

Exercises_7

November 5, 2018

1 Exercise 7

Maximum Likelihood method

```
In [ ]: from __future__ import print_function
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit, minimize, fsolve
from scipy.stats import norm, chi2, lognorm

In [151]: measurements = np.array([97.8621, 114.105, 87.7593, 93.2134, 86.6624, 87.4629, 79.777,
91.5024, 87.7737, 89.6926, 133.506, 91.4124, 94.4401, 97.3968, \
108.424, 103.197, 88.2166, 142.217, 89.0393, 102.438, 95.7987, \
94.5177, 96.8171, 90.903, 132.463, 92.3394, 84.1451, 87.3447, \
92.2861, 84.4213, 124.017, 90.4941, 95.7992, 92.3484, 95.9813, \
88.0641, 101.002, 97.7268, 137.379, 96.213, 140.795, 99.9332, \
130.087, 108.839, 90.0145, 100.313, 87.5952, 92.995, 114.457, \
90.7526, 112.181, 117.857, 95.2804, 115.922, 117.043, 104.317, \
126.728, 87.8592, 89.9614, 100.377, 107.38, 88.8426, 93.3224, \
138.947, 102.288, 123.431, 114.334, 88.5134, 124.7, 87.7316, 84.7141, \
91.1646, 87.891, 121.257, 92.9314])
```

1.1 1-D Maximum likelihood fit

We have a set of measurements which are distributed according to the sum of two Gaussians (e.g. this can be signal and background).

$$\rho = \frac{1}{3} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2} \left(\frac{x-p}{\sigma} \right)^2} + \frac{2}{3} \frac{1}{\sqrt{2\pi\sigma_b^2}} e^{-\frac{1}{2} \left(\frac{x-p_b}{\sigma_b} \right)^2}$$

where for one of the two peaks the parameters are known already

$$p_b = 91.0$$

$$\sigma_b = 5.0$$

```
In [228]: def likelihood_point(x, position, width):
return ...
```

First, we assume the width of the peak we want to fit is already known: $\sigma = 15.0$. Perform a 1-D Maximum Likelihood fit for the position of the peak p .

Complete the functions below which return the likelihood and negative log likelihood (NLL).

```
In [347]: def likelihood_1d(params):
           # hint: for products use np.prod
           return ...

           def nll_1d(params):
               return -np.log(likelihood_1d(params))
```

Minimize the NLL and give the best-fit result, including asymmetric errors and plot the NLL.

```
In [ ]: # find numeric minimum of NLL
        # hint: use e.g. minimize from scipy.optimize
        ...

        # hint: to compute the errors (solve roots of equation) use fsolve from scipy.optimize
        print("position:", ...)
        print("negative error:", ...)
        print("positive error:", ...)
        plt.show()
```

What happens if you try to maximize the likelihood directly?

1.2 2-D Likelihood fit

Now we perform the 2-D Maximum Likelihood fit, fitting for both σ and p at the same time.

```
In [350]: def likelihood(params):
           return ...

           def nll(params):
               return -np.log(likelihood(params))
```

Minimize the NLL and find the best-fit result.

```
In [ ]: solution = ...
        print("position:", ..., "width:", ...)
```

Create a 2D contour plot of the 1, 2 and 3 σ contours of the NLL and plot the best-fit solution.

```
In [ ]: #
        p1 = plt.contour(...)
```

Compute numerically the error matrix of the NLL for the 2-D fit.

```
In [ ]: from scipy.misc import derivative

        # compute the error matrix
        # hint: * you can use "derivative" from scipy.misc to compute numeric derivatives
        #         * for the mixed partial terms, the use of lambda functions might be practical
        #         function depending on more than one variable to a function depending on one
```

```

A = np.linalg.inv([
    [
        derivative(...),
        derivative(...)
    ],
    [
        derivative(...),
        derivative(...)
    ]
])
print(A, "\nsigma(position):", np.sqrt(A[0,0]), "sigma(width):", np.sqrt(A[1,1]))

```

1.3 Binned ML fit

With the same data as above, we now perform a binned ML fit and compare with the unbinned fit. First, create a histogram of the data using `np.histogram`.

```

In [ ]: nBins = 10
        histoMax = 170
        histoMin = 70
        binWidth = (histoMax - histoMin)/nBins
        h0 = np.histogram(...)
        print(h0[0])
        print(h0[1])

```

Compute the binned NLL:

```

In [375]: def nll_binned(params):
           # params is a list of [position, sigma]
           #...
           return #...

```

Minimize the binned NLL:

```

In [376]: solution_binned=...
           print(solution_binned)

fun: -138.93433719876123
jac: array([-1.90734863e-06,  1.90734863e-06])
message: 'Optimization terminated successfully.'
nfev: 60
nit: 6
njev: 15
status: 0
success: True
x: array([116.43876363,  15.33581135])

```

Make a contour plot of the 1,2, and 3 σ contours for the binned NLL and overlay it with the unbinned contours.

```
In [ ]: # show the two contour plots superimposed  
plt.show()
```

Repeat the same for 50 bins:

```
In [ ]: # show the two contour plots superimposed for 50 bins
```