Exercises- Goodness of Fit and the CLs method

Jan Conrad, Knut Dundas Morå
knut.Mora@fysik.SU.se

INFN School of Statistics, 2015

1 Coverage of a Limit

In this exercise, we will compute the coverage of a CLs upper limit- the probability (given an hypothesis) that the limit exceeds the true value. Ideally, we want exact coverage- a 95% limit should include the true value 95% of the time.

1.1 The model

We will consider setting limits on the mean of a gaussian random variable$x$, with a known variance $\sigma = 1$. Our null hypothesis is that $\mu = 0$, i.e. $x \sim \mathrm{N}(0,1)$. The coverage for a hypothesis $\mu = \mu_1$ is the fraction of upper limits greater than μ_1.

1.2 The CL limit

Given a measurement X from a Gaussian distribution with unknown mean, we can construct an upper limit as the mean at which a fluctuation down to x or farther has a probability α:

$$p_{\mathrm{s+b}} = p(x \leq X|\mu) = \int_{\infty}^{X} \mathrm{N}(x|\mu,1)$$
$$\mu_{\mathrm{limit}} = \mu \text{ so that } p_{\mathrm{s+b}} = \alpha$$

1.3 The CLs Limit

The CLs method is a common procedure for computing limits in particle physics. Higgs limits were one prominent example. The aim is to avoid excluding hypotheses that one does not have sensitivity to. To do this, the CLs is penalized using $p_{\mathrm{b}} = p(x \leq X|0)$:

$$\mathrm{CL_s} = \frac{p_{\mathrm{s+b}}}{1 - p_{\mathrm{b}}}$$

The upper limit is found by finding the largest μ_{limit} so that $\mathrm{CL_s} = \alpha$.

1.4 The exercise

You can use the ROOT function `ROOT::Math::normal_quantile(double z,double sigma)` to invert the normal CDF given the significance level (`z`) and the standard deviation.

- When you have functions to compute the CL and CLs limits, try out what happens with the upper limit if you happen to observe some likely values (say, -1,0,1 and 3).
- If you have time, you can generate 10000 or so random numbers and compute the CL and CLs limits and put them in two histograms.
- Finally, to compute the coverage, you need to generate (10000 should do fine) random numbers with the mean you wish to test μ_{test}, and record the fraction of the limits above μ_{test} for the CLs and CL limits.
- Compute the coverage for at least a couple of values between 0 and 5, or, if you have time, make a graph of the coverage. What happens to the CLs coverage when the mean approaches zero?

A proposed solution can be found in `CLs.cpp`

2 Goodness of Fit

In this exercise, we will compare two methods for computing the goodness of fit for unbinned data. Both methods compare the Empirical Distribution Function , defined for sorted observations X_i:

$$\text{EDF}(x) = \begin{cases} 0 & \text{if} x < X_1 \\ 1 & \text{if} X_N < x \\ i/N & \text{for} X_i \leq x < X_{i+1} \end{cases}$$

with the CDF.

Kolmogorov test

The Kolmogorov-Smirnov test of GOF is a simple and widely used test. It considers the maximal difference between the EDF and the CDF for all x: $D_{KS} = \sqrt(N) \cdot sup|\text{EDF}(x) - \text{CDF}(x)|$for all x. The p-value of this test may be computed in e.g. ROOT: $p =$`TMath::KolmogorovProb(D_{KS})` [1]

[1]Note that some packages, e.g. `scipy.stats.kstest` may include corrections to the asymptotic formula at low N

Anderson-Darling

The Anderson-Darling distribution is a more involved test, using the integrated quadratic distance between the EDF and CDF (weighted higher at the tails at the distribution).

2.1 Model

We will use the Gaussian distribution for our model again (although the example script extends this so that you can change the shape of the distribution, i.e: $p(x|\mu, \sigma, \alpha) \sim \exp(-0.5|(x-\mu)/\sigma|^{\alpha})$.

2.2 Exercise

In `ROOT`, you can use the `GoFTest` program to compute GOF, using either the Kolmogorov or Anderson-Darling tests. Your `ROOT` folder contain a tutorial macro `tutorials/math/goftest.C` that showcases the different uses, both GOF tests, and tests between two samples. In the tutorial script, you also get an example of how you can initialise the `GoFTest` class with a PDF of your choosing:

```
int nvalues;
double * values = new double[nvalues];
ROOT::Math::Functor1D f(&TMath::Gaus);
double min = 3*TMath::MinElement(nvalues, values)
    ;
double max = 3*TMath::MaxElement(nvalues, values)
    ;
ROOT::Math::GoFTest* goftest = new ROOT::Math::
    GoFTest(nvalues, values, f, ROOT::Math::
    GoFTest::kPDF, min,max);
```

where `double * values, int nvalues` represent the measured values and the size of the array, respectively.

The class `PDFunctionin` `gof_root.cpp` shows how you can write your own function in `ROOT`, which can then be passed to the GoodnessOfFit object:.

- To compute the power of a test for a certain alternative hypothesis H1 and significance $\alpha = 0.05$, you need to compute how many times you can exclude H0 if H1 is true. This is conveniently done with Monte Carlo:

- Generate a large numbers of observations (e.g. 10000 sets of 50 numbers) and compute the p-value using the KS or AD test.
- The power of the test is the fraction of p-values less than α- i.e. what fraction of times H0 is rejected
- Compute the power for 10 H1s where μ is between -1 and 1 for KS and AD and make a graph of the power versus μ. Is one test much better than the other?
- Repeat the above exercise, but change σ from 0.5 to 2 instead of μ. How does the tests compare?

A proposed solution can be found in `gof_root.cpp` The function `void exampleGOFPlot` will plot the power of the KS and AD tests for either μ,σ or α.