Exercises for the lecture „Moderne Methoden der Datenanalyse"

Prof. Dr. S. Hansmann-Menzemer, M. Schiller
Physikalisches Institut der Universität Heidelberg

June 8th 2009

Exercise 5: Maximum Likelihood Fit

The fitting of parametrised functions to measured data is important for checking models and determining their parameters. However, there are some pitfalls which can lead to wrong conclusions.

Exercise 5.1

We have generated exponential distributed random numbers in Exercise 2.5 and stored in an ntuple. These numbers can be interpreted as the measurements of the decay times of particles, the exponential distribution

$$f_\tau(t) = \frac{1}{\tau} e^{-t/\tau}$$

is used to describe the lifetime τ of these particles. Remember, we have generated the distribution with $\tau = 1$. First, use 50 entries from your ntuple for the fit.

We we will use the package MINUIT to perform an unbinned likelihood-fit to estimate the lifetime τ of the particles. Minuit is a very efficient package for minimisation and originates from the CERN library. It was ported to C++ and is available within the ROOT environment. The corresponding class is named `TMinuit`. We already used the package in previous exercises when we invoked a fit to a histogram in an interactive ROOT session.

The *likelihood-function* is defined as:

$$L(\vec{\theta}) = \prod_{i=1}^{n} f(x_i, \vec{\theta})$$

where the x_i are our (simulated) data following the probability density function $f(x,\vec{\theta})$ and $\vec{\theta}$ is the (set of) parameter(s) we want to determine. Note that $L(\vec{\theta})$ is regarded as a function of $\vec{\theta}$, the data $\vec{x}$ is treated as constant (the experiment is over). We expect that the likelihood function gives a high value for the true value of the fitted parameter $\vec{\theta}$.

For practical reasons (which?), one minimises the negative log-likelihood function:

$$-LogL(\vec{\theta}) = -\sum_{i=1}^{n} \log f(x_i, \vec{\theta})$$

Note that we make an *unbinned* fit here. What is the advantage over a binned fit?

To use MINUIT, you have to provide a user-defined function `fcn` which is used to compute the value of the function to be minimised:

```
void fcn(Int_t& npar, Double_t* gin, Double_t& f, Double_t*
    par, Int_t iflag);
```

This is the meaning of the variables:

`npar`	number of free parameters involved in minimisation
`gin`	computed gradient values (optional)
`f`	function to be minimised
`par`	vector of constant and variable parameters
`flag`	to switch between several actions of FCN

In the template we have already added code to cover cases where the function should do nothing, e.g. when called for initialization. It is left to you to program the default case. You need to access the ntuple with the random numbers (already done in the template).

The estimated value of the lifetime is stored in `par[0]`.

```
double EstimatedTau = par[0];
```

Add a loop over the entries of the tree and calculate `-LogL`. The i-th value in the ntuple can be accessed by first calling `GetEntry(i)` and then `GetArgs()[0]` (gives 0th argument) of the ntuple.

After you have finished the calculation of the likelihood function in `fcn` you need to setup MINUIT. Create an instance of the class `TMinuit` and tell MINUIT the name of the function `fcn` used in the minimisation process:

```
TMinuit *minu = new TMinuit(NumPar);
minu->SetFCN(fcn);
```

where `NumPar` indicates the number of parameters to be fitted (in our case, we only have one free parameter).

The MINUIT package is controlled via a number of commands which you have to call subsequently. First, you need to define two variables:

```
double arglist[10];
int ierflg = 0;
```

You may set the print level of MINUIT's output:

```
minu->SetPrintLevel(1);
```

The setting 1 (as above) means the default level of output, -1 means no output.

Next, introduce all parameters to MINUIT:

```
minu->mnparm(ParID, "ParName", StartValue, StepSize,
    LowerBound, UpperBound, ierflg);
if (ierflg != 0) cout << `` Error setting parameter `` <<
    endl;
```

`ParID` is an integer valued variable used to indicate the number of the parameter. The first parameter has the number 0. `"ParName"` is an arbitrary name for the parameter to be fitted. You also have to provide a value at which the fit starts (`StartValue`) and a step size (`StepSize`). You may limit the range of

values the parameter can take (e.g. to prevent unphysical solutions): Parameter $\in$ [LowerBound, UpperBound]. Use 0.0 if you don't want a boundary. Each parameter in the fit has to be introduced to MINUIT in this way.

Before you can start the fit, you need to tell MINUIT that you are going to perform a likelihood fit (and not a χ^2 one[1]). To do so, use the following piece of code:

```
arglist[0] = 0.5;
minu->mcexcm("SET␣ERR", arglist, 1, ierflg);
```

Then, set the maximal number of iterations MINUIT may use in the fit, e.g.

```
arglist[0]  = 500;
```

Use again the pattern

```
minu->mnexcm("COMMAND",arglist, 1, ierflg);
```

to steer MINUIT. To actually do the fit, use the following commands (in the given order):

"SIMPLEX"	(a "simple" algorithm to find a first estimate of the minimum)
"MIGRAD"	(a more sophisticated algorithm)
"HESSE"	(to compute the Hessian matrix for the error calculation)
"MINOS"	(to calculate errors)

Note that all commands have to be in capitals. After the fit has completed, you probably want to see the results:

```
double fmin,edm,errdef;
int nvpar,nparx,icstat;
minu->mnstat(fmin, edm, errdef, nvpar, nparx, icstat);
minu->mnprin(4,fmin);
```

The command `mnstat` retrieves the values at the minimum from MINUIT and `mnprin` prints it (here, a verbosity level of 4 was chosen). To store the fit-result in variables, the following command is used:

```
minu->mnpout(ParID, chname, Value, ParErr, LowerLimit,
    UpperLimit, InternalNr);
```

where `ParID` is the number associated with the name used in the definition of the parameter, `chname` is a variable of type `TString`, `Value` contains the value of the parameter at the minimum with the associated (parabolic) error `ParErr`. `LowerLimit` and `UpperLimit` contain the values of the parameter bounds, `InternalNr` is an integer-valued variable containing an internal number (if any).

Plot a histogram from 0 to 5 with the entries used in the log likelihood fit together with the fitted function normalised to the number of entries. Display the log likelihood value as a function of the fit parameter τ from 0.5 to 5. What can be learned from this plot about the errors of the fitted parameter? Redo this exercise with all entries in the ntuple.

[1] The uncertainties of the fitted parameters are determined from the curvature of the function which is minimized in the fit. The common definitions of χ^2- and likelihood fits differ in terms of their normalization, and hence, the meaning of the curvature. MINUIT was written with χ^2 fits in mind (and its default error definition is indeed `1.0`), this has to be adjusted for likelihood fits. Can you work out where this difference comes from?

Exercise 5.2

An estimator, in particular from the maximum likelihood method, may not be an **unbiased** estimator of the parameter's true value. Remember that an estimator is said to be unbiased if its expectation value is equal to the true value, i.e. for parameter a

$$< \hat{a} >= a$$

In case you are worrying, note that in almost all cases the maximum likelihood estimator is unbiased if the number of data points are large. For any probability density function, the sample mean is an unbiased estimator of the expectation value.

You can test whether or not the estimator for τ is biased by repeating Exercise 1 a number of times, each time based on an **independent** random distribution $f_\tau(t)$. Take each time 50 different events from the ntuple to examine this.

Exercise 5.3

Repeat Exercise 5.2 but this time make a likelihood fit for $\lambda = 1/\tau$ instead of τ. Is $1/\tau$ an unbiased estimator for the true quantity? Try it also with less events in an experiment (e.g. 5). What do you expect for the mean of the fitted τ and λ?