aufgabe1

January 10, 2019

1 Aufgabe 28 - Entfaltung in zwei Intervallen

1.1 Teilaufgabe a)

Ohne Akzeptanzkorrektur würde die Responsematrix $\begin{pmatrix} 1-\epsilon & \epsilon \\ \epsilon & 1-\epsilon \end{pmatrix}$ lauten (Siehe Vorlesung). Mit Akzeptanzkorrektur und perfekter Messgenauigkeit gilt laut Blobel (Kap. 11.2)

$$\mathbf{f} = \mathbf{P}\,\mathbf{g}\,, \tag{1}$$

wobei $\mathbf{P}$ eine Diagonalmatrix ist mit den Akzeptanzen P_j auf der Hauptdiagonalen. In unserem Fall ist die Akzeptanz 80%. Also erscheint in unserem Fall

$$\mathbf{A} = \frac{4}{5}\begin{pmatrix} 1-\epsilon & \epsilon \\ \epsilon & 1-\epsilon \end{pmatrix} \tag{2}$$

mit $\mathbf{g} = \mathbf{A}\,\mathbf{f}$ sinnvoll zu sein; für $\epsilon = 0$ ergibt sich der Spezialfall aus dem Blobel.

1.2 Teilaufgabe b)

Die Faltungsgleichung lässt sich zu

$$\mathbf{f} = \mathbf{A}^{-1}\,\mathbf{g} \tag{3}$$

invertieren. Die inverse Matrix ist nach der bekannten Formel für die Inverse von 2x2-Matrizen

$$\mathbf{A}^{-1} = \frac{5}{4(1-2\epsilon)}\begin{pmatrix} 1-\epsilon & -\epsilon \\ -\epsilon & 1-\epsilon \end{pmatrix}\,. \tag{4}$$

Dann ergibt sich dann

$$\mathbf{f} = \frac{5}{4(1-2\epsilon)}\begin{pmatrix} (1-\epsilon)g_1 - \epsilon g_2 \\ -\epsilon g_1 + (1-\epsilon)g_2 \end{pmatrix}\,. \tag{5}$$

1.3 Teilaufgabe c)

Mit der Definition $\mathbf{B} = \mathbf{A}^{-1}$ folgt $\mathbf{f} = B\,\mathbf{g}$, sodass sich die Kovarianzmatrix von $\mathbf{f}$ nach der BvB-Formel zu

$$\mathbf{V}[\mathbf{f}] = \mathbf{B}\,\mathbf{V}[\mathbf{g}]\,\mathbf{B}^T \tag{6}$$

ergibt. Explizite Rechnung ergibt unter Beachtung unabhängiger poissonverteilter g_1 und g_2 ($\Rightarrow \sigma^2_{g_{1,2}} = g^2_{1,2}$) für die Kovarianzmatrix

$$\mathbf{V}[\mathbf{f}] = \frac{25}{16(1-2\epsilon)^2} \begin{pmatrix} (1-\epsilon)^2 g_1^2 + \epsilon^2 g_2^2 & -\epsilon(1-\epsilon)(g_1^2+g_2^2) \\ -\epsilon(1-\epsilon)(g_1^2+g_2^2) & \epsilon^2 g_1^2 + (1-\epsilon)^2 g_2^2 \end{pmatrix}. \tag{7}$$

1.4 Teilaufgabe d)

```
In [1]: import numpy as np


        def f(g, e):
            # Sehr komische Dinge passieren hier, wenn man g1, g2 = *g, mehrmals hintereinande
            # Warum ist das so?
            #g1, g2 = *g, # Das Komma macht aus g1, g2 ein Tupel und der Stern entpackt das Ar
            g1, g2 = g
            f = np.array([(1-e)*g1-e*g2, -e*g1+(1-e)*g2])
            return 5/(4*(1-2*e))*f


        def Vf(g, e):
            g1, g2 = g
            varf1 = (1-e)**2*g1**2+e**2*g1**2
            cov = -e*(1-e)*(g1**2+g2**2)
            varf2 = e**2*g1**2+(1-e)**2*g2**2
            Vf = np.array([[varf1, cov], [cov, varf2]])
            return 25/(16*(1-2*e)**2)*Vf


        g = np.array([200,169])
        e = 0.1
        f = f(g, e)
        print('Die wahre Ereigniszahl ist f =', f)
        Vf = Vf(g, e)
        print('Die Kovarianzmatrix von f ist V[f]')
        print(Vf)
        sigmaf1 = np.sqrt(Vf[0,0])
        sigmaf2 = np.sqrt(Vf[1,1])
        cov = Vf[0,1]
        rho = cov/(sigmaf1*sigmaf2)
        print('Der Fehler von f1 ist', sigmaf1)
```

```
        print('Der Fehler von f2 ist', sigmaf2)
        print('Der Korrelationskoeffizient rho ist', rho)
```

```
Die wahre Ereigniszahl ist f = [254.84375 206.40625]
Die Kovarianzmatrix von f ist V[f]
[[ 80078.125      -15064.67285156]
 [-15064.67285156  57457.05566406]]
Der Fehler von f1 ist 282.9807855667943
Der Fehler von f2 ist 239.70201430956413
Der Korrelationskoeffizient rho ist -0.22209105819766697
```

1.5 Teilaufgabe e)

```
In [2]: import numpy as np


        def f(g, e):
            # Sehr komische Dinge passieren hier, wenn man g1, g2 = *g, mehrmals hintereinande
            # Warum ist das so?
            # Deswegen Funktionsdef. und Aufruf in gleicher Zelle
            g1, g2 = *g, # Das Komma macht aus g1, g2 ein Tupel und der Stern entpackt das Arr
            f = np.array([(1-e)*g1-e*g2, -e*g1+(1-e)*g2])
            return 5/(4*(1-2*e))*f


        def Vf(g, e):
            g1, g2 = *g,
            varf1 = (1-e)**2*g1**2+e**2*g1**2
            cov = -e*(1-e)*(g1**2+g2**2)
            varf2 = e**2*g1**2+(1-e)**2*g2**2
            Vf = np.array([[varf1, cov], [cov, varf2]])
            return 25/(16*(1-2*e)**2)*Vf


        g = np.array([200,169])
        e = 0.4
        print(g)
        f = f(g, e)
        print('Die wahre Ereigniszahl ist f =', f)
        Vf = Vf(g, e)
        print('Die Kovarianzmatrix von f ist V[f]')
        print(Vf)
        sigmaf1 = np.sqrt(Vf[0,0])
        sigmaf2 = np.sqrt(Vf[1,1])
        cov = Vf[0,1]
        rho = cov/(sigmaf1*sigmaf2)
        print('Der Fehler von f1 ist', sigmaf1)
```

```python
        print('Der Fehler von f2 ist', sigmaf2)
        print('Der Korrelationskoeffizient rho ist', rho)
```

```
[200 169]
Die wahre Ereigniszahl ist f = [327.5  133.75]
Die Kovarianzmatrix von f ist V[f]
[[ 812500.     -642759.375 ]
 [-642759.375   651639.0625]]
Der Fehler von f1 ist 901.3878188659976
Der Fehler von f2 ist 807.241638234798
Der Korrelationskoeffizient rho ist -0.8833507454292174
```

f_1 und f_2 sind asymmetrischer geworden. Die gröSSere Ungenauigkeit der Messung führt zu einer ungenauer bekannten wahren Ergebniszahl $\mathbf{f}$, das ist recht logisch. Die beiden Komponenten von f sind stärker (negativ) korreliert als zuvor.

1.6 Teilaufgabe f)

Bei $\epsilon = 0.5$ handelt es sich um einen reinen Zufallsprozess, da mit 50 Prozent Wahrscheinlichkeit die Messung korrekt bzw. falsch zugeordnet wird. Dafür sind unsere Methoden der Entfaltung aber nicht gedacht. Bei gleicher Wahrscheinlichkeit der korrekten und falschen Zuordnung kann man aus den gemessenen Zahlen nicht auf die wahren Zahlen zurückschlieSSen, weswegen $\mathbf{A}$ nicht mehr invertierbar wird. Anschaulich gesprochen hat man keinerlei Kenntnis mehr über die Messung und ist gezwungen, arithmetisch zu mitteln, sodass man nur aussagen kann, dass f_1 und f_2 gleich groSS sind. Das ist keine wertvolle Information. Deswegen ist es nicht sinnvoll, Fehler anzugeben.