Sheeeeesh 6

Dienstag, 31. Mai 2022 19:16

12)

a) $\vec{\mu}_0 = \begin{pmatrix} \bar{X}_0 \\ \bar{Y}_0 \end{pmatrix} = \frac{1}{N_0} \begin{pmatrix} \sum_i x_{0,i} \\ \sum_i y_{0,i} \end{pmatrix}$

Population 0

$$\left.\begin{aligned} \sum_i x_0 &= 11{,}5 \\ \sum_i y_0 &= 12 \end{aligned}\right\} \overset{\text{with } N_0 = 6}{\Rightarrow} \underline{\underline{\vec{\mu}_0 = \begin{pmatrix} \frac{23}{12} \\ 2 \end{pmatrix}}}$$

$$\vec{\mu}_1 = \frac{1}{N_1} \begin{pmatrix} \sum_i x_{1,i} \\ \sum_i y_{1,i} \end{pmatrix}$$

$$\left.\begin{aligned} \sum_i x_{1,i} &= 18 \\ \sum_i y_{1,i} &= 9 \end{aligned}\right\} \overset{\text{with } N_0 = N_1}{\Rightarrow} \underline{\underline{\vec{\mu}_1 = \begin{pmatrix} 3 \\ \frac{3}{2} \end{pmatrix}}}$$

Scatter Matrix:

$$S_B = (\vec{\mu}_0 - \vec{\mu}_1)(\vec{\mu}_0 - \vec{\mu}_1)^T$$

$$= \left(\begin{pmatrix} \frac{23}{12} \\ 2 \end{pmatrix} - \begin{pmatrix} 3 \\ \frac{3}{2} \end{pmatrix}\right) \cdot (\vec{\mu}_0 - \vec{\mu}_1)^T$$

$$= \begin{pmatrix} -\frac{13}{12} \\ \frac{1}{2} \end{pmatrix} \cdot \left(-\frac{13}{12}, \frac{1}{2}\right) = \underline{\underline{\begin{pmatrix} \frac{169}{144} & -\frac{13}{24} \\ -\frac{13}{24} & \frac{1}{4} \end{pmatrix}}}$$

$$S_W = \sum_{j=0}^{1} S_j$$

$$S_j = \sum_{i=1}^{6} (\vec{x}_i - \vec{\mu}_j)(\vec{x}_i - \vec{\mu}_j)^T$$

$$\Rightarrow S_0 = \sum_{i=1}^{6} (\vec{x}_{0i} - \vec{\mu}_0)(\vec{x}_{0i} - \vec{\mu}_0)^T$$

$$= \left(\begin{pmatrix} 1 \\ 1 \end{pmatrix} - \begin{pmatrix} \frac{23}{12} \\ 2 \end{pmatrix}\right)(\vec{x}_1 - \vec{\mu}_0)^T = \begin{pmatrix} \frac{121}{144} & \frac{11}{12} \\ \frac{11}{12} & 1 \end{pmatrix} + \begin{pmatrix} \frac{1}{144} & -\frac{1}{12} \\ -\frac{1}{12} & 1 \end{pmatrix} + \begin{pmatrix} \frac{25}{144} & 0 \\ 0 & 0 \end{pmatrix}$$

$$+ \left(\begin{pmatrix} 2 \\ 1 \end{pmatrix} - \begin{pmatrix} \frac{23}{12} \\ 2 \end{pmatrix}\right)(\vec{x}_2 - \vec{\mu}_0)^T$$

$$+ \ldots \qquad + \begin{pmatrix} \frac{1}{144} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \frac{1}{144} & \frac{1}{12} \\ \frac{1}{12} & 1 \end{pmatrix} + \begin{pmatrix} \frac{169}{144} & \frac{13}{12} \\ \frac{13}{12} & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{53}{24} & 2 \\ 2 & 4 \end{pmatrix}$$

$$\Rightarrow S_1 = \sum_{i=1}^{6} (\vec{x}_{1,i} - \vec{\mu}_1)(\vec{x}_{1,i} - \vec{\mu}_1)^T = \begin{pmatrix} \frac{11}{2} & \frac{3}{2} \\ \frac{3}{2} & \frac{3}{2} \end{pmatrix}$$

$$\Rightarrow S_w = S_0 + S_1 = \underline{\begin{pmatrix} \frac{105}{24} & \frac{7}{2} \\ \frac{7}{2} & \frac{11}{2} \end{pmatrix}}$$

S_{ij} ??

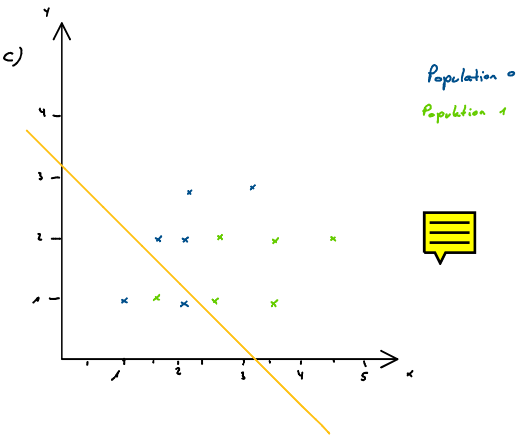
b)

$$\vec{\alpha}^* = S_w^{-1}(\vec{\mu}_1 - \vec{\mu}_2)$$

$$S_w^{-1} = \underbrace{\frac{1}{\det(S_w)}}_{= \frac{1447}{48}} \begin{pmatrix} \frac{11}{2} & -\frac{7}{2} \\ -\frac{7}{2} & \frac{105}{24} \end{pmatrix} = \begin{pmatrix} \frac{15917}{96} & -\frac{10129}{96} \\ -\frac{10129}{96} & \frac{267695}{1152} \end{pmatrix}$$

$$\Rightarrow \vec{\alpha}^* = S_w^{-1} \begin{pmatrix} -\frac{13}{12} \\ \frac{1}{2} \end{pmatrix} = \begin{pmatrix} -370 \\ 367 \end{pmatrix} \cdot \frac{1}{1447}$$

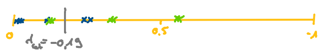
c)



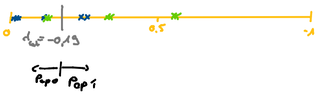
d)

$$x'_{i,j} = \vec{\alpha}^* \vec{x}_{i,j}$$

$x_{0,1} = -0{,}002$	$x_{1,1} = -0{,}13$
$x_{0,2} = -0{,}258$	$x_{1,2} = -0{,}316$
$x_{0,3} = 0{,}124$	$x_{1,3} = -0{,}441$
$x_{0,4} = -0{,}004$	$x_{1,4} = -0{,}132$
$x_{0,5} = 0{,}249$	$x_{1,5} = -0{,}388$
$x_{0,6} = -0{,}006$	$x_{1,6} = -0{,}643$



$\cdot\ x_{0,6} = -0{,}006 \quad ; \quad x_{1,6} = -0{,}643$



e)

d_{cut} has been choosen, so that the error for both sides are equal.
Obviously the way d_{cut} would normally be chosen would totaly depend on the context of the groups, since sometimes you would want a lower error rate on one group than the other e.g. corona tests.

$$\text{Efficiency} = \frac{t_p}{t_p + f_n} \quad ; \quad \text{Reinheit} = \frac{t_p}{t_p + f_p}$$

$$= \frac{4}{4+2} = \frac{4}{6} = \frac{2}{3} \qquad\qquad = \frac{4}{4+2} = \frac{2}{3}$$

Sheet06

May 31, 2022

Exercise 13 a)

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt

     %matplotlib widget

     p0 = pd. read_hdf ('two_populations.h5', key='P_0_10000')
     p1 = pd. read_hdf ('two_populations.h5', key='P_1')

     p0_1000 = pd. read_hdf ('two_populations.h5', key='P_0_1000')

     mu_p0 = np.array([np.mean(p0['x']), np.mean(p0['y'])])
     mu_p1 = np.array([np.mean(p1['x']), np.mean(p1['y'])])

     print('mu_p0 ',mu_p0)
     print('mu_p1 ',mu_p1)
```

```
mu_p0  [-0.00729968  2.96367644]
mu_p1  [6.0962719  3.17467385]
```

b) covariance V__p0 and V__p1

```
[2]: p0_np = np.array(p0)
     p1_np = np.array(p1)



     V_p0 = np.dot((p0_np-mu_p0).reshape(2, 10000), (p0_np-mu_p0))
     V_p1 = np.dot((p1_np-mu_p1).reshape(2, 10000), (p1_np-mu_p1))

     S_W = V_p0 + V_p1

     print(S_W)
```

```
[[1126.02785845  845.49698256]
 [-159.09265778 -614.57006974]]
```

c) Linear fisher Discriminat

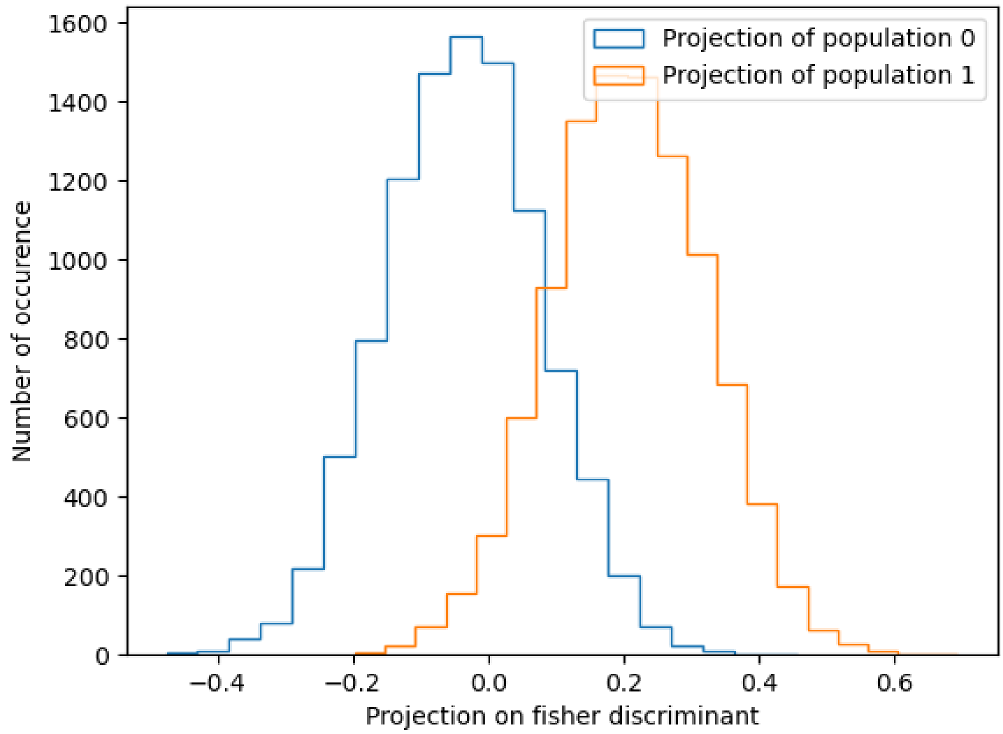
$$S_W^{-1} S_B \cdot \vec{v} = D \cdot \vec{v}$$

[3]:
```
S_B = np.dot((mu_p0-mu_p1).reshape(2,1), (mu_p0-mu_p1).reshape(1,2))

S_W_inv = np.linalg.inv(S_W)

A = np.dot(S_W_inv, S_B)

w , v= np.linalg.eig(A)

#want biggest eigenvalue, thus take the first one

print('eigenvalues', w)
print('eigenvectors', v)

eigen_value = w[0]
eigen_vector = v[:,0]

print('Lambda', eigen_value)
print('Linear Fisher Discriminat', eigen_vector)
```

```
eigenvalues [0.04256203 0.        ]
eigenvectors [[ 0.95580945 -0.03454886]
 [-0.29398689  0.99940301]]
Lambda 0.04256203070888654
Linear Fisher Discriminat [ 0.95580945 -0.29398689]
```

d) projection

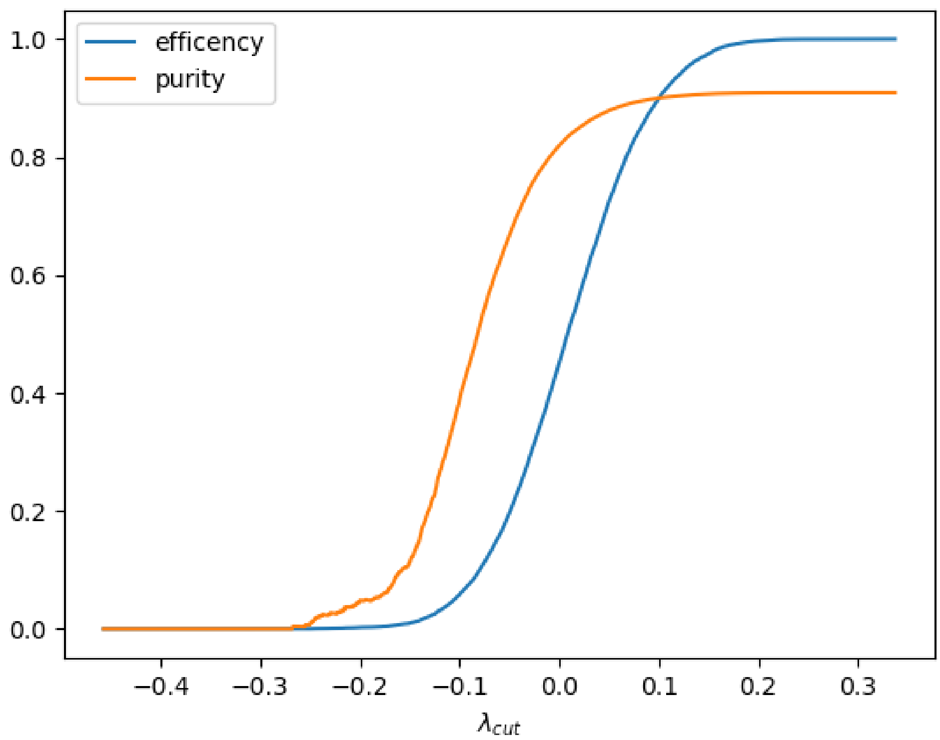
[4]:
```
pop_0_proj = eigen_value * np.dot(p0_np, eigen_vector)
pop_1_proj = eigen_value * np.dot(p1_np, eigen_vector)

plt.figure()
plt.hist(pop_0_proj, bins = 20, label = 'Projection of population 0',␣
 ↪histtype='step')
plt.hist(pop_1_proj, bins = 20, label = 'Projection of population 1',␣
 ↪histtype='step')
plt.xlabel('Projection on fisher discriminant')
plt.ylabel('Number of occurence')
plt.legend()

None
```



e) efficiency and purity

[10]:
```
def purity(lambda_cut,pop_0, pop_1):
    purity = len(pop_1[pop_1 <= lambda_cut])/(len(pop_1[pop_1 <=␣
 ↪lambda_cut])+len(pop_0[pop_0 <= lambda_cut]))
    return purity

def eff(lambda_cut, pop_0, pop_1):
    efficiency = len(pop_1[pop_1 <= lambda_cut])/(len(pop_1[pop_1 <=␣
 ↪lambda_cut])+ len(pop_1[pop_1 > lambda_cut]))
    return efficiency

cuts_array = np.linspace(min(pop_0_proj), max(pop_1_proj), 1000)

eff_array = np.array([eff(cuts_array[i],pop_0_proj, pop_1_proj)  for i in␣
 ↪range(len(cuts_array))])
purity_array = np.array([purity(cuts_array[i],pop_0_proj,pop_1_proj)  for i in␣
 ↪range(len(cuts_array))])


plt.figure()
```

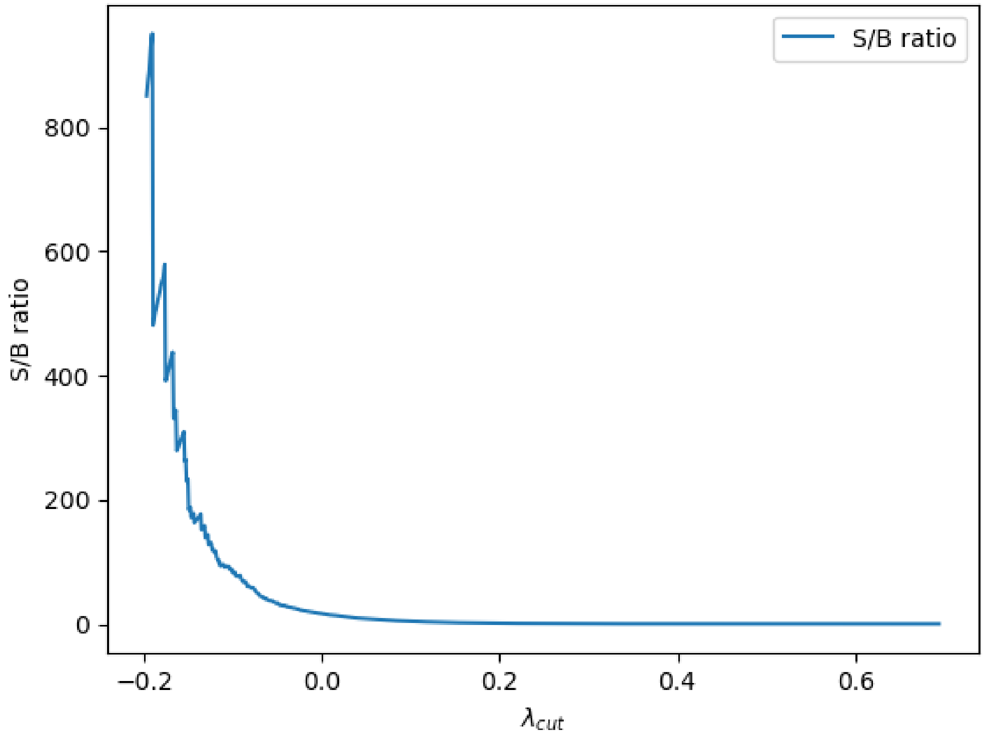
```
plt.plot(cuts_array, eff_array, label='efficency')
plt.plot(cuts_array, purity_array, label='purity')
plt.xlabel(r'$\lambda_{cut}$')
plt.ylabel('')
plt.legend()

None
```



f) signal to background ratio

[5]:
```
def s_b_ratio(cut, pop_0, pop_1):
    return len(pop_0[pop_0 < cut])/len(pop_1[pop_1 <= cut])

cuts_array = np.linspace(min(pop_1_proj), max(pop_1_proj), 1000)

signal_background = [s_b_ratio(cuts_array[i], pop_0_proj, pop_1_proj) for i in␣
 ↪range(len(cuts_array)-1)]

plt.figure()
plt.xlabel(r'$\lambda_{cut}$')
plt.ylabel('S/B ratio')
```

```
plt.plot(cuts_array[:-1], signal_background, label = 'S/B ratio')
plt.legend()
None
```



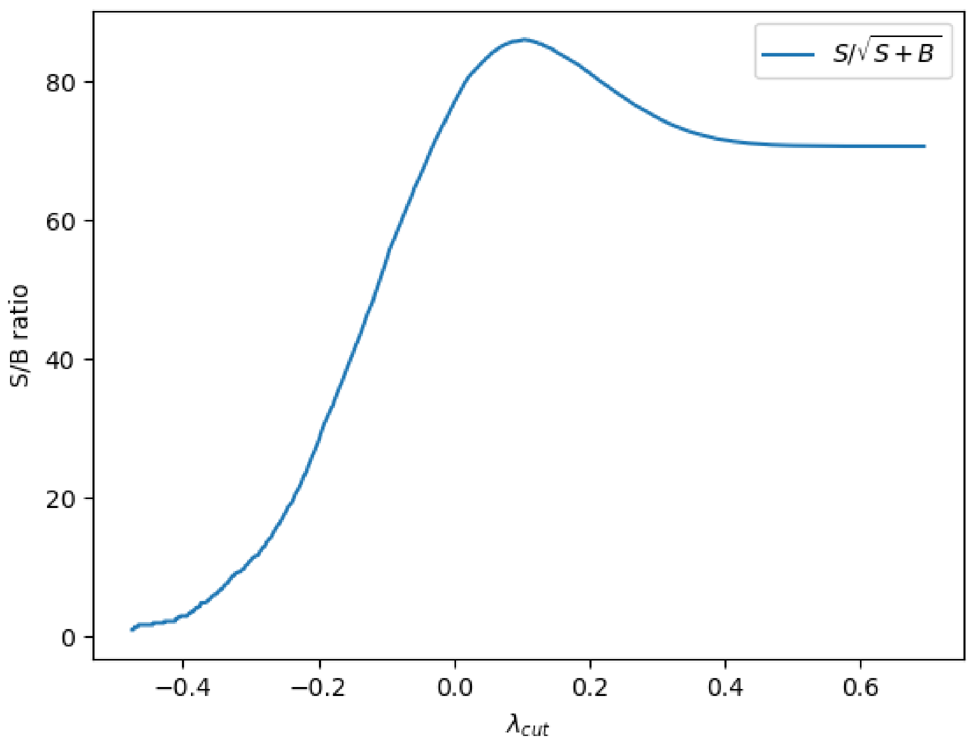
The ratio has its maximum when there is no background, however we cant plot it. If there is background signal, we find the maximum around -0.2.

g)

```
[6]: def s_b_sqrt(cut, pop_0, pop_1):
         return len(pop_0[pop_0 < cut])/np.sqrt(len(pop_1[pop_1 <
      ↪cut])+len(pop_0[pop_0 < cut]))

     cuts_array = np.linspace(min(pop_0_proj), max(pop_1_proj), 1000)

     signal_background_sqrt = [s_b_sqrt(cuts_array[i], pop_0_proj, pop_1_proj) for i
      ↪in range(len(cuts_array)-1)]

     plt.figure()
     plt.xlabel(r'$\lambda_{cut}$')
     plt.ylabel('S/B ratio')
```

```
plt.plot(cuts_array[:-1], signal_background_sqrt, label = r'$S/\sqrt{S+B}$')
plt.legend()
None
```

```
/tmp/ipykernel_10602/4171118362.py:2: RuntimeWarning: invalid value encountered
in double_scalars
  return len(pop_0[pop_0 < cut])/np.sqrt(len(pop_1[pop_1 < cut])+len(pop_0[pop_0
< cut]))
```



h)

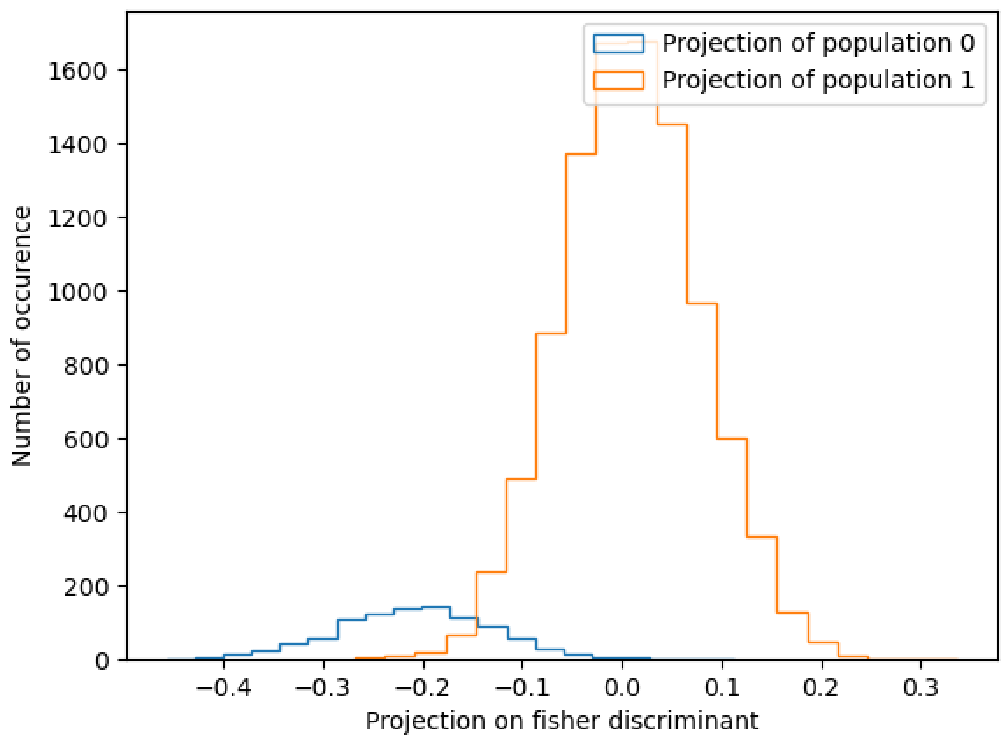
[7]:
```
p1 = pd. read_hdf ('two_populations.h5', key='P_1')

p0 = pd. read_hdf ('two_populations.h5', key='P_0_1000')

mu_p0 = np.array([np.mean(p0['x']), np.mean(p0['y'])])
mu_p1 = np.array([np.mean(p1['x']), np.mean(p1['y'])])

print('mu_p0 ',mu_p0)
print('mu_p1 ',mu_p1)
```

```
p0_np = np.array(p0)
p1_np = np.array(p1)


V_p0 = np.dot((p0_np-mu_p0).reshape(2, 1000), (p0_np-mu_p0))
V_p1 = np.dot((p1_np-mu_p1).reshape(2, 10000), (p1_np-mu_p1))

S_W = V_p0 + V_p1

print(S_W)

S_B = np.dot((mu_p0-mu_p1).reshape(2,1), (mu_p0-mu_p1).reshape(1,2))

S_W_inv = np.linalg.inv(S_W)

A = np.dot(S_W_inv, S_B)

w , v= np.linalg.eig(A)

#want biggest eigenvalue, thus take the first one

print('eigenvalues', w)
print('eigenvectors', v)

eigen_value = w[0]
eigen_vector = v[:,0]

print('Lambda', eigen_value)
print('Linear Fisher Discriminat', eigen_vector)

pop_0_proj = eigen_value * np.dot(p0_np, eigen_vector)
pop_1_proj = eigen_value * np.dot(p1_np, eigen_vector)
```

```
mu_p0  [-0.02678076  3.01578747]
mu_p1  [6.0962719  3.17467385]
[[1178.01319517  888.03358762]
 [-934.62047268 -500.92371023]]
eigenvalues [-7.79785121e-02 -8.67361738e-19]
eigenvectors [[-0.47709725  0.02594015]
 [ 0.87885051 -0.9996635 ]]
Lambda -0.07797851206133405
Linear Fisher Discriminat [-0.47709725  0.87885051]
```

[8]:
```
pop_0_proj = eigen_value * np.dot(p0_np, eigen_vector)
pop_1_proj = eigen_value * np.dot(p1_np, eigen_vector)

plt.figure()
```

```
plt.hist(pop_0_proj, bins = 20, label = 'Projection of population 0',
 ↪histtype='step')
plt.hist(pop_1_proj, bins = 20, label = 'Projection of population 1',
 ↪histtype='step')
plt.xlabel('Projection on fisher discriminant')
plt.ylabel('Number of occurence')
plt.legend()

None
```

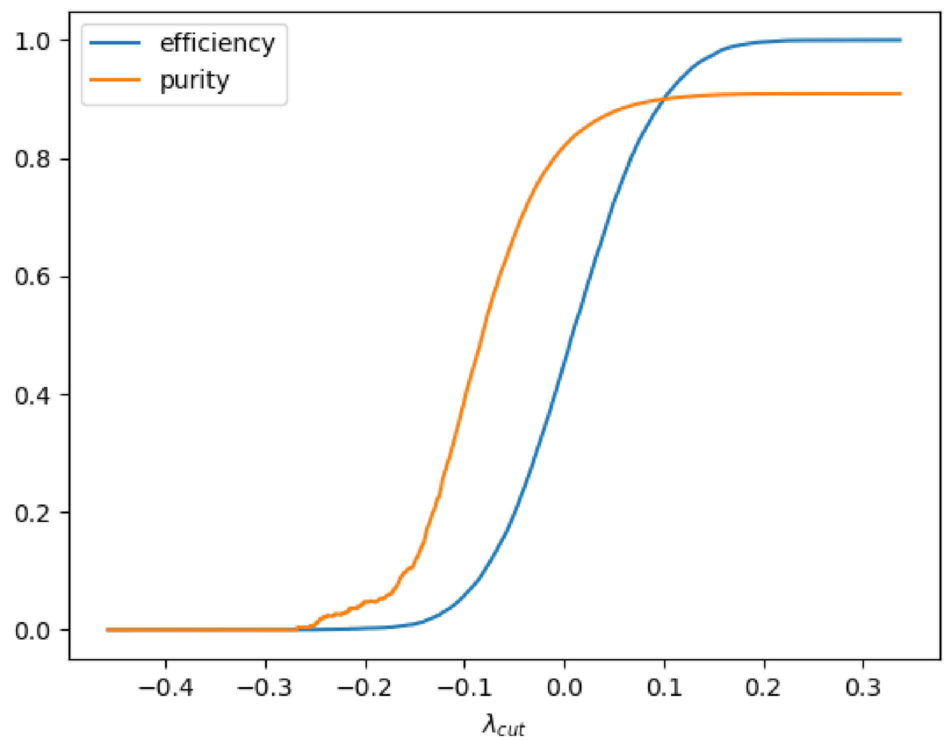


Compared to our previous signal, we have fewer datapoints. Additionally the distribution has a bigger spread and is shifted to the left

[11]:
```
cuts_array = np.linspace(min(pop_0_proj), max(pop_1_proj), 1000)

eff_array = np.array([eff(cuts_array[i],pop_0_proj, pop_1_proj)  for i in
 ↪range(len(cuts_array))])
purity_array = np.array([purity(cuts_array[i],pop_0_proj, pop_1_proj)  for i in
 ↪range(len(cuts_array))])
```

```python
plt.figure()
plt.plot(cuts_array, eff_array, label='efficiency')
plt.plot(cuts_array, purity_array, label='purity')
plt.xlabel(r'$\lambda_{cut}$')
plt.ylabel('')
plt.legend()
```

[11]: <matplotlib.legend.Legend at 0x7f929c37ba30>

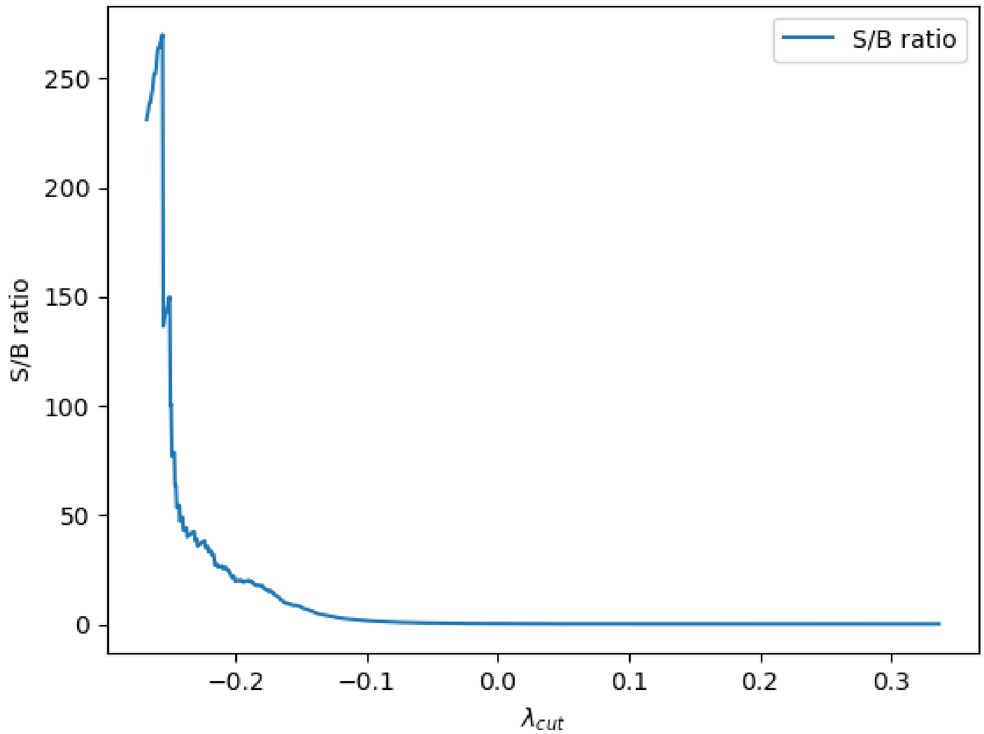


As a result of the aforementioned shift, the efficiency and purity curves are shifted to the left as well. Due to the smaller cross section between signal and noise, the purity reaches a higher upper limit.

[12]:
```python
cuts_array = np.linspace(min(pop_1_proj), max(pop_1_proj), 1000)

signal_background = [s_b_ratio(cuts_array[i], pop_0_proj, pop_1_proj) for i in␣
 ↪range(len(cuts_array)-1)]

plt.figure()
plt.xlabel(r'$\lambda_{cut}$')
plt.ylabel('S/B ratio')
```

```
plt.plot(cuts_array[:-1], signal_background, label = 'S/B ratio')
plt.legend()
```

[12]: <matplotlib.legend.Legend at 0x7f929c33bbb0>

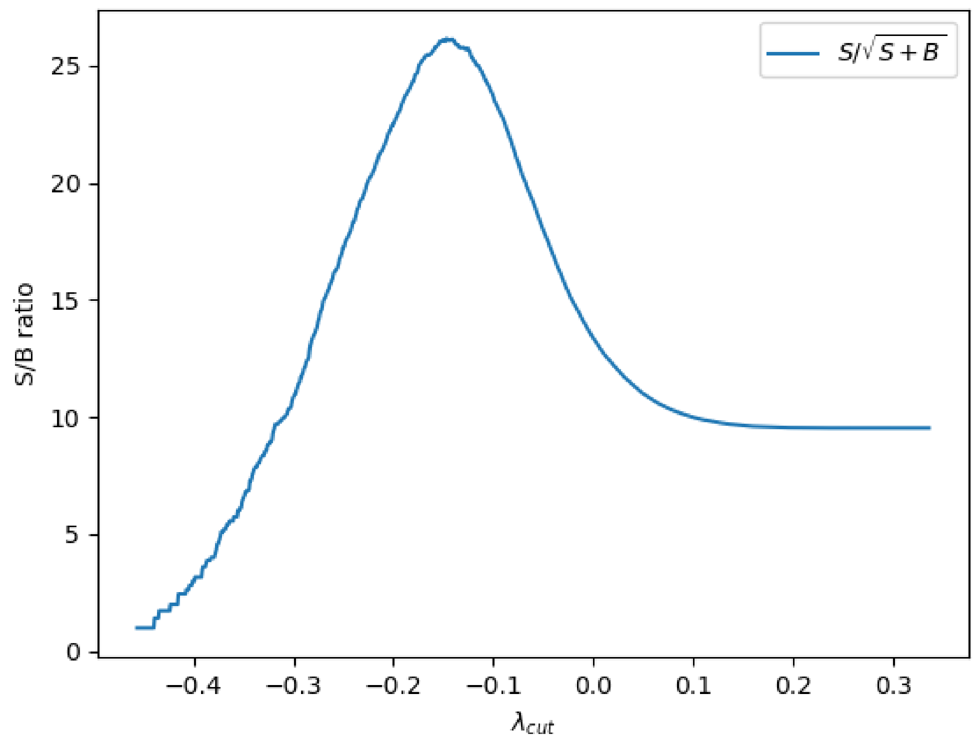


The same shift to the left occurs here, moving the maximum further left. The maximum is smaller here, as the signal itself has significantly fewer values as the noise.

[13]:
```
cuts_array = np.linspace(min(pop_0_proj), max(pop_1_proj), 1000)

signal_background_sqrt = [s_b_sqrt(cuts_array[i], pop_0_proj, pop_1_proj) for i␣
↪in range(len(cuts_array)-1)]

plt.figure()
plt.xlabel(r'$\lambda_{cut}$')
plt.ylabel('S/B ratio')
plt.plot(cuts_array[:-1], signal_background_sqrt, label = r'$S/\sqrt{S+B}$')
plt.legend()
None
```

```
/tmp/ipykernel_10602/4171118362.py:2: RuntimeWarning: invalid value encountered
```

```
in double_scalars
  return len(pop_0[pop_0 < cut])/np.sqrt(len(pop_1[pop_1 < cut])+len(pop_0[pop_0
< cut]))
```



The graph shows a similar form to the previous one with the peak shifted to the left. Additionally the dropoff after the peak is bigger, as the denumerator grows bigger with more background data (and less signal data).

Sheeeeeeeesh 420

Dienstag, 31. Mai 2022 22:43

Könnten wir bei diesem Blatt bitte nicht -1,5P Abzug bekommen?