Karlsruher Institut für Technologie

Computerpraktikum am 02.06. und 03.06.2022

Verantwortlich für Vorlesung bzw. Computerpraktikum:
Dr. P. Goldenzweig, Dr. R. Wolf, F. Metzner
Tutoren: M. Bauer, P. Ecker, M. Horzela, Dr. S. Stefkova, Dr. N. Trevisani, T. Voigtländer

Am 26. und 27. Mai finden keine Beratungstutorien statt!

Computerpraktikum zur Vorlesung Moderne Methoden der Datenanalyse - Blatt 5

Exercise 5.1: Parameterization of Data

- **Exercise 5.1.1:** **obligatory**

  If the underlying probability distribution function (PDF) of a dataset is unknown, empirical fit functions have to be employed. The most common empirical fit functions are n-th order polynomials with constant coefficients p_k to be determined by the fit:

  $$P_n(x) = \sum_{k=0}^{n} p_k \, x^k.$$

  The fit results can usually be "stabilized" by using orthogonal polynomials

  $$L_n(x) = \sum_{k=0}^{n} p_k \, l_k(x),$$

  where $l_k(x)$ are Legendre polynomials, which can be defined recursively by

  $$l_0(x) = 1; \quad l_1(x) = x;$$

  $$(k+1)\, l_{k+1}(x) = (2k+1)\, x\, l_k(x) - k\, l_{k-1}(x)$$

  The Legendre polynomials fulfill the orthogonality relation

  $$\int_{-1}^{1} \mathrm{d}x \, l_m(x) \, l_n(x) = \frac{2}{2n-1} \delta_{mn},$$

  where δ_{mn} denotes the Kronecker delta.

  Fit the data points given by the following pairs of x and y values assuming a constant uncertainty of $\sigma_y = 0.5$ for y, and no uncertainty for x:

```
x = { -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1,
       0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 }
y = { 5.0935, 2.1777, 0.2089, -2.3949, -2.4457, -3.0430, -2.2731,
     -2.0706, -1.6231, -2.5605, -0.7703, -0.3055, 1.6817, 1.8728,
      3.6586, 3.2353, 4.2520, 5.2550, 3.8766, 4.2890 }
```

a) Use the polynomials $P_2(x)$, $P_3(x)$, ..., $P_7(x)$ as fit functions.

b) Use the Legendre polynomials $L_2(x)$, $L_3(x)$, ..., $L_7(x)$ as fit functions.

Plot the data and the fitted curves for all fits. In total 12 fits have to be performed. Compare the resulting values for p_k and their correlation matrices (to be obtained most conveniently via the `GetCorrelationMatrix()` methodof the Root class `TFitResult` or via the `scipy.optimize.curve_fit()` function).

In which sense is the fit using orthogonal polynomials "more stable"? Discuss which order n you would choose for the polynomial fit function.

Hint: When using Root to solve this exercise, a convenient framework for fitting and visualisation of problems like this one is included in the Root class `TGraphErrors` and its methods.

- **Exercise 5.1.2:** **obligatory**

In an accelerator experiment, the following data are numbers of events measured in 60 energy intervals equally distributed between 0 and 3 GeV:

6	1	10	12	6	13	23	22	15	21	23	26	36	25	27	35	40	44	66	81
75	57	48	45	46	41	35	36	53	32	40	37	38	31	36	44	42	37	32	32
43	44	35	33	33	39	29	41	32	44	26	39	29	35	32	21	21	15	25	15

The data shows a signal resonance visible on top of a background sample. For the uncertainties of all data points we assume the statistical uncertainty according to a Poisson distribution.

The goal of this exercise is to extract information on the signal by parameterizing both signal and background. The information we are interested in are the width of the signal (which is related to the lifetime) and the number of signal events. Let us assume that the background can be parametrized as a polynomial of second order in the energy (i.e., a function with 3 parameters), and the signal as a Lorentz function (also 3 parameters) given by

$$L(x; A_{norm}, \mu, \Gamma) = \frac{A_{norm}}{\pi} \frac{\Gamma/2}{(x-\mu)^2 + (\Gamma/2)^2}$$

There are two possible methods to extract the signal:

a) Fit the data with a function with 6 parameters composed of the sum of the signal function plus the background function.

b) Define two intervals, left and right of the signal peak, to fit the background function. Then, fit the signal function to the data in the signal region after subtracting the fitted background function.

   There are (at least) two ways how to exclude certain data points for the fit. Either you can define new arrays for the fit which contain only a subset of the original data points, or you can define your own fit function which excludes certain intervals. For a Root example how to do this (not in Python, but in C) see here: https://root.cern/doc/master/fitExclude_8C.html

Plot the fitted functions on top of the data. Determine the width of the Lorentz peak and the number of signal events and their statistical uncertainties, and compare the results of both methods.

Exercise 5.2:
Minimization via Simulated Annealing obligatory

Data analysis often requires to find the optimal solution, e.g., the minimum of a function in a multi-dimensional space. As an example we use the following two-dimensional function which has several local minima, but just one global minimum:

$$f(x, y) = (x^2 + y - a)^2 + (x + y^2 - b)^2 + c \cdot (x + y)^2 \tag{1}$$

with arbitrary parameters a, b and c (for $c = 0$ this function would be the *Rosenbrock function*, which is often used to validate minimization algorithms). For this exercise sheet, we make the arbitrary choice $a = 11$, $b = 7$, $c = 0.1$. With these parameters, the function $f(x, y)$ has four local minima of different depth.

In this exercise you will write your own minimization algorithm following the *Simulated Annealing* strategy and test with the above defined function. Use the code fragment given in the jupyter notebook as help.

a) Play with the parameters: initial and final temperature, cooling speed, and step size. Choose a starting point close to the global minimum, and check if the algorithm converges into the minimum.

b) Choose a starting point close to a local minimum which is not the global minimum, e.g., $(x, y) = (3, -2)$. Find a set of parameters for the algorithm such that it converges to the global minimum, but still keeping the number of iterations as low as possible, and motivate your choice.
For tuning the parameters, you have two possibilities: Either you perform a scan over a meaningful range for each parameters, or you study how the algorithm reacts on changing certain parameters and then try to tune them by hand. In any case, first think what is the role of each parameter in the algorithm. E.g., both, the difference between initial and final temperature, and the cooling speed directly affect the number of iterations, but the temperature scale in addition affects the probability for jumps.

c) Repeat the analysis for different random seeds. If the minimum found depends on the random seed, re-tune the parameters of the algorithm until the result is independent of the random seed.