalization())
695
696     # Output layer
697     model.add(Dense(1, kernel_initializer="normal"))
698     model.compile(loss="mean_squared_error", optimizer="adam")
699     return model
700
701
702 def model_xlm(input_dim, layers=5, dropout_ratio=0.2):
703     """
704     Extreme Learning machine deep neural network architecture with relu,
705     Dropout and Batch Normalization after every layer.
706     Worse than model_0
707     """
708     if input_dim < 200:
709         raise ValueError("Not enough features for this model!")
710
711     model = Sequential()
712
713     # Input Dropout and Normalization
714     model.add(Dropout(dropout_ratio, input_shape=(input_dim,)))
715     model.add(BatchNormalization())
716
717     # Fully connected layers
718     for i in range(0, layers):
719         model.add(Dense(input_dim, kernel_initializer="normal",
720                         activation="relu"))
721         model.add(Dropout(dropout_ratio))
722         model.add(BatchNormalization())
723
724     # Output layer
725     model.add(Dense(1, kernel_initializer="normal"))
726     model.compile(loss="mean_squared_error", optimizer="adam")

```

```

798     # Transform the data with the chosen threshold
799     selection = SelectFromModel(regr, threshold=threshold,
800                               prefit=True)
801     X_sel = selection.transform(X)
802     X_submission = selection.transform(X_submission)
803     print(X_sel.shape[1], "features left after feature selection")
804
805     print("Evaluating model...")
806     # show_learning(model_0(X_sel.shape[1]), X_sel, y)
807     # Evaluate the model with cross validation on the training data
808     print(evaluate_model(X_sel, y, lambda input_dim=X_sel.shape[1]:
809             model_0(input_dim), seed=4321, n_splits=10,
810             epochs=230, batch_size=200))
811
812     print("Evaluating XGB...")
813     np.random.seed(4321)
814     kfold = KFold(n_splits=10, random_state=4321)
815     model = xgb.XGBRegressor(colsample_bytree=0.2, gamma=0.0,
816                               learning_rate=0.01, max_depth=4,
817                               min_child_weight=1.5, n_estimators=7200,
818                               reg_alpha=0.9, reg_lambda=0.6, subsample=0.2,
819                               seed=42, silent=1)
820     results = np.sqrt(-cross_val_score(model, X, y, cv=kfold, verbose=2,
821                                         scoring="neg_mean_squared_error"))
822     print(results.mean(), results.std())
823
824     # Train the model without validation_split and create the submission file
825     print("Training DNN for submission...")
826     model = model_0(X_sel.shape[1])
827     model.fit(X_sel, y, batch_size=200, epochs=300, verbose=2,
828               validation_split=0)
829     y_pred = np.exp(model.predict(X_submission))
830
831     pred_df = pd.DataFrame(y_pred, index=test_df["Id"],
832                            columns=["SalePrice"])
833     pred_df.to_csv("submission_0.csv", header=True, index_label='Id')
834
835     print("Creating XGB submission...")
836     pred_df = pd.DataFrame(y_pred_xgb, index=test_df["Id"],
837                            columns=["SalePrice"])
838     pred_df.to_csv("submission_xgb.csv", header=True, index_label='Id')

```

.../biatt09/Bange_Burkowitz_Harnisch/plot_feature_scores.py

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from pylab import rcParams
4
5 rcParams['figure.figsize'] = 10, 5.8
6
7 threshold, score, std = np.loadtxt("feature_scores.csv", unpack=True,
8                                   delimiter=",")
9
10 plt.errorbar(threshold, score, yerr=std, fmt=".")
11 plt.xlabel("Feature Score Threshold")
12 plt.ylabel("CV Score")
13 plt.ylim(0.113, 0.135)
14 plt.grid()
15 # plt.show()
16 plt.savefig("feature_scores.pdf")

```

Zeit	Raum	Abgabe im Moodle; Mails mit Betreff: [SMD1718]
Di. 10-12	CP-03-150	philipp2.hoffmann@udo.edu und jan.soedingrekso@udo.edu
Di. 16-18	P1-02-110	felix.neubuerger@udo.edu und tobias.hoinka@udo.edu
Di. 16-18	CP-03-150	simone.mender@udo.edu und maximilian.meier@udo.edu

Aufgabe 29: *Likelihoodkurve*

6 P.

Aus einer Poisson-Verteilung werden 3 Stichproben, nämlich die Zahlen 13, 8 und 9 entnommen.

- a) Berechnen Sie die negative Log-Likelihood-Funktion als Funktion des einzigen Parameters λ und stellen Sie sie graphisch dar.
- b) Bei welchem Wert von λ liegt das Minimum $-\ln \mathcal{L}_{\max}$?
- c) Für welche Werte von λ nimmt $-\ln \mathcal{L}$ die Werte

$$\begin{aligned} -\ln \mathcal{L}_{\max} + \frac{1}{2}, \\ -\ln \mathcal{L}_{\max} + 2 \quad \text{und} \\ -\ln \mathcal{L}_{\max} + \frac{9}{2} \end{aligned}$$

an und was sagen diese Werte aus?

- d) Vergleichen Sie diese Werte mit der Näherung über eine Taylor-Entwicklung 2. Ordnung, indem Sie die Näherung sowohl zusammen mit der Likelihood graphisch darstellen als auch die Werte aus c) bestimmen. Wofür könnte die Näherung nützlich sein?

Aufgabe 30: *F-Praktikum*

7 P.

In einem Praktikumsversuch werden folgende Werte gemessen:

$\Psi / {}^\circ$	Asymmetrie	$\Psi / {}^\circ$	Asymmetrie	$\Psi / {}^\circ$	Asymmetrie
0	-0,032	30	0,010	60	0,057
90	0,068	120	0,076	150	0,080
180	0,031	210	0,005	240	-0,041
270	-0,090	300	-0,088	330	-0,074

Die Asymmetriewerte haben einen Messfehler von $\pm 0,011$. Die Theorie sagt, dass die Asymmetrie durch einen Ansatz der Form

$$f(\Psi) = A_0 \cos(\Psi + \delta)$$

beschrieben wird.

- a) Machen Sie zunächst den Ansatz

$$f(\Psi) = a_1 f_1(\Psi) + a_2 f_2(\Psi)$$

mit

$$f_1(\Psi) = \cos(\Psi) \quad \text{und} \quad f_2(\Psi) = \sin(\Psi)$$

und schreiben Sie die Designmatrix \mathbf{A} auf.

- b) Berechnen Sie den Lösungsvektor \mathbf{a} für die Parameter nach der Methode der kleinsten Quadrate.
c) Berechnen Sie die Kovarianzmatrix $\mathbf{V}[\mathbf{a}]$ sowie die Fehler von a_1 und a_2 und den Korrelationskoeffizienten.
d) Berechnen Sie A_0 und δ und deren Fehler und Korrelation aus a_1 und a_2 .

7 P.

Aufgabe 31: Regularisierte kleinste Quadrate

Ein Kollege hat für Sie eine Verteilung gemessen. Diese Verteilung wird Teil einer Monte-Carlo-Simulation. Um besser mit der Verteilung arbeiten zu können, suchen Sie nach einer geeigneten Parametrisierung. Sie wissen, dass sich die Verteilung durch ein Polynom sechsten Grades gut beschreiben lässt. Jedoch ist die Messung stark verrauscht und Ihr Kollege war auch nur in der Lage acht Wertepaare (x, y) zu nehmen.

- a) Fitten sie mit der Methode der kleinsten Quadrate ein Polynom sechsten Grades an die Daten der Datei *aufg_a.csv*. Geben Sie die resultierenden Koeffizienten an und zeichnen Sie das gefittete Polynom und die Daten in eine Abbildung ein.
b) Fitten sie mit der Methode der kleinsten Quadrate ein Polynom sechsten Grades an die Daten der Datei *aufg_a.csv* und nutzen Sie dabei zusätzlich die Regularisierung über die zweite Ableitung ($\Gamma = \sqrt{\lambda} C A$). Für die Regularisierungsstärke nutzen sie $\lambda \in (0.1, 0.3, 0.7, 3, 10)$. Geben Sie die resultierenden Koeffizienten an und zeichnen Sie das gefittete Polynom und die Daten in eine Abbildung.

Ihr Kollege macht sich die Mühe und fertigt 50 neue Messungen des Spektrums an.

- c) Fitten Sie mit der Methode der kleinsten Quadrate ein Polynom sechsten Grades an die Mittelwerte der Daten aus der Datei *aufg_c.csv*. Gewichten Sie die berechneten Mittelwerte mit dem Fehler des Mittelwerts. Nutzen Sie diese Gewichte beim Fitten. Zeichnen Sie das gefittete Polynom und die gemittelten Daten in eine Abbildung ein.

Aufgabe 31

Die Variablennamen im Code richten sich alle nach der Vorlesung, wegen der fehlenden Kommentare sei daher auf diese verwiesen. Das zu fittende Polynom hat die Form *test*

$$p(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6. \quad (1)$$

a)

Für die Koeffizienten erhalten wir

$$\begin{aligned} a_0 &= -6,74453270 \cdot 10^{-2} \\ a_1 &= 6,09609038 \cdot 10^{-1} \\ a_2 &= -5,13748213 \cdot 10^{-1} \\ a_3 &= 2,10566521 \cdot 10^{-1} \\ a_4 &= -4,52007751 \cdot 10^{-2} \\ a_5 &= 4,78568049e \cdot 10^{-3} \\ a_6 &= -1,96288196e \cdot 10^{-4}. \end{aligned} \quad (2)$$

Das Ergebnis ist in Abbildung 1 dargestellt.

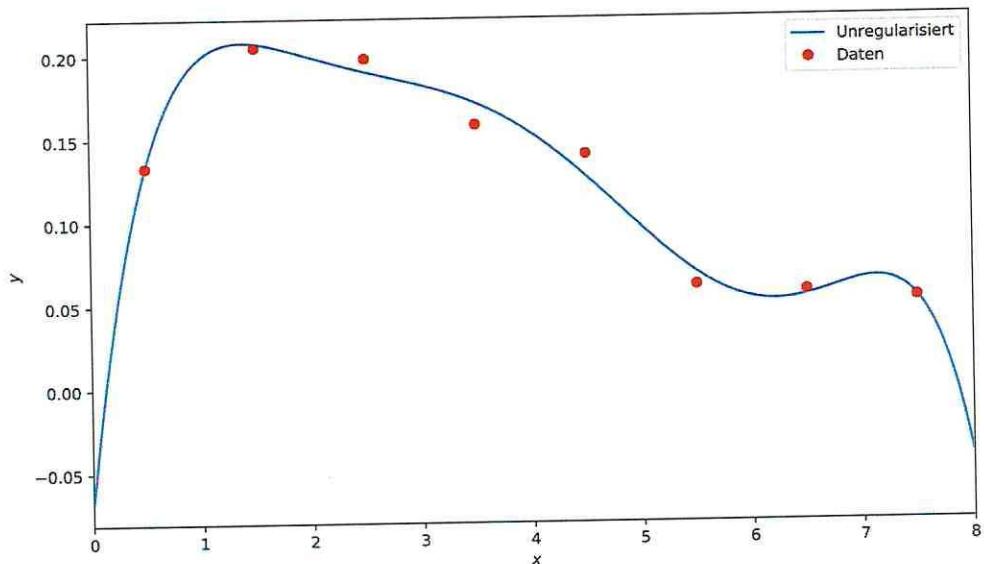


Abbildung 1: Geringste Quadrate Fit ohne Regularisierung an die Daten *aufg_a.csv*.

AZ 9

$$a) -\ln L = -\ln \left(\frac{\lambda^{13}}{13!} \cdot \frac{\lambda^8}{8!} \cdot \frac{\lambda^9}{9!} e^{-3\lambda} \right)$$

$$= -30 \ln(\lambda) + 3\lambda + \underbrace{\ln(13! 8! 9!)}_A$$

$$b) (-\ln L)' = -\frac{30}{\lambda} + 3 = 0$$

$$\Leftrightarrow \lambda = 10$$

$$(-\ln L)'' \Big|_{\lambda=10} = \frac{30}{\lambda^2} \Big|_{\lambda=10} = 0,3 > 0$$

$$c) -\ln L(10) = -30 \ln(10) + 30 + A \approx 6,88 =: -\ln L_{\max}$$

$$-\ln L(\lambda_{1/2}) \stackrel{!}{=} 6,88 + \frac{1}{2} = -30 \ln(\lambda_{1/2}) + 3\lambda_{1/2} + A$$

$$-\ln L(\lambda_2) = 6,88 + 2, \quad -\ln L(\lambda_{9/2}) = 6,88 + 9/2$$

$$\text{Python: } \lambda_{1/2} = 8,28 \quad \lambda_2 = 6,78 \quad \lambda_{9/2} = 5,47 \\ \lambda_{1/2} = 11,9 \quad \lambda_2 = 14,1 \quad \lambda_{9/2} = 16,5$$

Dies sind die 10-, 20-, bzw. 30-Umgebungen.

10-Grenze bei $F(\lambda = \lambda_{\min} \pm r_{\text{rechts}}) = F(\lambda_{\min}) \pm \frac{r^2}{2}$

$$d) -\ln L = -\ln L_{\max} + 0 \cdot (\lambda - 10) + \frac{1}{2} \cdot 0,3 (\lambda - 10)^2 + O((\lambda - 10)^3)$$

$$= -\ln L_{\max} + 0,15 (\lambda - 10)^2 + O((\lambda - 10)^3)$$

$$\lambda_{1/2}' = 8,12 \quad \lambda_2' = 6,35 \quad \lambda_{9/2}' = 4,52$$

$$\lambda_{1/2}' = 11,8 \quad \lambda_2' = 13,7 \quad \lambda_{9/2}' = 15,5$$

Die Taylorentwicklung vereinfacht das analytische Rechnen um das Minimum.

✓

5/6

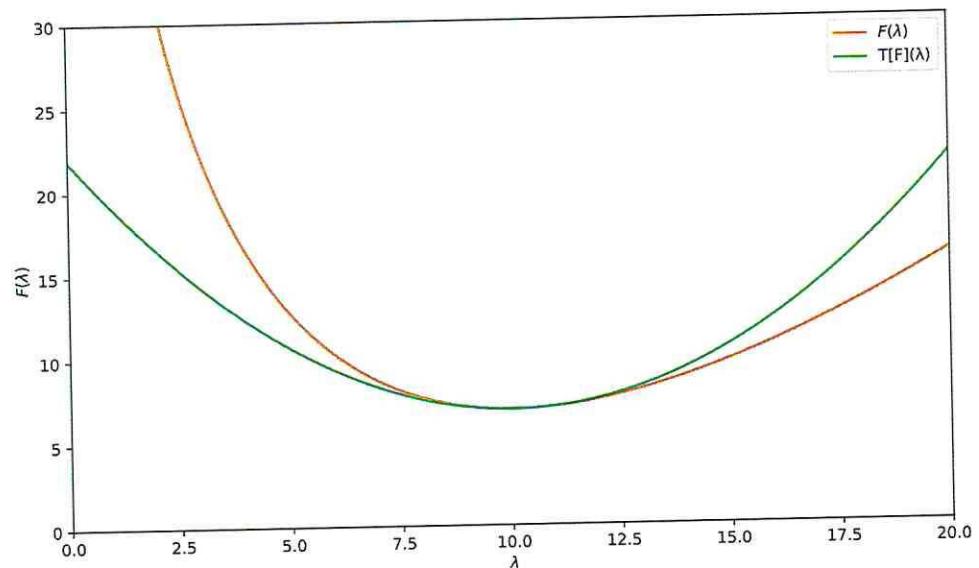


Abbildung 5: Negative Log-Likelihood-Funktion, zusammen mit der Taylorentwicklung um das Minimum.

$$d) a_1 \cos(\varphi) + a_2 \sin(\varphi) = A_0 \cos(\varphi + \delta)$$

$$= A_0 \cos(\varphi) \cos(\delta) - A_0 \sin(\varphi) \sin(\delta)$$

$$\Rightarrow a_1 = A_0 \cos(\delta) \quad I \quad \Leftrightarrow A_0 = \frac{a_1}{\cos(\delta)}$$

$$a_2 = -A_0 \sin(\delta) \quad II \quad \Leftrightarrow A_0 = -\frac{a_2}{\sin(\delta)}$$

$$I=II \Rightarrow \frac{a_1}{a_2} = -\frac{\cos(\delta)}{\sin(\delta)} = -\cot(\delta)$$

$$\Rightarrow \delta = \arccot\left(-\frac{a_1}{a_2}\right) \approx \underline{\underline{64,1^\circ}} + \text{faktisch an tan}$$

$$\Rightarrow A_0 = \frac{a_1}{\cos(\arccot(-\frac{a_1}{a_2}))} \approx -0,0860$$

$$\Delta \delta = \sqrt{\left(\frac{\partial \delta}{\partial a_1} \Delta a_1\right)^2 + \left(\frac{\partial \delta}{\partial a_2} \Delta a_2\right)^2} \quad \text{folge Tabelle}$$

$$= \sqrt{\left(-\frac{a_2}{a_1^2 + a_2^2} \Delta a_1\right)^2 + \left(\frac{a_1}{a_1^2 + a_2^2} \Delta a_2\right)^2} \approx 141^\circ$$

$$\Delta A_0 = \sqrt{\left(\frac{\partial A_0}{\partial a_1} \Delta a_1\right)^2 + \left(\frac{\partial A_0}{\partial \delta} \Delta \delta\right)^2}$$

$$= \sqrt{\left(\frac{1}{\cos(\delta)} \Delta a_1\right)^2 + \left(a_1 \tan(\delta) \sec(\delta) \Delta \delta\right)^2} \approx 0,651$$

$\frac{s}{\sqrt{2}}$

Code für blatt10

Bange, Burkowitz, Harnisch

22. Januar 2018

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy.misc import factorial
4 from scipy.optimize import newton
5 from pylab import rcParams
6
7 rcParams['figure.figsize'] = 10, 5.8
8 rcParams['legend.numpoints'] = 1
9
10
11 # ----- Funktionen -----
12 def F(l):
13     return -30*np.log(l) + 3*l +
14         np.log(factorial(13)*factorial(8)*factorial(9))
15
16
17 def taylorF(l):
18     return F(10) + 0.15*(l - 10)**2
19
20
21 def aufg29a():
22     x = np.linspace(.001, 50, 1000)
23     plt.xlim(.001, 50)
24     plt.ylim(0, 100)
25     plt.xlabel("$\lambda$")
26     plt.ylabel("$F(\lambda)$")
27     plt.plot(x, F(x))
28     # plt.show()
29     plt.savefig("A29a.pdf")
30
31
32 def aufg29c():
33     print(newton(lambda x: F(x) - F(10) - 0.5, 9), "< λ <",
34           newton(lambda x: F(x) - F(10) - 0.5, 11))
35
36     print(newton(lambda x: F(x) - F(10) - 2, 9), "< λ <",
37           newton(lambda x: F(x) - F(10) - 2, 11))
38
39     print(newton(lambda x: F(x) - F(10) - 4.5, 7), "< λ <",
40           newton(lambda x: F(x) - F(10) - 4.5, 12))
41
42
43 def aufg29d():
44     print(newton(lambda x: taylorF(x) - F(10) - 0.5, 9), "< λ <",
45           newton(lambda x: taylorF(x) - F(10) - 0.5, 11))
46
47     print(newton(lambda x: taylorF(x) - F(10) - 2, 9), "< λ <",
48           newton(lambda x: taylorF(x) - F(10) - 2, 11))
49
50     print(newton(lambda x: taylorF(x) - F(10) - 4.5, 7), "< λ <",
51           newton(lambda x: taylorF(x) - F(10) - 4.5, 12))
52
53     x = np.linspace(.001, 20, 1000)
54     plt.xlim(.001, 20)
55     plt.ylim(0, 30)
56     plt.xlabel("$\lambda$")
57     plt.plot(x, F(x), label="$F(\lambda)$")
```

```

58     plt.plot(x, taylorF(x), label=r"$\rm{T}[F](\lambda)$")
59     plt.legend()
60     # plt.show()
61     plt.savefig("A29d.pdf")
62
63
64 # _____ Main _____
65 if __name__ == "__main__":
66     aufg29a()
67     aufg29c()
68     aufg29d()

```

.../blatt10/Bange_Burkowitz_Harnisch/aufg30.py

```

1 import numpy as np
2 import uncertainties.unumpy as unp
3 from uncertainties import ufloat
4
5
6 psi, y = np.loadtxt("data.txt", unpack=True)
7 psi = np.deg2rad(psi)
8 # a)
9 A = np.vstack((np.cos(psi), np.sin(psi))).T
10
11 # b)
12 a = np.linalg.inv(A.T@A)@A.T@y
13 print("Lösungsvektor a =", a)
14
15 # c)
16 X_v2 = np.average((y - A@a)**2/0.011**2)/2
17 print("Chi_v^2 =", X_v2)
18 a_err = np.sqrt(1/6*X_v2)
19 print("sqrt(1/6*Chi_v^2) =", a_err)
20 a1 = ufloat(a[0], a_err)
21 a2 = ufloat(a[1], a_err)
22
23 # d)
24 delta = unp.arctan(-a2/a1)
25 A0 = a1/unp.cos(delta)
26 print("A0 =", A0)
27 print("delta =", delta*180/np.pi)
28
29 delta = np.arctan(-a[1]/a[0])
30 delta_err = np.sqrt((-a[1]/(a[0]**2 + a[1]**2)*a_err)**2
31             + (a[0]/(a[0]**2 + a[1]**2)*a_err)**2)
32 A0 = a[0]/np.cos(delta)
33 A0_err = np.sqrt((a_err/np.cos(delta))**2
34                   + (a[0]*np.tan(delta)/np.cos(delta)
35                     *delta_err)**2)
36 print(np.rad2deg(delta_err))
37 print(A0_err)

```

.../blatt10/Bange_Burkowitz_Harnisch/aufg31.py

```

1 import numpy as np
2 from scipy.sparse import diags
3 import matplotlib.pyplot as plt
4 from pylab import rcParams
5
6 rcParams['figure.figsize'] = 10, 5.8
7 rcParams['legend.numpoints'] = 1
8
9
10 # _____ Funktionen _____
11 def poly(x, a0, a1, a2, a3, a4, a5, a6):
12     return a0 + a1*x + a2*x**2 + a3*x**3 + a4*x**4 + a5*x**5 + a6*x**6
13
14
15 def aufg31a(x, y, A):
16     print("a")

```

```

17 a = np.linalg.inv(A.T@A) @ A.T@y
18 print("Koeffizienten:", a)
19 xr = np.linspace(min(x) - 0.5, max(x) + 0.5, 1000)
20 plt.xlim(min(x) - 0.5, max(x) + 0.5)
21 plt.plot(xr, poly(xr, *a), label="Unregularisiert")
22 plt.plot(x, y, "ro", label="Daten")
23 plt.legend()
24 plt.xlabel("$x$")
25 plt.ylabel("$y$")
26 # plt.show()
27 plt.savefig("A3la.pdf")
28 plt.clf()
29 return a
30
31
32 def aufg3lb(x, y, A, a):
33     print("b)")
34     C = np.asarray(diags([np.ones(A.shape[0])*(-2), np.ones(A.shape[0] - 1),
35                           np.ones(A.shape[0] - 1)], [0, -1, 1]).todense())
36     C[0][0] = -1
37     C[-1][-1] = -1
38
39     ls = [0.1, 0.3, 0.7, 3, 10]
40     a_reg = []
41     for l in ls:
42         a_reg.append(np.linalg.inv(A.T@A + l*(C@A).T@(C@A)) @ A.T@y)
43
44     xr = np.linspace(min(x) - 0.5, max(x) + 0.5, 1000)
45     plt.xlim(min(x) - 0.5, max(x) + 0.5)
46     plt.plot(xr, poly(xr, *a), label="Unregularisiert")
47     for a, l in zip(a_reg, ls):
48         print("Koeffizienten mit  $\lambda = {}$ : {}".format(l, a))
49         plt.plot(xr, poly(xr, *a), label="$\lambda = {}$".format(l))
50     plt.plot(x, y, "ro", label="Daten")
51     plt.legend()
52     plt.xlabel("$x$")
53     plt.ylabel("$y$")
54     # plt.show()
55     plt.savefig("A3lb.pdf")
56     plt.clf()
57
58
59 def aufg3lc():
60     X = np.loadtxt("aufg_c.csv", delimiter=", ", skiprows=1)
61     x = X[:, 0]
62     y = np.mean(X[:, 1:], axis=1)
63     A = np.vstack((x**i for i in range(0, 7))).T
64     y_err = np.std(X[:, 1:], axis=1) — dof? / N? für Fehler des Mittelwerts
65     W = np.diag(1/y_err)
↳ hier fließen die Degrees of Freedom ein! => dof = 1
66
67     a = np.linalg.inv(A.T@A) @ A.T@y
68     a_weighted = np.linalg.inv(A.T@W@A) @ A.T@W@y
69
70     xr = np.linspace(min(x) - 0.5, max(x) + 0.5, 1000)
71     plt.xlim(min(x) - 0.5, max(x) + 0.5)
72     plt.plot(xr, poly(xr, *a_weighted), label="Gewichtet")
73     plt.plot(xr, poly(xr, *a), "—", label="Ungewichtet")
74     plt.errorbar(x, y, yerr=y_err, fmt="ro", label="Daten")
75     plt.legend()
76     plt.xlabel("$x$")
77     plt.ylabel("$y$")
78     # plt.show()
79     plt.savefig("A3lc.pdf")
80     plt.clf()
81
82
83 # ----- Main -----
84 if __name__ == "__main__":
85     x, y = np.loadtxt("aufg_a.csv", unpack=True, delimiter=", ", skiprows=1)
86     A = np.vstack((x**i for i in range(0, 7))).T
87     a = aufg3la(x, y, A)

```

```
58     aufg31b(x, y, A, a)
59     aufg31c()
```

Zeit	Raum	Abgabe im Moodle; Mails mit Betreff: [SMD1718]
Di. 10-12	CP-03-150	philipp2.hoffmann@udo.edu und jan.soedingrekso@udo.edu
Di. 16-18	P1-02-110	felix.neubuerger@udo.edu und tobias.hoinka@udo.edu
Di. 16-18	CP-03-150	simone.mender@udo.edu und maximilian.meier@udo.edu

Aufgabe 32: *Gamma-Astronomie*

5 P.

Bei dieser Aufgabe handelt es sich um eine Fortführung der Aufgabe 25 γ -Astronomie (Blatt 8). Jetzt soll festgestellt werden ob sich an der Position, auf die das Teleskop gerichtet war, wirklich eine γ -Quelle befindet. Hierzu wird die Nullhypothese verwendet. Zur Erinnerung die Likelihoodfunktion lautete

$$\ln L = -F = N_{\text{off}} \ln(b) + N_{\text{on}} \ln(s + \alpha b) - (1 + \alpha)b - s - \ln(N_{\text{off}}!) - \ln(N_{\text{on}}!) \quad (1)$$

und folgende Werte für s und b machten diese Likelihood maximal:

$$\hat{s} = N_{\text{on}} - \alpha N_{\text{off}} \quad (2)$$

$$\hat{b} = N_{\text{off}} \quad (3)$$

- Die Nullhypothese besagt, dass es gar keine γ -Quelle gibt, also $s_0 = 0$. Welcher Wert und welcher Fehler ergeben sich unter dieser Annahme für b_0 nach der Methode der maximalen Likelihood?
- Wie lautet das Verhältnis λ der beiden Likelihoods?
- Unter den gegebenen Hypothesen und mit großen N_{on} , N_{off} ist $D = -2 \ln \lambda$ χ^2 -verteilt mit einem Freiheitsgrad. Mit welcher Konfidenz lehnen Sie die Nullhypothese ab? Geben Sie Ihr Ergebnis in Einheiten von Sigma an.

Tipp: Betrachten Sie eine standardnormalverteilte Variable u . Welcher Verteilung folgt u^2 ? Vergleichen Sie mit D .

- Berechnen Sie die Signifikanz für die Messung eines Signals für folgende Zahlenbeispiele:
 - $N_{\text{on}} = 120$, $N_{\text{off}} = 160$, $\alpha = 0,6$.
 - $N_{\text{on}} = 150$, $N_{\text{off}} = 320$, $\alpha = 0,3$.

Aufgabe 33: χ^2 -Test

5 P.

In einem Experiment werden 7 verschiedene Energiedifferenzen mit den Werten ¹

$$31,6 \text{ meV}, \quad 32,2 \text{ meV}, \quad 31,2 \text{ meV}, \quad 31,9 \text{ meV}, \\ 31,3 \text{ meV}, \quad 30,8 \text{ meV}, \quad 31,3 \text{ meV}$$

mit jeweils einem Fehler von 0,5 meV gemessen.

- Hypothese A sagt einen Wert von 31,3 meV für diese Messgröße voraus. Machen Sie einen χ^2 -Test und entscheiden Sie, ob die These bei 5 % gewählter Signifikanz verworfen werden muss, oder nicht.
- Wie a), aber mit der Hypothese B, die den Wert 30,7 meV vorhersagt.

Aufgabe 34: Likelihood-Quotienten-Test

5 P.

In einer Honigfabrik wird je eine Portion Honig in ein Glas zu μ_0 Millilitern abgefüllt. Es wird angenommen, dass die Füllmengen produktionsbedingt einer Normalverteilung mit Mittelwert $\mu = \mu_0$ und einer unbekannten Varianz σ^2 folgen.

- Stellen Sie die Testbedingung für einen Likelihood-Quotienten-Test auf, in dem Sie die Nullhypothese von oben gegen die Gegenhypothese, dass die Füllmenge einer Normalverteilung folgt, die *nicht* den Mittelwert μ_0 hat, testen.
- Für jeweils welche Wahl der Parameter μ und σ^2 werden die Likelihood-Funktionen der einzelnen Hypothesen auf dem jeweiligen Parameterbereich maximal?
- Setzen Sie die in b) erhaltenen Parameter ein und reduzieren Sie die Testbedingung auf einen Ausdruck der einer t-Statistik folgt.
Hinweis: Die Größe $T = \sqrt{N}(\bar{x} - \mu_0)/s$ folgt unter der Nullhypothese der t-Statistik, wobei s die Stichprobenvarianz ist.
- Es wird aufgrund der eingesetzten Maschinen eine Füllmenge von $\mu_0 = 200$ ml erwartet. Aus einer Stichprobe mit 25 Messungen zur Qualitätskontrolle wird ein Mittelwert von $\bar{x} = 205$ ml bei einer geschätzten Standardabweichung von $s = 10$ ml gemessen. Wird die oben aufgestellte Nullhypothese bei einer Signifikanz von 5 % abgewiesen oder beibehalten?

¹Das Beispiel kommt aus der Festkörperphysik

5 P.

Aufgabe 35: *Teilchenidentifikation*

In einem Experiment der Teilchenphysik wird ein Čerenkov-Zähler zur Teilchenidentifikation verwendet. Das Messergebnis des Zählers kann in Form von Likelihood-Ratios angegeben werden. Für eine bestimmte Teilchenspur ergibt sich jeweils

- a) $L_\pi : L_K : L_p = 0,13 : 1,5 : 0,5$
- b) $L_\pi : L_K : L_p = 2,0 : 0,5 : 0,05$
- c) $L_\pi : L_K : L_p = 0,07 : 0,5 : 1,3$

Es ist bekannt, dass unter den gegebenen experimentellen Bedingungen 80 % der Teilchen Pionen, 10 % Kaonen und 10 % Protonen sind (*Prior Information*). Wie groß ist die Wahrscheinlichkeit, dass jeweils ein Pion, ein Kaon oder ein Proton beobachtet wurde?

$$A 32 \quad a) -F(0, b) = N_{\text{eff}} \ln(b) + N_{\text{on}} \ln(\alpha b) - (1+\alpha)b - \ln(N_{\text{eff}}!) - \ln(N_{\text{on}}!)$$

$$-\frac{dF(0, b)}{db} = \frac{N_{\text{eff}}}{b} + \frac{N_{\text{on}}}{b} - 1 - \alpha \stackrel{!}{=} 0$$

$$\Rightarrow b_0 = \frac{N_{\text{eff}} + N_{\text{on}}}{1 + \alpha}$$

$$\sigma^2 = \left(\frac{d^2 F(0, b)}{db^2} \right)^{-1} \Big|_{b=b_0} = \left(\frac{N_{\text{eff}} + N_{\text{on}}}{b_0^2} \right)^{-1} = \frac{N_{\text{eff}} + N_{\text{on}}}{(1 + \alpha)^2}$$

$$b) \lambda = \frac{\zeta_0}{2} = \exp\left(\ln\left(\frac{\zeta_0}{2}\right)\right) = \exp\left(\ln(\zeta_0) - \ln(2)\right) = \exp(-F_0 - (-F))$$

$$= \exp\left[N_{\text{eff}} \ln\left(\frac{b_0}{b}\right) + N_{\text{on}} \ln\left(\frac{\alpha b_0}{2 + \alpha b}\right) - (1 + \alpha)(b_0 - b) + \tilde{s}\right]$$

$$= \exp\left[N_{\text{eff}} \ln\left(\frac{1 + \frac{N_{\text{on}}}{N_{\text{eff}}}}{1 + \alpha}\right) + N_{\text{on}} \ln\left(\alpha \frac{N_{\text{eff}} + N_{\text{on}}}{1 + \alpha} \cdot \frac{1}{N_{\text{on}} - \alpha N_{\text{eff}} + \alpha N_{\text{eff}}}\right) - (1 + \alpha) \left(\frac{N_{\text{eff}} + N_{\text{on}}}{1 + \alpha} - N_{\text{eff}}\right) + N_{\text{on}} - \alpha N_{\text{eff}}\right]$$

$$= \exp\left[N_{\text{eff}} \ln\left(\frac{1 + \frac{N_{\text{on}}}{N_{\text{eff}}}}{1 + \alpha}\right) + N_{\text{on}} \ln\left(\alpha \cdot \frac{1 + \frac{N_{\text{eff}}}{N_{\text{on}}}}{1 + \alpha}\right) - N_{\text{eff}} - N_{\text{on}} + N_{\text{eff}} + \alpha N_{\text{eff}} + N_{\text{on}} - \alpha N_{\text{eff}}\right]$$

$$= \exp\left[N_{\text{eff}} \ln\left(\frac{1 + \frac{N_{\text{on}}}{N_{\text{eff}}}}{1 + \alpha}\right) + N_{\text{on}} \ln\left(\alpha \cdot \frac{1 + \frac{N_{\text{eff}}}{N_{\text{on}}}}{1 + \alpha}\right)\right]$$

c) χ^2 ist $\chi^2(1)$ verteilt

$$\Rightarrow \chi^2 \sim \chi^2 \quad \wedge \quad D \sim \chi^2$$

$$\Rightarrow \sqrt{D} \sim |u| \sim |N(0, 1)|$$

$\Rightarrow \sqrt{D} \equiv$ Konfidenzschranke in Einheiten von σ

d) i) $\sqrt{D} \approx 1,8363 \sigma$ Signifikanz Level = p-Value = p

$$p = 1 - 2 \cdot \int_0^{\sqrt{D}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \approx 0,0663 = 6,63\%$$

$$\text{ii) } \sqrt{D} \approx 4,3724 \sigma$$

✓

$$p \approx 0,0000120 = 0,00120\%$$

9/5

A33)

$$\chi^2 = \sum_{i=1}^n \frac{(x_i - \mu)^2}{\sigma_i^2}$$
$$a) \chi^2 = \frac{(37,6 \text{ meV} - 37,3 \text{ meV})^2}{(0,5 \text{ meV})^2} + \frac{(32,2 \text{ meV} - 31,3 \text{ meV})^2}{(0,5 \text{ meV})^2} + \dots$$

$$\approx 6,08$$

$$\chi^2(p=0,05, \text{DoF}=2) = 14,067 > 6,08$$

$\Rightarrow H_0$ kann nicht abgelehnt werden

$$b) \chi^2 \approx 21,9$$

$$\chi^2(p=0,05, \text{DoF}=2) = 14,067 < 21,9$$

$\Rightarrow H_0$ wird abgelehnt



✓

A34/ a)

$$\Gamma = \frac{\sup_{\theta \in \Theta_0} L(\theta | X)}{\sup_{\theta \in \Theta} L(\theta | X)}$$
$$= \frac{\sup_{\theta=0} \prod_{i=1}^N \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2\sigma^2}(x_i - \mu_0)^2\right]}{\sup_{\theta \neq 0} \prod_{i=1}^N \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2\sigma^2}(x_i - \mu)^2\right]}$$

$\Gamma < k_\alpha \Rightarrow H_0$ verworfen

$\Gamma \geq k_\alpha \Rightarrow H_0$ annehmen

b) $L = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N \exp\left[-\frac{1}{2\sigma^2} \sum_i (x_i - \mu)^2\right]$

$$-\ln(L) = F = N \ln(\sqrt{2\pi}) + N \underbrace{\ln(\sigma)}_{\frac{1}{2} \ln(\sigma^2)} + \frac{1}{2\sigma^2} \sum_i (x_i - \mu)^2$$

$$\frac{\partial F}{\partial \sigma^2} = \frac{N}{2\sigma^2} - \frac{1}{2\sigma^4} \sum_i (x_i - \mu)^2 = 0$$

$$\Leftrightarrow \sigma^2 = \frac{1}{N} \sum_i (x_i - \mu)^2$$

$$\frac{\partial F}{\partial \mu} = -\frac{1}{\sigma^2} \sum_i (x_i - \mu) = 0$$

$$\Leftrightarrow \sum_i x_i = N\mu$$

$$\Leftrightarrow \mu = \frac{1}{N} \sum_i x_i = \bar{x}$$

c)

$$\Gamma = \frac{\left(\frac{1}{N} \sum_i (x_i - \mu_0)^2\right)^{-\frac{N}{2}}}{\left(\frac{1}{N} \sum_i (x_i - \bar{x})^2\right)^{-\frac{N}{2}}} \exp\left[-\frac{1}{2} \left(\left(\frac{1}{N} \sum_i (x_i - \mu_0)^2\right)^{-1} \sum_i (x_i - \mu_0)^2 - \left(\frac{1}{N} \sum_i (x_i - \bar{x})^2\right)^{-1} \sum_i (x_i - \bar{x})^2 \right)\right]$$
$$= \frac{\left(\sum_i (x_i - \mu_0)^2\right)^{-\frac{N}{2}}}{\left(\sum_i (x_i - \bar{x})^2\right)^{-\frac{N}{2}}} \exp\left[-\frac{1}{2} (N - N)\right]$$
$$= \frac{\left(\frac{1}{N-1} \sum_i (x_i - \mu_0)^2\right)^{-\frac{N}{2}}}{S^2}$$

$$\Rightarrow \Gamma^{-\frac{2}{N}} = \frac{1}{N-1} \sum_i (x_i - \mu_0)^2 / S^2$$
$$= \frac{1}{S^2(N-1)} \sum_i (x_i^2 - 2x_i \mu_0 + \mu_0^2)$$
$$= \frac{1}{S^2(N-1)} \left(\underbrace{\sum_i x_i^2}_{N(\mu_0 - \bar{x})^2} - \underbrace{2N\bar{x}\mu_0 + N\mu_0^2 + N\bar{x}^2 - N\bar{x}^2}_{N\bar{x} \frac{N}{N-1} \sum_i x_i} \right)$$

$$\begin{aligned}
 \Gamma^{-\frac{2}{N}} &= \frac{1}{s^2(N-1)} \left(\underbrace{\sum_i (x_i^2) - 2N\bar{x}\mu_0 + N\mu_0^2 + N\bar{x}^2 - \overline{\sum_i x_i^2}}_{N(\mu_0 - \bar{x})^2} \right) \\
 &= \frac{\sum_i (x_i^2 - \bar{x}x_i) + N(\mu_0 - \bar{x})^2}{(N-1)s^2} \\
 &= \frac{\sum_i (x_i^2 - \bar{x}x_i - \bar{x}x_i + \bar{x}x_i) + N(\mu_0 - \bar{x})^2}{(N-1)s^2} \\
 &= \frac{\sum_i (x_i^2 - 2\bar{x}x_i) + N\bar{x}^2 + N(\mu_0 - \bar{x})^2}{(N-1)s^2} \\
 &= \frac{\sum_i (x_i^2 - 2\bar{x}x_i + \bar{x}^2) + N(\mu_0 - \bar{x})^2}{(N-1)s^2} \\
 &= \frac{\sum_i (x_i - \bar{x})^2}{(N-1)s^2} + \frac{N(\mu_0 - \bar{x})^2}{(N-1)s^2} \\
 &= 1 + \frac{T^2}{N-1}
 \end{aligned}$$

$$\Leftrightarrow \sqrt{(N-1)(\Gamma^{-\frac{2}{N}} - 1)} = T$$

$$\begin{array}{l}
 \Gamma \stackrel{!}{\geq} k_\alpha \Leftrightarrow T \stackrel{!}{\leq} k_\alpha^* \\
 (\Gamma \stackrel{!}{<} k_\alpha \Leftrightarrow T \stackrel{!}{>} k_\alpha^*) \\
 \end{array}
 \quad \left. \begin{array}{l}
 \text{Ungleichung ändert sich nicht!} \\
 \text{Die Umformung wird (vermutlich) in der Tabelle für } T \text{ berücksichtigt!}
 \end{array} \right\}$$

k_α^* kann in Tabellen nachgeschlagen werden

$$d) T = \frac{\sqrt{25} (205 \text{ ml} - 200 \text{ ml})}{10 \text{ ml}} = 2,5$$

$\alpha = 5\%$; DoF = 24; einseitiger Test
 $\Rightarrow k_\alpha^* = 1,711 < |T| \Rightarrow H_0$ wird nicht abgelehnt

hier wird ein 2-seitiger Test gemacht

zweiseitiger Test:

$\Rightarrow k_\alpha^* = 2,064 < |T| \Rightarrow H_0$ wird nicht abgelehnt
sonst anders.



5/5

A35 Bayes Theorem:

$$p(H_i | D, I) = \frac{p(H_i | I) p(D | H_i, I)}{p(D | I)}$$

$$p(H_{\pi} | I) = 0,8$$

$$p(H_k | I) = 0,1$$

$$p(H_p | I) = 0,1$$

a) $p(D | I) = \sum_i p(H_i | I) p(D | H_i, I)$
 $= 0,8 \cdot 0,13 + 0,1 \cdot 0,5 + 0,1 \cdot 0,5 = 0,304$

$$\Rightarrow p(H_{\pi} | D, I) = \frac{0,8 \cdot 0,13}{0,304} \approx 34,2\%$$

$$p(H_k | D, I) = \frac{0,1 \cdot 0,5}{0,304} \approx 16,7\%$$

$$p(H_p | D, I) = \frac{0,1 \cdot 0,5}{0,304} \approx 16,7\%$$

b) $p(H_{\pi} | D, I) = 36,7\%$

$$p(H_k | D, I) = 3,02\%$$

$$p(H_p | D, I) = 0,302\%$$

c) $p(H_{\pi} | D, I) = 23,7\%$

$$p(H_k | D, I) = 21,2\%$$

$$p(H_p | D, I) = 55,2\%$$

✓

5/5

Zeit	Raum	Abgabe im Moodle; Mails mit Betreff: [SMD1718]
Di. 10-12	CP-03-150	philipp2.hoffmann@udo.edu und jan.soedingrekso@udo.edu
Di. 16-18	P1-02-110	felix.neubuerger@udo.edu und tobias.hoinka@udo.edu
Di. 16-18	CP-03-150	simone.mender@udo.edu und maximilian.meier@udo.edu

Aufgabe 36: *Entfaltung in zwei Intervallen*

6 P.

Betrachten Sie ein Experiment, in dem 2 Typen von Ereignissen gezählt werden. Die Wahrscheinlichkeit ein Ereignis dem falschen Typ zuzuordnen sei ϵ und die Wahrscheinlichkeit für den richtigen Typ $1 - \epsilon$. Die Wahrscheinlichkeit der falschen Zuordnung ist also für $1 \rightarrow 2$ die gleiche wie für $2 \rightarrow 1$. Des weiteren gehen bei dem Experiment leider 20% der Messungen verloren.

Die gemessenen Ereigniszahlen $\mathbf{g} = (g_1, g_2)^T$ sind Poisson-verteilt und unkorreliert.

- a) Stellen Sie die Antwortmatrix \mathbf{A} auf und den Zusammenhang zwischen \mathbf{A} , \mathbf{g} und der wahren Ereigniszahl $\mathbf{f} = (f_1, f_2)^T$.
- b) Berechnen Sie \mathbf{f} als Funktion von \mathbf{g} und ϵ .
- c) Berechnen Sie die Kovarianzmatrix $\mathbf{V}[\mathbf{f}]$ als Funktion von \mathbf{g} und ϵ .
- d) Berechnen Sie \mathbf{f} , $\mathbf{V}[\mathbf{f}]$, die Fehler von f_1 und f_2 und den Korrelationskoeffizienten für $g_1 = 200$, $g_2 = 169$ und $\epsilon = 0,1$.
- e) Wie d), aber mit $\epsilon = 0,4$. Was hat sich geändert und was bedeutet dies?
- f) Was passiert bei $\epsilon = 0,5$?

Aufgabe 37: *Entfaltung mit quadratischen Matrizen*

7 P.

In dieser Aufgabe sollen die gezeigten Plots aus der Vorlesung nachgebaut werden, um die Vorgänge bei der Entfaltung besser zu verstehen.

- a) Die quadratische $n \times n$ Antwortmatrix \mathbf{A} wird hier nicht aus MC erzeugt, sondern ist vorgegeben:

$$A = \begin{pmatrix} 1 - \epsilon & \epsilon & 0 & 0 & \dots & 0 \\ \epsilon & 1 - 2\epsilon & \epsilon & 0 & \dots & 0 \\ 0 & \epsilon & 1 - 2\epsilon & \epsilon & \dots & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & \dots & 0 & \epsilon & 1 - 2\epsilon & \epsilon \\ 0 & \dots & 0 & 0 & \epsilon & 1 - \epsilon \end{pmatrix}$$

Was für einen Messprozess beschreibt die Matrix A ? Im Folgenden sei $\epsilon = 0, 23$. Implementieren sie eine Methode, mit der die Matrix A für beliebige Dimensionen $n \geq 3$ erzeugt werden kann.

- b) Die wahre Verteilung f sei im Intervall $[0, 2]$ in 20 gleichgroßen Bins gegeben durch

$$f = [193, 485, 664, 763, 804, 805, 779, 736, 684, 626, \\ 566, 508, 452, 400, 351, 308, 268, 233, 202, 173]$$

Nun wird eine Messung simuliert, indem zuerst die wahre Verteilung f mit A gefaltet wird: $g = A \cdot f$. Die „gemessenen“ Bin-Einträge der Verteilung g^{mess} erhält man durch Ziehen aus einer Poisson-Verteilung mit dem Erwartungswert des Bin-Eintrags aus der vorherigen Faltung: $\lambda_i = g_i$. Dadurch ergeben sich statistische Schwankungen in der Messung, die die Entfaltung erschweren.

Beispiel: Die Faltung ergibt einen Wert λ_i in Bin i der Verteilung g . Dann ergibt sich der gemessene Eintrag g_i^{mess} durch eine Zufallszahl, die aus einer Poisson-Verteilung mit Erwartungswert λ_i gezogen wird.

- c) Transformieren Sie die Faltungsgleichung in die Diagonalbasis von $A = U \cdot D \cdot U^{-1}$. Wie lautet die Faltungsgleichung in der neuen Basis und welchen Vorteil bietet sie? Sortieren Sie die Eigenwerte absteigend nach ihrem Betrag und ändern Sie dementsprechend die Reihenfolge der zugehörigen Eigenvektoren in der Transformationsmatrix U .
- d) Transformieren Sie die Verteilungen $f \rightarrow b$ und $g \rightarrow c$ mit der Matrix U aus c) in die neue Basis. Berechnen Sie ebenfalls die Kovarianzmatrix der Verteilung b mittels der „BVB“-Formel. Normieren Sie die entfalteten Koeffizienten $b = D^{-1}U^{-1}g^{\text{mess}}$ auf ihre Standardabweichungen und stellen Sie die Einträge b_j in einem Plot gegen ihren Index dar. Was lässt sich über Koeffizienten b_j sagen, die in dem Plot unterhalb der 1 liegen?

Info: Falls Sie die Verteilung g nicht erzeugen konnten, benutzen Sie hier die folgenden Werte:

$$g = [256, 457, 630, 787, 774, 816, 771, 749, 665, 647, \\ 543, 510, 440, 410, 348, 296, 279, 243, 205, 174]$$

- e) Regularisieren Sie, indem Sie nicht signifikante Koeffizienten b_j ab einen Cutoff-Index auf 0 setzen. Entfalten sie dann je einmal mit Regularisierung und ohne. Stellen Sie beide Lösungen mit statistischen Fehlern in einem Plot zusammen mit der gegebenen wahren Verteilung dar. Was ist der Unterschied der Lösung mit und ohne Regularisierung?

Aufgabe 38: *Entfaltung: Implementierung*

7 P.

Zur Lösung dieser Aufgabe benötigen Sie die im moodle befindliche Datei `unfolding_data.hdf5`.

- a) Lesen Sie den Key `train` aus der Datei `unfolding_data.hdf5` aus dem moodle ein und stellen Sie alle vorhandenen Attribute zusammen mit `energy_true` in jeweils einem zwei-dimensionalen Histogram dar. Wählen Sie das ihnen am geeignetsten erscheinende Attribut als Entfaltungs-Observable aus.
- b) Berechnen Sie die Migrations-Matrix A . Nutzen Sie für die Observable 24 bins von 120 bis 500, und für die Zielvariable 16 bins von 15 bis 200. Stellen Sie die Migrationsmatrix grafisch dar.
- c) Nehmen Sie eine Poisson-Statistik für die Werte in \mathbf{g} an. Stellen Sie eine Likelihood-Funktion für die Verteilung der Zielvariable \mathbf{f} auf.
- d) Fügen Sie nun den Tikhonov-Regularisierungs-Term $\frac{\tau}{2} \|\Gamma \cdot \mathbf{f}\|^2$ hinzu. Berechnen Sie analytisch den Gradienten $(\nabla(-\log \mathcal{L}))(\mathbf{f})_i = \frac{\partial(-\log \mathcal{L})}{\partial f_i}$ und die Hesse-Matrix $H[-\log \mathcal{L}](\mathbf{f})_{ij} = \frac{\partial^2(-\log \mathcal{L})}{\partial f_i \partial f_j}$
- e) Implementieren Sie zur Minimierung der Likelihood das Newton-Verfahren gemäß der Rechenvorschrift

$$\mathbf{f}^{(k+1)} = \mathbf{f}^{(k)} - [H[-\log \mathcal{L}](\mathbf{f}) + \epsilon \mathbf{1}]^{-1} \cdot (\nabla(-\log \mathcal{L}))(\mathbf{f}) \quad (1)$$

und nutzen Sie $\epsilon = 10^{-6}$. Brechen Sie das Verfahren ab, sobald $\|\mathbf{f}^{(k+1)} - \mathbf{f}^{(k)}\| < 10^{-10}$ ist.

- f) Laden Sie nun den Key `test` aus der Datei `unfolding_data.hdf5` aus dem moodle und berechnen Sie mit dem Binning aus b) die Vektoren \mathbf{g} und \mathbf{f} . Entfalten Sie die Verteilung \mathbf{g} und vergleichen Sie diese mit der wahren Verteilung \mathbf{f} mit den Regularisierungsstärken $\tau = \{10^{-6}, 10^{-3}, 10^{-1}\}$. Welche Regularisierung erscheint angemessen?

A36

a) $g = Af$, $A = \begin{pmatrix} 1-\varepsilon & \varepsilon \\ \varepsilon & 1-\varepsilon \end{pmatrix}$

b) $f = A^{-1}g$, $A^{-1} = :B$

$$B = (1-2\varepsilon)^{-1} \begin{pmatrix} 1-\varepsilon & -\varepsilon \\ -\varepsilon & 1-\varepsilon \end{pmatrix}$$

$$f_{1/2} = (1-2\varepsilon)^{-1}((1-\varepsilon)g_{12} - \varepsilon g_{21})$$

c) Poisson \Rightarrow Fehler $\sqrt{n} = 7$ Varianz n

$$A \begin{pmatrix} g_1 \\ g_2 \end{pmatrix} = \begin{pmatrix} g_1 & 0 \\ 0 & g_2 \end{pmatrix}$$

$$V[f] = B V[g] B^T = (1-2\varepsilon)^{-2} \begin{pmatrix} (1-\varepsilon)^2 g_1 + \varepsilon^2 g_2 & -\varepsilon(1-\varepsilon)(g_1 + g_2) \\ -\varepsilon(1-\varepsilon)(g_1 + g_2) & (1-\varepsilon)^2 g_2 + \varepsilon^2 g_1 \end{pmatrix}$$

d) $f_1 = 203, 275, f_2 = 765, 725, f_1 + f_2 = 369 \checkmark$

$$V[f] = \begin{pmatrix} 255, 765 & 625 & -57, 890 & 625 \\ -57, 890 & 625 & 272, 075 & 625 \end{pmatrix}$$

$$\sigma_{f_1} = \sqrt{(V[f])_{11}}, \quad \sigma_{f_1} = 75, 9926741$$
$$\sigma_{f_2} = 74, 7374582$$

$$\rho = \frac{\text{cov}(f_1, f_2)}{\sigma_{f_1} \cdot \sigma_{f_2}} = -0, 22025324337796553$$

e) $f = (262, 707)^T \quad f_1 + f_2 = 369 \checkmark$

$$V[f] = \begin{pmatrix} 2476 & -2274 \\ -2274 & 2327 \end{pmatrix} \quad \sigma_{f_1} = 49, 75942722$$
$$\sigma_{f_2} = 48, 77675788$$

$$\rho = -0, 9235597729758676$$

Durch Fehlereingangsuntreue erhält man, dass das Ergebnis zufälliger, ob es spricht dich in den höheren Fehlern und der negativen Korrelation niedrigen.

f) Für $\varepsilon \rightarrow 0,5$ wird $V[f]$ singulär bzw. $\rho \rightarrow 1$.

Die Daten sind also vollkommen nutzlos, tippen sprachlichen keine Information.

A 37

c) $g = Af$, $A = U \cdot D \cdot U^T \Rightarrow U$ hat die EVs
 $\Leftrightarrow U^T g = U^T U D U^T f$, $\tilde{A} = D$ \Leftrightarrow die Spalten

$\Leftrightarrow \tilde{g} = D \tilde{f}$, $\tilde{g} = c$, $\tilde{f} = b$ caat Abfrage
 $\Leftrightarrow c = D b$

d) $V[g] = \text{diag}(s)$ (Position beehor)

$$\Rightarrow V[c] = U^T V[g] U$$

$$\Rightarrow V[b] = D^{-1} U^T V[g] U D^{-1} \quad ((D^{-1})^T = D^{-1})$$

e) $bf = Ub$

$$\Rightarrow V[f] = U V[b] U^T$$

b) $\delta = Af$ $f: \text{size } 76 \times 200$
 $g_i: \text{size } 24 \times 76$, $g_i \in \text{true-energy}$
 $24 \text{ bits}, 760-500, 76 \text{ bits}, 75 \text{ bits} \leq 200$

A (so A ist 24×76 Matrix)

$$c) p(k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

g_i seien die Measured

$$\lambda_i = A_i f \quad \text{a.l. so likelihood:}$$

$$L = \prod_{i=1}^{24} \frac{\lambda_i^{g_i} e^{-\lambda_i}}{g_i!} = \prod_{i=1}^{24} \frac{(A_i \cdot f)^{g_i} e^{-A_i \cdot f}}{g_i!}$$

$$F = -\log L = \sum_{i=1}^{24} [g_i \log(g_i!) + g_i \log(\lambda_i) - \lambda_i]$$

$$d) F_{reg} = F + \frac{\gamma}{2} \| Pf \|^2$$

$$\frac{\partial F_{reg}}{\partial f_i} = -\sum_{j=1}^{24} \left(g_j \frac{A_{ji} f_i}{A_j \cdot f} + A_{ji} f_i \right) + \frac{\partial}{\partial f_i} \frac{\gamma}{2} \left(\sum_{j=1}^{76} (\Pi_j f_j)^2 \right)$$

$$= -\sum_{j=1}^{24} A_{ji} (f_j - \frac{g_j}{A_j f}) + \gamma \sum_{j=1}^{76} (\Pi_j f_j) \cdot \Pi_j f_i$$

$$\frac{\partial^2 F}{\partial f_i \partial f_j} = \frac{\partial}{\partial f_j} \left(\frac{\partial F}{\partial f_i} \right) = + \sum_{k=1}^{24} \left[\frac{\gamma k}{A_{ki} f_j} + A_{kj} \delta_{ij} \right] + \gamma \left(\sum_{k=1}^{76} \Pi_{kj} f_j \Pi_{ki} f_i + \delta_{ij} \sum_{k=1}^{76} (\Pi_k f) \Pi_{kj} \right)$$

Mit Sicherheit falsch, ich versuch gen
nicht erst das zu implementieren/abzusagen
oder zu wenig Zeit.