

# SMD-Übungsblatt 4

Abgabe: 16.11.18

Yvonne Kasper yvonne.kasper@udo.edu ,  
Robert Appel robert.appel@udo.edu ,  
Julian Schröer julian.schroeer@udo.edu

~~10  
4,5P | 5P | 12  
9,5P~~  
~~19/20 P NB.~~

## 1 Aufgabe1

### 1.1 a)

Teilaufgabe a) befindet sich auf den abgegebenen Zetteln.

### 1.2 b)

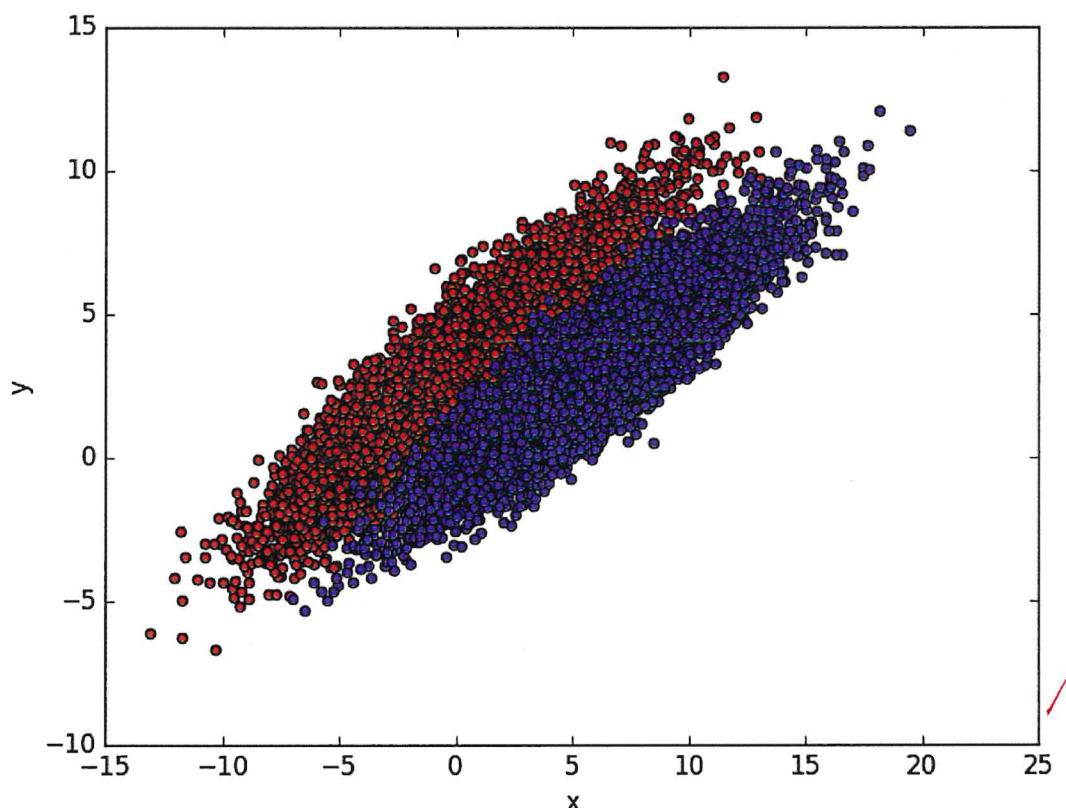


Abbildung 1: Scatterplot der beiden Populationen P\_0 und P\_1.

### 1.3 c)

Um die Korrelation  $\rho$  zu berechnen wurde folgende Formel benutzt.

$$\rho = \frac{\sigma_{xx} \cdot \sigma_{yy}}{\sigma_{xy}} \quad (1)$$

Tabelle 1: Mithilfe des Codes errechnete Werte.

	$\mu_x$	$\mu_y$	$\sigma_{xx}^2$	$\sigma_{yy}^2$	Kovarianz $\sigma_{xy} = \sigma_{yx}$	
Population 1	0.0029	2.9818	11.921	6.430	7.915 1.10	
Population 2	6.0153	3.1254	12.207	5.406	7.340 1.11	
Gesamt	3.0091	3.0536	21.101	5.922	7.843 1.43	✓

d) ~~korrekt -0,5P~~

## 2 Aufgabe 2

Aufgabe 2 befindet sich auf den abgegebenen Zetteln.

## 3 Aufgabe 3

- a) Die Mittelwerte  $\mu_{P0}, \mu_{P1}$  wurden nach den Methoden der VL berechnet in Python wurde dies wie folgt implementiert:

```
1 p1x = np.array(Pone['x'])
2 p1y = np.array(Pone['y'])
3 p0x = np.array(Pzero['x'])
4 p0y = np.array(Pzero['y'])
5 ##Berechnung der Mittelwertvektoren##
6 mup0 = np.array([np.mean(p0x), np.mean(p0y)])
7 mup1 = np.array([np.mean(p1x), np.mean(p1y)])
8 mup01 = np.array([np.mean(p01x), np.mean(p01y)])
9 ##Berechnung der Kovarianzmatrizen##
10 Vp0 = np.cov(p0x, p0y)
11 Vp1 = np.cov(p1x, p1y)
12 Vp01 = np.cov(np.append(p0x, p1x), np.append(p0y, p1y))
13 Vp2 = np.cov(p01x, p01y)
```

plots/Aufgabe3.py

Dabei werden in den Zeilen 4 bis 9 nur jeweils die x und y Einträge in arrays geschrieben. Die Mittelwerte dieser in ein 2D-array gesetzt ergeben die  $\mu_{P0} = \text{mup0}, \mu_{P1} = \text{mup1}$  Mittelwerte. Der Mittelwert mup001 ist für den Aufgabenteil h. ✓ *Code korrekt aber wo sind die Werte? -0,25P*

- b) Die Kovarianzmatrizen wurden mit einer Numpy Funktion berechnet.

```
1 print('Kovarianzmatrix p0:', Vp0)
2 print('Kovarianzmatrix p1:', Vp1)
3 print('Kovarianzmatrix p0 + p1:', Vp01)
4 print('Kovarianzmatrix p01:', Vp2)
5 print('Kovarianzmatrix p0 + p01:', Vp001)
```

plots/Aufgabe3.py

Kovarianzmatrix Population 0:

$$\begin{pmatrix} 12.20892862 & 8.15840984 \\ 8.15840984 & 6.72286327 \end{pmatrix} / \quad (2)$$

Kovarianzmatrix Population 1:

$$\begin{pmatrix} 12.35218537 & 7.4107561 \\ \cancel{X} 7.41075614 & 5.47731503 \end{pmatrix} / \quad (3)$$

Kovarianzmatrix Population 0 und 1:

$$\begin{pmatrix} 21.32208007 & 7.94257453 \\ 7.94257453 & 6.1025583 \end{pmatrix} / \quad (4)$$

Kovarianzmatrix Population 0\_1000:

$$\begin{pmatrix} 12.23612255 & 8.16049883 \\ 8.16049883 & 6.75819008 \end{pmatrix} / \quad (5)$$

Kovarianzmatrix Population 0 und 0\_1000:

$$\begin{pmatrix} 12.21067453 & 8.15842886 \\ 8.15842886 & 6.72630542 \end{pmatrix} / \quad (6)$$

c)

```

1 S1 = np.array([p - mup1 for p in p1.T]).T
2 S01 = np.array([p - mup01 for p in p1.T]).T
3 S0 = [np.matrix([s[0], s[1]]).T * (np.matrix([s[0], s[1]])) for s in S0.T]
4 S0 = reduce(lambda x,y: x+y, S0)
5 S1 = [np.matrix([s[0], s[1]]).T * (np.matrix([s[0], s[1]])) for s in S1.T]
6 S1 = reduce(lambda x,y: x+y, S1)
7 S01 = [np.matrix([s[0], s[1]]).T * (np.matrix([s[0], s[1]])) for s in S01.T]
8 S01 = reduce(lambda x,y: x+y, S01)
9 SW = S0 + S1 #Berechnung der Gesamtstreuung = Within class scatter matrix
10 SW001 = S0 + S01
11
12 lamb = SW.I * np.matrix([[mup0[0]-mup1[0]], [(mup0[1]-mup1[1])]]) #Berechnung der
   Projektion
13 lamb = lamb.T
14 lambnorm = np.sqrt(np.dot(lamb, lamb.T)) #lambda normierung
15 lamb/= lambnorm #Müsste jetzt Einheitsvektor sein
16 print('Normiertes Lambda p0:', lamb)
17 print(lambnorm)
18 lamb001 = SW.I * np.matrix([[mup0[0]-mup01[0]], [(mup0[1]-mup01[1])]]) #Berechnung der
   Projektion
19 lamb001 = lamb001.T
20 lambnorm001 = np.sqrt(np.dot(lamb001, lamb001.T)) #lambda normierung

```

plots/Aufgabe3.py

In den Zeilen eins bis drei werden hier nur einmal die Populationen zusammengefasst. Darauf werden die Streumatrizen wie aus der VL bestimmt (Zeilen 5-15). Die Lösung für die Fisher Diskriminante ist aus der VL bekannt und wird dann nur noch durch die gegebene Gleichung berechnet (Zeile 17). Danach wird diese nur noch normiert. Das selbe (Zeile 17-20) wird für den Aufgabenteil h wiederholt. Dann ergibt sich für die Population 0 und 1

$$\vec{\lambda} = \lambda \cdot \begin{pmatrix} -0.61886608 \\ 0.78549652 \end{pmatrix} / \quad (7)$$

und für die Population 0 und 0\_1000:

$$\vec{\lambda} = \lambda \cdot \begin{pmatrix} -0.46250292 \\ 0.88661776 \end{pmatrix} \quad (8)$$

d) Zur Berechnung der Projektion wurde folgende Funktion implementiert:

```
1 plt.hist(proj0, bins='auto', label='Population 0', histtype = 'step')
2
```

plots/Aufgabe3.py

Die Histogramme sind in den Abbildungen 2 und 3 (Aufgabenteil h) zu sehen.

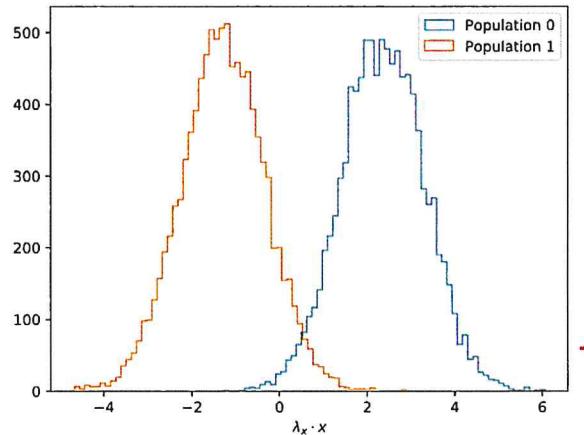


Abbildung 2: Projektion der Populationen 0 und 1.

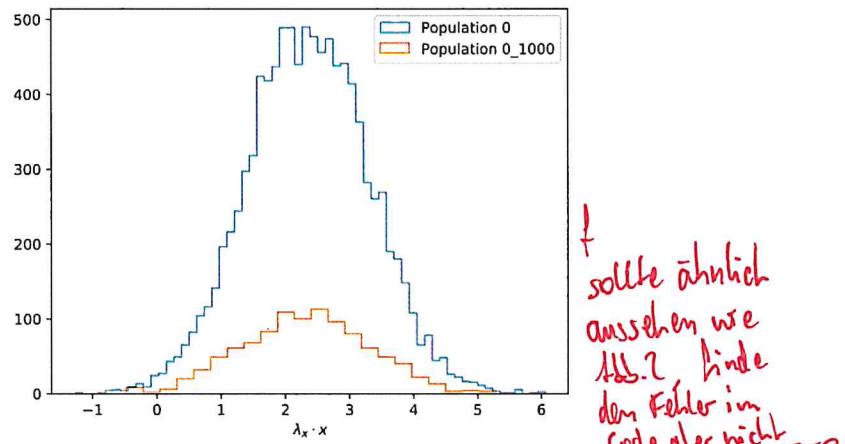


Abbildung 3: Projektion der Populationen 0 und 0\_1000.

e) Zur Bestimmung der Reinheit und Effizienz wurde folgende Funktion implementiert:

```

1  for l in lamcut:
2      projzero = projektion(lam,P0)
3      projone = projektion(lam,P1)
4      fn = projzero[projzero < 1].size #false negativ
5      tp = projzero[projzero > 1].size # true positiv
6      tn = projone[projone < 1].size #true negativ
7      fp = projone[projone > 1].size #false positiv
8      if (tp + fp) <= 0 : break
9      if (tp + fn) <= 0 : break
10     reinheit = np.append(reinheit , [tp/(tp+fp)])
11     effizienz = np.append(effizienz , [tp/(tp+fn)])
12 return reinheit , effizienz
13 ## reinheit und effizienz plot ##
14 lambcut = np.linspace(min(min(proj0),min(proj1)),max(max(proj0),max(proj1)),10**3)
15 lambcut2 = np.linspace(min(min(proj0),min(proj01)),max(max(proj0),max(proj01)),10**3)
16 rein , effi = lambdaCut(lambcut , lamb , p0 , p1)
17 rein2 , effi2 = lambdaCut(lambcut2 , lamb001 , p0 , p01)

```

plots/Aufgabe3.py

Die Ergebnisse sind in den Abbildungen 4 und 5 (AT h) dargestellt. Gesuchte  $\lambda_{Cut}$  Werte:

$$\lambda_{0\&1} = 2.13667347 \quad \text{und} \quad \lambda_{0\&0\_1} = 5.19070384 . \quad (9)$$

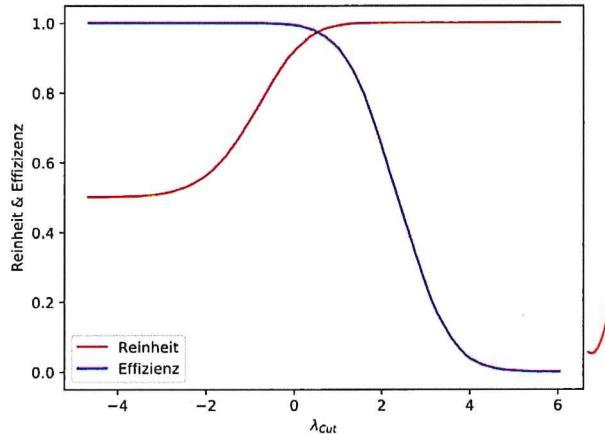


Abbildung 4: Reinheit und Effizienz - Populationen 0 und 1.

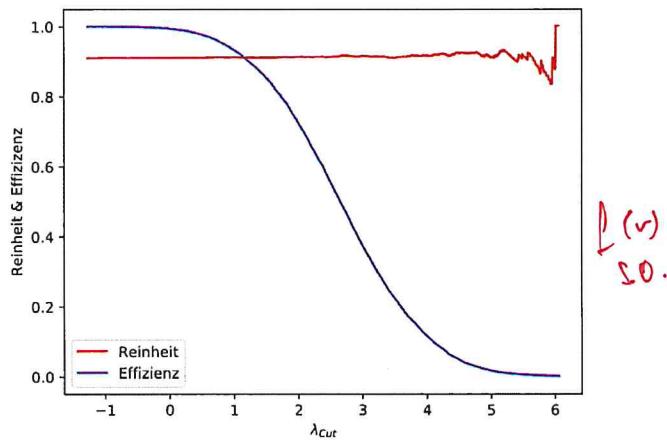


Abbildung 5: Reinheit und Effizienz - Populationen 0 und 0\_1000.

f) Zur Berechnung des Signal-Untergrundverhältnisses nach der Trennung wurde folgende Funktion implementiert.

```

1 tp = projzero[projzero > 1].size # true positiv
2 if projone[projone > 1].size <= 0: break
3 fp = projone[projone > 1].size #false positiv
4 SB = np.append(SB, tp/fp)
5 return SB
6 SBRatio = SBRatio(lambcut, lamb, p0,p1)
7 SBRatio2 = SBRatio(lambcut2, lamb001, p0,p01)
8 placeholder = lambcut[:SBRatio.size]
9 print('max sbr 1:',placeholder[SBRatio==max(SBRatio)])
10 placeholder = lambcut2[:SBRatio2.size]
```

plots/Aufgabe3.py

Die Ergebnisse sind in den Abbildungen 6 und 6 (AT h) dargestellt. Gesuchte  $\lambda_{Cut}$  Werte:

$$\lambda_{0\&1} = 0.52587484 \quad \text{und} \quad \lambda_{0\&0\_1} \approx -1.2 . \quad (10)$$

g) Zur Berechnung der Signifikanz nach der Trennung wurde folgende Funktion implementiert.

```

1 def Signiratio(lamcut, lam, P0, P1):
2     Sign = np.array([])
3     for l in lamcut:
4         projzero = projektion(lam,P0)
5         projone = projektion(lam,P1)
6         tp = projzero[projzero > 1].size # true positiv
7         fp = projone[projone > 1].size #false positiv
8         if np.sqrt((tp+fp))<= 0: break
9         Sign = np.append(Sign, tp/np.sqrt((tp+fp)))
10    return Sign
```

plots/Aufgabe3.py

Die Ergebnisse sind in den Abbildungen 8 und 9 (AT h) dargestellt.

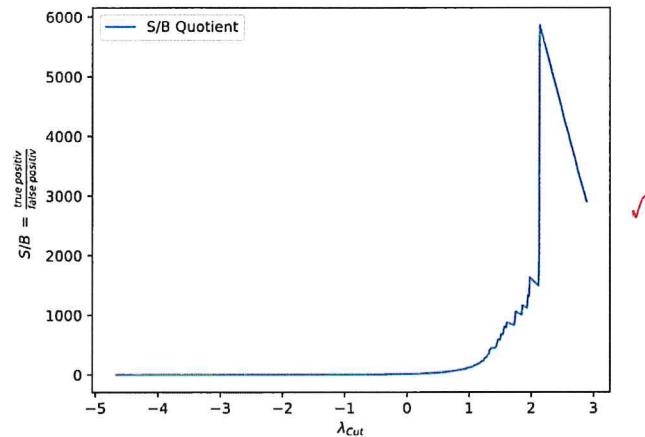


Abbildung 6: Signal-Untergrundverhältnisse für Populationen 0 und 1.

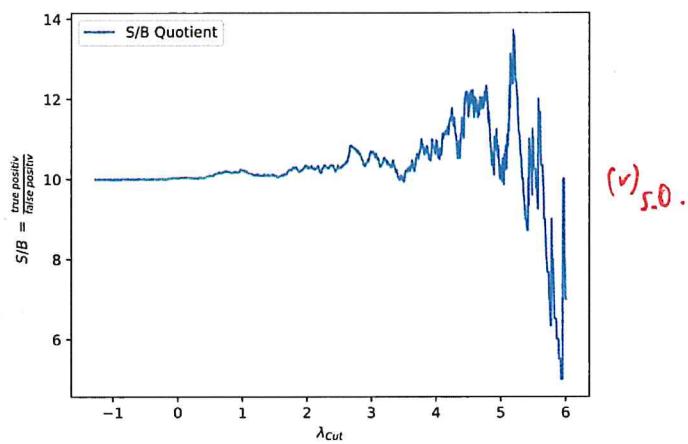


Abbildung 7: Signal-Untergrundverhältnisse für Populationen 0 und 0\_1000.

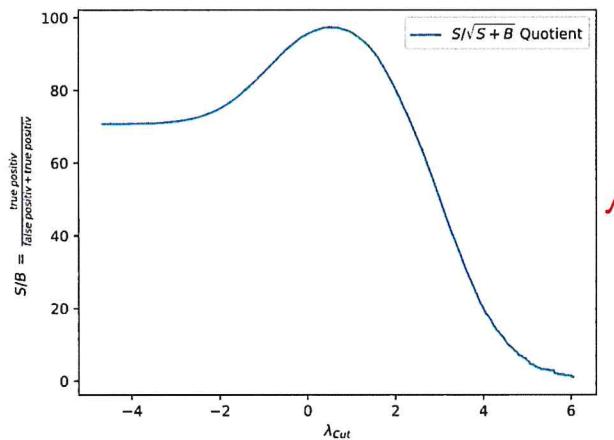


Abbildung 8: Signifikanz nach der Trennung Populationen 0 und 1.

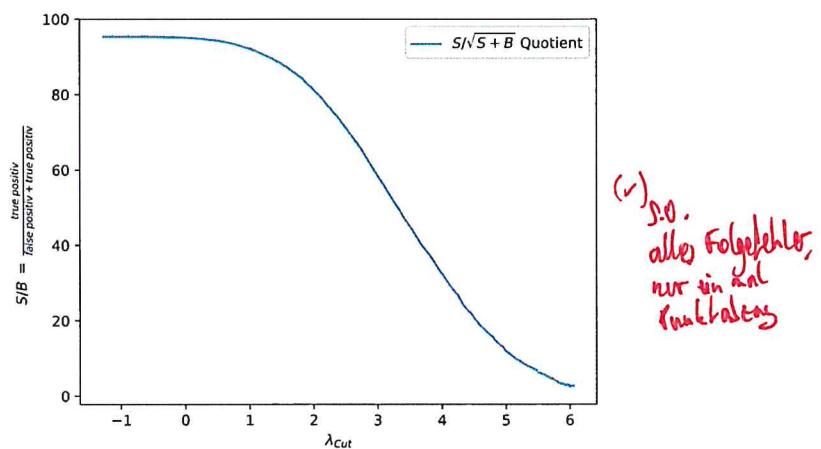


Abbildung 9: Signifikanz nach der Trennung Populationen 0 und 0\_1000.

**d)** siehe oben



# Code fuer Blatt04

Kasper, Appel, Schroeer

16. November 2018

```
..../B/1/Blatt04_Kasper_Appel_Schroeer/Blatt4.py

1 def Aufgabe1():
2     np.random.seed(seed=7)
3
4     # b)
5     # pop sind Populationen, mean Mittelwerte und cov Kovarianzmatrizen
6     mean0 = (0, 3)
7     cov0 = ([12.25, 8.19], [8.19, 6.67])
8     pop0 = np.random.multivariate_normal(mean0, cov0, 10000)
9
10    mean1 = (6, 3.1)
11    cov1 = ([12.25, 7.351323], [7.351323, 5.410276])
12    pop1 = np.random.multivariate_normal(mean1, cov1, 10000)
13
14    plt.scatter(pop0[:, 0], pop0[:, 1], c='r')
15    plt.scatter(pop1[:, 0], pop1[:, 1])
16    plt.xlabel('x')
17    plt.ylabel('y')
18    plt.savefig('scatterplot.png')
19
20    # c)
21    # pop sind Populationen, mu_ Mittelwerte und cova Kovarianzmatrizen
22    mu_0_x = np.mean(pop0[:, 0])
23    mu_0_y = np.mean(pop0[:, 1])
24
25    cova0 = np.cov(pop0[:, 0], pop0[:, 1])
26
27    mu_1_x = np.mean(pop1[:, 0])
28    mu_1_y = np.mean(pop1[:, 1])
29
30    cova1 = np.cov(pop1[:, 0], pop1[:, 1])
31
32    pop_ges_x = np.array([])
33    pop_ges_x = np.append(pop_ges_x, pop0[:, 0])
34    pop_ges_x = np.append(pop_ges_x, pop1[:, 0])
35    mu_ges_x = np.mean(pop_ges_x)
36
37    pop_ges_y = np.array([])
38    pop_ges_y = np.append(pop_ges_y, pop0[:, 1])
39    pop_ges_y = np.append(pop_ges_y, pop1[:, 1])
40    mu_ges_y = np.mean(pop_ges_y)
41
42    pop_ges = np.array([pop_ges_x, pop_ges_y])
43    cova_ges = np.cov(pop_ges_x, pop_ges_y)
44
45    # Essensausgabe:
46    print('Population 0 x : ')
47    print(mu_0_x)
48    print('Population 0 y : ')
49    print(mu_0_y)
50    print('Kovarianzmatrix 0 :')
51    print(cova0)
52    print()
53    print('Population 1 x : ')
54    print(mu_1_x)
55    print('Population 1 y : ')
56    print(mu_1_y)
57    print('Kovarianzmatrix 1 :')
```

```

58     print(coval)
59     print()
60     print('Population Gesamt x : ')
61     print(mu_ges_x)
62     print('Population Gesamt y : ')
63     print(mu_ges_y)
64     print('Kovarianzmatrix Gesamt :')
65     print(cova_ges)
66
67
68 def Aufgabe2():
69     ##Einlesen der Daten ##
70     Pzero = pd.read_hdf('zwei_populationen.h5',key='P_0_10000')
71     Pone = pd.read_hdf('zwei_populationen.h5',key='P_1')
72     P01 = pd.read_hdf('zwei_populationen.h5',key='P_0_1000')
73     p01x = np.array(P01['x'])
74     p01y = np.array(P01['y'])
75     p1x = np.array(Pone['x'])
76     p1y = np.array(Pone['y'])
77     p0x = np.array(Pzero['x'])
78     p0y = np.array(Pzero['y'])
79     ##Berechnung der Mittelwertvektoren##
80     mup0 = np.array([np.mean(p0x),np.mean(p0y)])
81     mup1 = np.array([np.mean(p1x),np.mean(p1y)])
82     mup01 = np.array([np.mean(p01x),np.mean(p01y)])
83     ##Berechnung der Kovarianzmatrizen##
84     Vp0 = np.cov(p0x,p0y)
85     Vp1 = np.cov(p1x,p1y)
86     Vp01 = np.cov(np.append(p0x,p1x),np.append(p0y,p1y))
87     Vp2 = np.cov(p01x,p01y)
88     Vp001 = np.cov(np.append(p0x,p01x),np.append(p0y,p01y))
89     print('Kovarianzmatrix p0:',Vp0)
90     print('Kovarianzmatrix p1:',Vp1)
91     print('Kovarianzmatrix p0 + p1:',Vp01)
92     print('Kovarianzmatrix p01:',Vp2)
93     print('Kovarianzmatrix p0 + p01:',Vp001)
94     ##Fisher Diskriminanzanalyse##
95     p0 = np.array([p0x,p0y]) #shape 2,1000
96     p1 = np.array([p1x,p1y])
97     p01 = np.array([p01x,p01y])
98     #Berechnung der Streuung der Klassen
99     S0 = np.array([p - mup0 for p in p0.T]).T
100    S1 = np.array([p - mup1 for p in p1.T]).T
101    S01 = np.array([p - mup01 for p in p01.T]).T
102    S0 = [np.matrix([s[0],s[1]]).T*(np.matrix([s[0],s[1]])) for s in S0.T]
103    S0 = reduce(lambda x,y: x+y, S0)
104    S1 = [np.matrix([s[0],s[1]]).T*(np.matrix([s[0],s[1]])) for s in S1.T]
105    S1 = reduce(lambda x,y: x+y, S1)
106    S01 = [np.matrix([s[0],s[1]]).T*(np.matrix([s[0],s[1]])) for s in S01.T]
107    S01 = reduce(lambda x,y: x+y, S01)
108    SW = S0 + S1 #Berechnung der Gesamtstreuung = Within class scatter matrix
109    SW001 = S0 + S01
110
111    lamb = SW.I*np.matrix([[mup0[0]-mup1[0]],[mup0[1]-mup1[1]]]) #Berechnung der Projektion
112    lamb = lamb.T
113    lambnorm = np.sqrt(np.dot(lamb,lamb.T)) #lambda normierung
114    lamb/= lambnorm #Müsste jetzt Einheitsvektor sein
115    print('Normiertes Lambda p0:',lamb)
116    print(lambnorm)
117    lamb001 = SW.I*np.matrix([[mup0[0]-mup01[0]],[mup0[1]-mup01[1]]]) #Berechnung der
118    lamb001 = lamb001.T
119    lambnorm001 = np.sqrt(np.dot(lamb001,lamb001.T)) #lambda normierung
120    lamb001/= lambnorm001 #Müsste jetzt Einheitsvektor sein
121    print('Normiertes Lambda p01:',lamb001)
122    print(lambnorm001)
123
124    ##1dim Hist der Populationen##
125    def projektion(lam,data):
126        return np.squeeze(np.asarray(np.dot(lam,data)/np.dot(lam, lam.T)))
127    proj0 = projektion(lamb,p0)

```

```

128 proj1 = projektion(lamb,p1)
129 proj01 = projektion(lamb,p01)
130
131 plt.hist(proj0,bins='auto', label='Population 0',histtype = 'step')
132 plt.hist(proj1,bins='auto', label='Population 1',histtype = 'step')
133 plt.xlabel(r'$\lambda \cdot x$')
134 plt.legend(loc='best')
135 plt.savefig('Projektion1dimhist.pdf')
136 plt.clf()
137 plt.hist(proj0,bins='auto', label='Population 0',histtype = 'step')
138 plt.hist(proj01,bins='auto', label='Population 0_1000',histtype = 'step')
139 plt.xlabel(r'$\lambda \cdot x$')
140 plt.legend(loc='best')
141 plt.savefig('2Projektion1dimhist.pdf')
142
143 ## Lambda Cut Funktion ##
144 def lambdaCut(lamcut,lam,P0,P1):
145     reinheit = np.array([])
146     effizienz = np.array([])
147     #hier steck ich jetzt rein das ich weiß, dass Population 0 rechts liegt
148     # Population 0 ist signal also "positiv"
149     for l in lamcut:
150         projzero = projektion(lam,P0)
151         projone = projektion(lam,P1)
152         fn = projzero[projzero < 1].size #false negativ
153         tp = projzero[projzero > 1].size # true positiv
154         tn = projone[projone < 1].size #true negativ
155         fp = projone[projone > 1].size #false positiv
156         if (tp + fp) <= 0 : break
157         if (tp + fn) <= 0 : break
158         reinheit = np.append(reinheit,[tp/(tp+fp)])
159         effizienz = np.append(effizienz,[tp/(tp+fn)])
160     return reinheit,effizienz
161
162 ## reinheit und effizienz plot ##
163 lamcut = np.linspace(min(min(proj0),min(proj1)),max(max(proj0),max(proj1)),10**3)
164 lamcut2 = np.linspace(min(min(proj0),min(proj01)),max(max(proj0),max(proj01)),10**3)
165 rein,effi = lambdaCut(lamcut,lamb,p0,p1)
166 rein2,effi2 = lambdaCut(lamcut2,lamb001,p0,p01)
167
168 plt.clf()
169 plt.plot(lamcut[:rein.size],rein, '-r', label='Reinheit')
170 plt.plot(lamcut[:effi.size],effi, '-b', label='Effizienz')
171 plt.xlabel(r'$\lambda_{Cut}$')
172 plt.ylabel(r'Reinheit & Effizienz')
173 plt.legend(loc='best')
174 plt.savefig('ReinheitEffizienzplot.pdf')
175 plt.clf()
176 plt.plot(lamcut2[:rein2.size],rein2, '-r', label='Reinheit')
177 plt.plot(lamcut2[:effi2.size],effi2, '-b', label='Effizienz')
178 plt.xlabel(r'$\lambda_{Cut}$')
179 plt.ylabel(r'Reinheit & Effizienz')
180 plt.legend(loc='best')
181 plt.savefig('2ReinheitEffizienzplot.pdf')
182 #plt.show()
183
184 ## S/B Quotient
185 def SBRatio(lamcut,lam,P0,P1):
186     SB = np.array([])
187     for l in lamcut:
188         projzero = projektion(lam,P0)
189         projone = projektion(lam,P1)
190         tp = projzero[projzero > 1].size # true positiv
191         if projone[projone > 1].size <= 0: break
192         fp = projone[projone > 1].size #false positiv
193         SB = np.append(SB,tp/fp)
194     return SB
195 SBRatio = SBRatio(lamcut, lamb, p0,p1)
196 SBRatio2 = SBRatio(lamcut2, lamb001, p0,p01)
197 placeholder = lamcut[:SBRatio.size]
198 print('max sbr 1:',placeholder[SBRatio==max(SBRatio)])
199 placeholder = lamcut2[:SBRatio2.size]
200 print('max sbr2:',placeholder[SBRatio2==max(SBRatio2)])

```

```

199
200 plt.clf()
201 plt.plot(lamcut[:SBRatio.size],SBRatio, label='S/B Quotient')
202 plt.xlabel(r'$\lambda_{Cut}$')
203 plt.ylabel(r'$S/B \wedge \frac{true}{false} \wedge positiv$ ')
204 plt.legend(loc='best')
205 plt.savefig('SBRatioplot.pdf')
206 plt.clf()
207 plt.plot(lamcut2[:SBRatio2.size],SBRatio2, label='S/B Quotient')
208 plt.xlabel(r'$\lambda_{Cut}$')
209 plt.ylabel(r'$S/B \wedge \frac{true}{false} \wedge positiv$ ')
210 plt.legend(loc='best')
211 plt.savefig('2SBRatioplot.pdf')
212
213 ##Signifikanz
214 def Signiratio(lamcut, lam, P0, P1):
215     Sign = np.array([])
216     for l in lamcut:
217         projzero = projektion(lam,P0)
218         projone = projektion(lam,P1)
219         tp = projzero[projzero > 1].size # true positiv
220         fp = projone[projone > 1].size #false positiv
221         if np.sqrt((tp+fp))<= 0: break
222         Sign = np.append(Sign,tp/np.sqrt((tp+fp)))
223     return Sign
224 Sign = Signiratio(lamcut,lamb,p0,p1)
225 Sign2 = Signiratio(lamcut2,lamb001,p0,p01)
226 placeholder = lamcut[:Sign.size]
227 print('max sign1:',placeholder[Sign==max(Sign)])
228 placeholder = lamcut2[:Sign2.size]
229 print('max sign2:',placeholder[Sign2==max(Sign2)])
230
231 plt.clf()
232 plt.plot(lamcut[:Sign.size],Sign, label=r'$S/\sqrt{S+B}$ Quotient')
233 plt.xlabel(r'$\lambda_{Cut}$')
234 plt.ylabel(r'$S/B \wedge \frac{true}{false} \wedge positiv + true \wedge positiv$ ')
235 plt.legend(loc='best')
236 plt.savefig('Signifikanzplot.pdf')
237 plt.clf()
238 plt.plot(lamcut2[:Sign2.size],Sign2, label=r'$S/\sqrt{S+B}$ Quotient')
239 plt.xlabel(r'$\lambda_{Cut}$')
240 plt.ylabel(r'$S/B \wedge \frac{true}{false} \wedge positiv + true \wedge positiv$ ')
241 plt.legend(loc='best')
242 plt.savefig('2Signifikanzplot.pdf')
243
244
245 if __name__ == '__main__':
246     import matplotlib.pyplot as plt
247     import numpy as np
248     import pandas as pd
249     from functools import reduce
250
251     Aufgabe1()
252     Aufgabe2()

```

# SMD Blatt 4 Aufgabe 1

a)  $x \sim G(\mu_x, \sigma_x)$

Die Gewächse

Robert Appel  
Julian Schröer  
Yvonne Wasper

$G \hat{=} \text{Gaus-verteilung}$

$$y \sim G(\mu_{y|x}, \sigma_{y|x})$$

nach  
Aufgabenstellung

$$G_1(y) = \frac{1}{\sqrt{2\pi} \sigma_{y|x}} \exp \left( -\frac{1}{2} \underbrace{\left( \frac{y - \mu_{y|x}}{\sigma_{y|x}} \right)^2}_{:= II} \right)$$

Index  
für  
Population

$$G_1(y|x) = \frac{1}{\sqrt{2\pi} \sigma_y \sqrt{1-s^2}} \exp \left( -\frac{1}{2} \underbrace{\frac{1}{1-s^2} \left[ \frac{y - \mu_y}{\sigma_y} - s \frac{x - \mu_x}{\sigma_x} \right]^2}_{:= I} \right)$$

aus Aufgabe

$$\text{mit } \tilde{y} = y - \mu_y \quad \tilde{x} \text{ analog}$$

Koeff.  
Vgl.

$$\sqrt{I} = \underbrace{\frac{\mu_y}{\sqrt{1-s^2} \sigma_y}}_{\textcircled{1}} - \underbrace{\frac{\mu_x}{\sqrt{1-s^2} \sigma_x}}_{\textcircled{2}} + \underbrace{\frac{s \mu_x}{\sqrt{1-s^2} \sigma_x}}_{\textcircled{3}} - \underbrace{\frac{s}{\sqrt{1-s^2} \sigma_x} \cdot x}_{\textcircled{4}}$$

$$\sigma_x \cdot \sqrt{1-s^2} = \sigma_{y|x}$$

$$\Rightarrow \textcircled{1} = \frac{y}{\sigma_{y|x}} \quad \textcircled{2} + \textcircled{3} = \frac{a}{\sigma_{y|x}} \quad \textcircled{4} = \frac{b}{\sigma_{y|x}} \cdot x$$

$$\sigma_{y|x} = \sqrt{1-s^2} \sigma_y \quad \text{mit } \mu_{y|x} = a + b \cdot x \text{ in } \textcircled{II}$$

$$\Rightarrow b = \frac{s \sigma_y}{\sigma_x} \quad \text{und} \quad \frac{a}{\sigma_{y|x}} = \frac{\sigma_x \mu_y - s \sigma_y \mu_x}{\sqrt{1-s^2} \sigma_x \sigma_y}$$

$$\Rightarrow b \sigma_x = s \sigma_y \quad \Leftrightarrow a = \mu_y - s \frac{\sigma_y}{\sigma_x} \mu_x$$

$$\Rightarrow \boxed{\mu_y = a + \mu_x \frac{b \sigma_x}{\sigma_x} = a + \mu_x b} \quad \begin{array}{l} \text{Formel} \\ \text{für } \mu_y \end{array}$$

$$b = \frac{s \sigma_{y|x}}{\sqrt{1-s^2}} \quad \frac{1}{\sigma_x} \quad (\text{einsetzen } \sigma_y \dots)$$

$$\Leftrightarrow \frac{s}{\sqrt{1-s^2}} = \frac{\sigma_x b}{\sigma_{x|y}} =: A$$

$$A^2 = \frac{s^2}{1-s^2} = \frac{1}{\frac{1}{s^2}-1}$$

$$\Leftrightarrow \cancel{\frac{1}{s^2}} = 1 + \frac{1}{A^2}$$

$$s^2 = \sqrt{\frac{1}{1+\frac{1}{A^2}}} = \frac{A}{\sqrt{1+A^2}} // \quad \begin{matrix} \text{Formel} \\ \text{für } s \end{matrix}$$

$$\sigma_y = \frac{\sigma_{y|x}}{\sqrt{1-s^2}} \quad \begin{matrix} \text{Formel für } \sigma_y \end{matrix}$$

$$A = \frac{\sigma_x \cdot b}{\sigma_{x|y}} = 2 \cdot 1 // \quad s = \frac{A}{\sqrt{1+A^2}} \approx 0.903 //$$

$$\mu_y = a + \mu_x b = 3 \cdot 1 // \quad \sigma_y = \frac{\sigma_{y|x}}{\sqrt{1-s^2}} \approx 2.326 //$$

2P

## Aufgabe 2

$$P_0 : \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 1,5 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ 3 \end{pmatrix}, \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

$$P_1 : \begin{pmatrix} 1,5 \\ 1 \end{pmatrix}, \begin{pmatrix} 2,5 \\ 1 \end{pmatrix}, \begin{pmatrix} 3,5 \\ 1 \end{pmatrix}, \begin{pmatrix} 2,5 \\ 2 \end{pmatrix}, \begin{pmatrix} 3,5 \\ 2 \end{pmatrix}, \begin{pmatrix} 4,5 \\ 2 \end{pmatrix}$$

a)  $\vec{\mu}_i = \frac{1}{N} \left( \sum x_i \right)$

$$\vec{\mu}_0 = \frac{1}{6} \left( \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 2 \\ 1 \end{pmatrix} + \begin{pmatrix} 1,5 \\ 2 \end{pmatrix} + \begin{pmatrix} 2 \\ 2 \end{pmatrix} + \begin{pmatrix} 2 \\ 3 \end{pmatrix} + \begin{pmatrix} 3 \\ 3 \end{pmatrix} \right) = \frac{1}{6} \begin{pmatrix} 11,5 \\ 12 \end{pmatrix} = \begin{pmatrix} 23/12 \\ 2 \end{pmatrix} \checkmark$$

$$\vec{\mu}_1 = \frac{1}{6} \left( \begin{pmatrix} 1,5 \\ 1 \end{pmatrix} + \begin{pmatrix} 2,5 \\ 1 \end{pmatrix} + \begin{pmatrix} 3,5 \\ 1 \end{pmatrix} + \begin{pmatrix} 2,5 \\ 2 \end{pmatrix} + \begin{pmatrix} 3,5 \\ 2 \end{pmatrix} + \begin{pmatrix} 4,5 \\ 2 \end{pmatrix} \right) = \frac{1}{6} \begin{pmatrix} 18 \\ 9 \end{pmatrix} = \begin{pmatrix} 3 \\ 3/2 \end{pmatrix} \checkmark$$

$$S_i = \sum_{j=1}^n (\vec{x}_i - \vec{\mu}_j)(\vec{x}_i - \vec{\mu}_j)^T$$

$$\begin{aligned} S_0 &= \begin{pmatrix} \frac{11}{2} \\ -1 \end{pmatrix} \left( -\frac{11}{12} \quad -1 \right) + \begin{pmatrix} \frac{1}{12} \\ -1 \end{pmatrix} \left( \frac{1}{12} \quad -1 \right) + \begin{pmatrix} -\frac{5}{12} \\ 0 \end{pmatrix} \left( -\frac{5}{12} \quad 0 \right) \\ &\quad + \begin{pmatrix} \frac{1}{12} \\ 0 \end{pmatrix} \left( \frac{1}{12} \quad 0 \right) + \begin{pmatrix} \frac{1}{12} \\ 1 \end{pmatrix} \left( \frac{1}{12} \quad 1 \right) + \begin{pmatrix} \frac{13}{12} \\ 1 \end{pmatrix} \left( \frac{13}{12} \quad 1 \right) \\ &= \begin{pmatrix} \frac{121}{144} & \frac{11}{12} \\ \frac{11}{12} & 1 \end{pmatrix} + \begin{pmatrix} \frac{1}{144} & -\frac{1}{12} \\ -\frac{1}{12} & 1 \end{pmatrix} + \begin{pmatrix} \frac{25}{144} & 0 \\ 0 & 0 \end{pmatrix} \\ &\quad + \begin{pmatrix} \frac{1}{144} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \frac{1}{144} & \frac{1}{12} \\ \frac{1}{12} & 1 \end{pmatrix} + \begin{pmatrix} \frac{169}{144} & \frac{13}{12} \\ \frac{13}{12} & 1 \end{pmatrix} \\ &= \begin{pmatrix} \frac{53}{24} & 2 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} 2,2083 & 2 \\ 2 & 4 \end{pmatrix} \checkmark \end{aligned}$$

$$\begin{aligned} S_1 &= \begin{pmatrix} -\frac{3}{2} \\ -\frac{1}{2} \end{pmatrix} \left( -\frac{3}{2} \quad -\frac{1}{2} \right) + \begin{pmatrix} -\frac{1}{2} \\ -\frac{1}{2} \end{pmatrix} \left( -\frac{1}{2} \quad -\frac{1}{2} \right) + \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \end{pmatrix} \left( \frac{1}{2} \quad -\frac{1}{2} \right) \\ &\quad + \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \left( -\frac{1}{2} \quad \frac{1}{2} \right) + \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \left( \frac{1}{2} \quad \frac{1}{2} \right) + \begin{pmatrix} \frac{3}{2} \\ \frac{1}{2} \end{pmatrix} \left( \frac{3}{2} \quad \frac{1}{2} \right) \\ &= \begin{pmatrix} \frac{9}{4} & \frac{2}{4} \\ \frac{3}{4} & \frac{1}{4} \end{pmatrix} + \begin{pmatrix} \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} \end{pmatrix} + \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} \end{pmatrix} + \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} \end{pmatrix} + \dots \end{aligned}$$

$$\dots + \begin{pmatrix} \frac{1}{6} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \end{pmatrix} + \begin{pmatrix} \frac{3}{4} & \frac{3}{4} \\ \frac{3}{4} & \frac{1}{4} \end{pmatrix} = \begin{pmatrix} \frac{22}{6} & \frac{6}{4} \\ \frac{6}{4} & \frac{6}{4} \end{pmatrix} = \begin{pmatrix} \frac{11}{2} & \frac{3}{2} \\ \frac{3}{2} & \frac{3}{2} \end{pmatrix}$$

$$= \begin{pmatrix} 5,5 & 1,5 \\ 1,5 & 1,5 \end{pmatrix} \checkmark$$

$$S_W = S_0 + S_1$$

$$= \begin{pmatrix} \frac{53}{24} & 2 \\ 2 & 4 \end{pmatrix} + \begin{pmatrix} \frac{11}{2} & \frac{3}{2} \\ \frac{3}{2} & \frac{3}{2} \end{pmatrix} = \begin{pmatrix} \frac{185}{24} & \frac{7}{2} \\ \frac{7}{2} & \frac{11}{2} \end{pmatrix}$$

$$= \begin{pmatrix} 7,708\bar{3} & 3,5 \\ 3,5 & 5,5 \end{pmatrix} \checkmark$$

b)  $\vec{\lambda} = S_W^{-1} (\vec{\mu}_0 - \vec{\mu}_1)$

mit  $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \Rightarrow A^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$

$$S_W^{-1} = \frac{1}{\frac{185}{24} \cdot \frac{11}{2} - \frac{7}{2} \cdot \frac{7}{2}} \begin{pmatrix} \frac{11}{2} & -\frac{7}{2} \\ -\frac{7}{2} & \frac{185}{24} \end{pmatrix}$$

$$= \frac{48}{1447} \begin{pmatrix} \frac{11}{2} & -\frac{7}{2} \\ -\frac{7}{2} & \frac{185}{24} \end{pmatrix}$$

$$\vec{\lambda} = \frac{48}{1447} \begin{pmatrix} \frac{11}{2} & -\frac{7}{2} \\ -\frac{7}{2} & \frac{185}{24} \end{pmatrix} \begin{pmatrix} -\frac{13}{12} \\ \frac{1}{2} \end{pmatrix}$$

$$= \frac{48}{1447} \begin{pmatrix} -\frac{143}{24} - \frac{7}{4} \\ \frac{91}{24} + \frac{185}{48} \end{pmatrix} = \frac{48}{1447} \begin{pmatrix} -\frac{185}{24} \\ \frac{367}{48} \end{pmatrix}$$

$$= \frac{1}{1447} \begin{pmatrix} -370 \\ 367 \end{pmatrix}$$

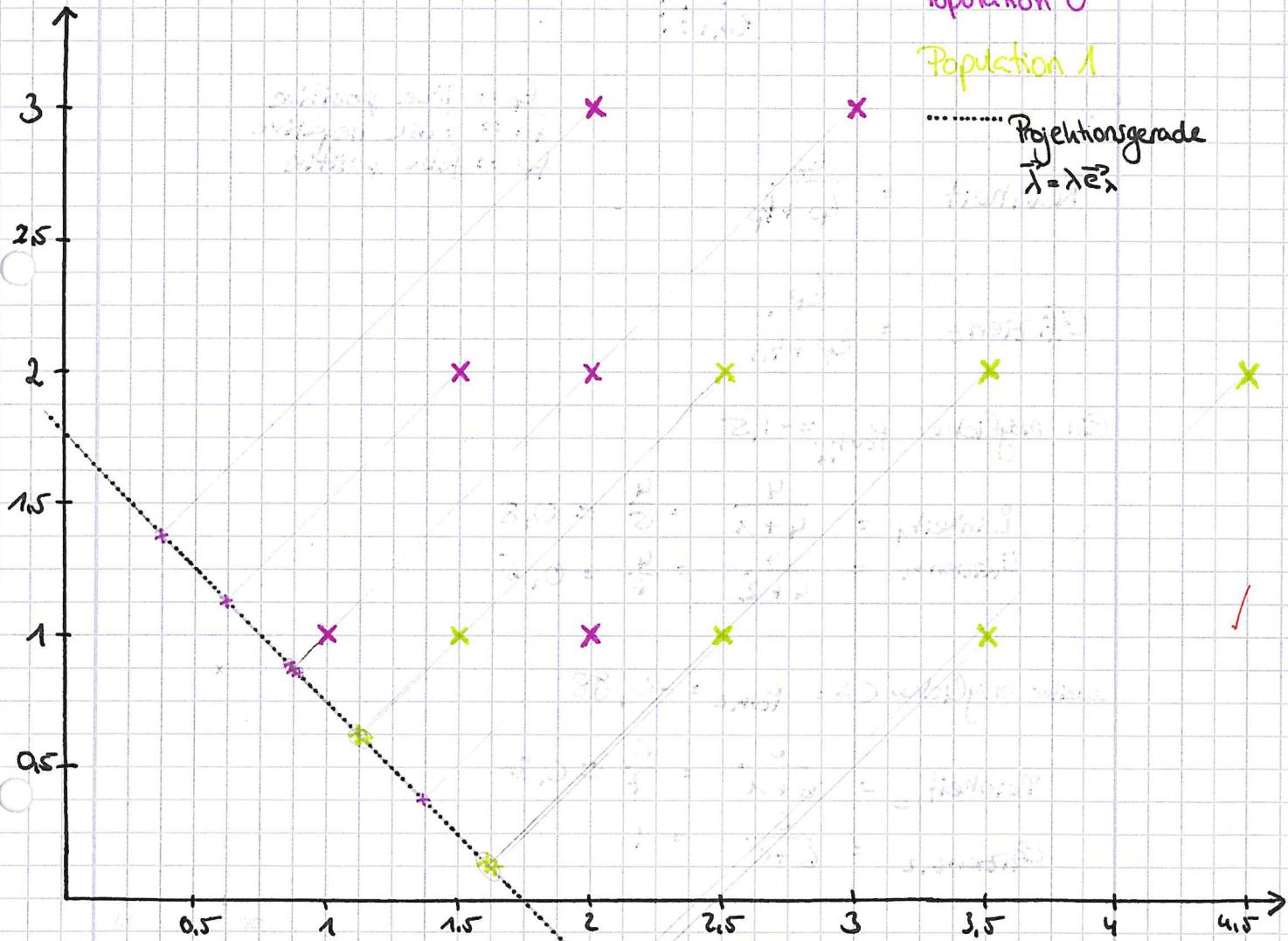
c)  $|\vec{\lambda}| = \sqrt{\left(\frac{-370}{1447}\right)^2 + \left(\frac{367}{1447}\right)^2}$

$$= \sqrt{\left(\frac{136900}{1447^2}\right) + \left(\frac{134689}{1447^2}\right)} = \dots$$

$$\dots = \sqrt{\frac{271589}{1447^2}} = \sqrt{\frac{271589}{1447}} = |\vec{\lambda}|$$

$$\vec{e}_\lambda = \frac{1}{\sqrt{271589}} \begin{pmatrix} -370 \\ 367 \end{pmatrix} \approx \begin{pmatrix} -0,70998 \\ 0,70422 \end{pmatrix}$$

c)



d)

$$\text{Projektion } P_i = \frac{\vec{x}_i \cdot \vec{\lambda}}{\vec{\lambda} \cdot \vec{\lambda}}$$

Sorry für das ganze  
TipEx... hatte mich  
erst voll verrechnet ...

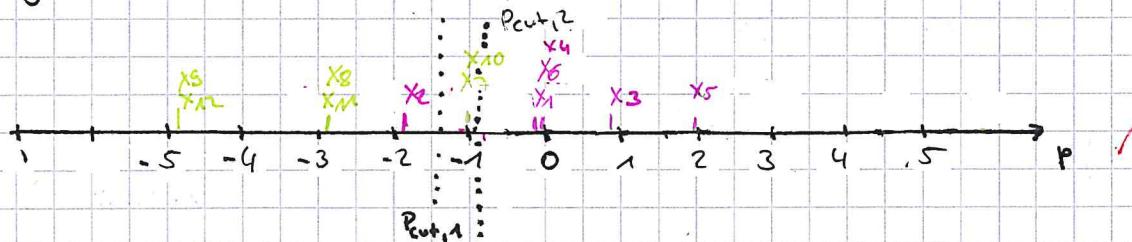
$$P_1: P_1 = \frac{1447^2}{271589} \cdot \frac{1}{1447} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} -370 \\ 367 \end{pmatrix} \approx -0,016$$

$$P_2 \approx -1,99, \quad P_3 \approx 0,95, \quad P_4 \approx -0,03, \quad P_5 \approx 1,92$$

$$P_6 \approx -0,05$$

$$P_1: P_4 \approx -1,00, P_8 \approx -2,97, P_9 \approx -4,94 \\ P_{10} \approx -1,02, P_{11} \approx -2,99, P_{12} \approx -4,96$$

Projektion auf eine Dimension:



e)

$$\text{Reinheit} = \frac{tp}{tp + fp}$$

$$\text{Effizienz} = \frac{tp}{tp + fn}$$

Ein möglicher  $P_{\text{cut},1} = -1,5$

$$\text{Reinheit}_1 = \frac{4}{4+1} = \frac{4}{5} = 0,8$$

$$\text{Effizienz}_1 = \frac{4}{4+2} = \frac{4}{6} = 0,6$$

Zweiter möglicher Cut:  $P_{\text{cut},2} = -0,9$

$$\text{Reinheit}_{1,2} = \frac{6}{6+1} = \frac{6}{7} \approx 0,86$$

$$\text{Effizienz}_{1,2} = \frac{6}{6+0} = 1$$

$tp := \text{True positive}$   
 $fn := \text{false negative}$   
 $fp := \text{false positive}$

Die bessere Wahl ist  $P_{\text{cut}} = -0,9$ , da Effizienz und Reinheit größer sind.

5/5