Point estimation

1 Poisson process with unknown background

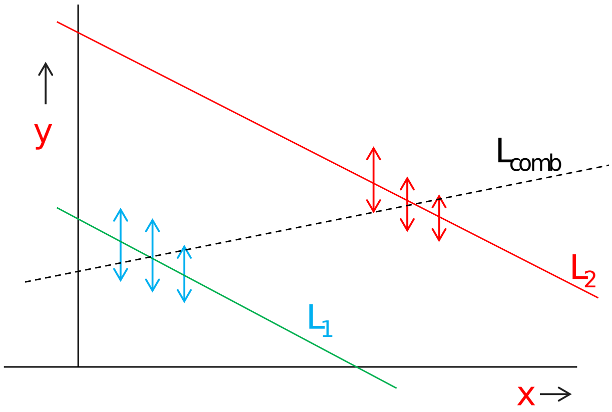
We come back to the example seen in class: $\gamma\gamma \to \gamma\gamma$ from ATLAS (Nature Phys. 13 (2017) no. 9, 852-858).

- $D = 13$ observed events
- $B = 2.6 \pm 0.7$ background events (Poisson distributed, estimated from a control sample)

The unknowns are the expected background count b and the expected signal count s. The expected event count is $d = b + s$.

1. Write the likelihood $\mathcal{L}(D|s, b)$.
2. What is the MLE $\hat{s}_{\text{MLE}}$? Draw the profile likelihood $-2\ln\mathcal{L}_{\text{prof}}(D|s)$ (e.g. using a `TGraph`, or with any other tool).
3. Draw the (marginalised) posterior for s, for each of the following priors for s (and a flat prior for b). You will perform the marginalisation using the MCMC method, also drawing the Markov chain in one of the cases.
   (a) using a flat prior in $[0, +\infty[$
   (b) using Jeffrey's prior for a Poisson process without background
   (c) using Jeffrey's prior for a Poisson process with background?
4. Estimate credible intervals at 68% confidence level using the three cases above.
5. **Profile likelihood scan.** On the same graph, draw:
   (a) The 2D likelihood scan in s and b, as a coloured histogram;
   (b) The contour corresponding to $2\Delta\ln\mathcal{L} = 1$;
   (c) The profile likelihood "path" $\hat{\hat{b}}(s)$.
6. Estimate the approximate 68% confidence level interval using the profile likelihood scan. Compare to the Bayesian credible intervals.

2 Straight line tracking



We consider a tracking detector without magnetic field (particle trajectories are straight lines $y = a + bx$). The detector is made of 2 tracking stations, each with 3 layers, situated at $x_{11} = 10$, $x_{12} = 11$, $x_{13} = 12$; $x_{21} = 20$, $x_{22} = 21$, $x_{23} = 22$ (distances in mm). We want to measure a, the impact parameter of the particle with respect to the origin, while we are not interested in b (it is a "nuisance parameter"). Each layer has a resolution of 1 mm: in other words, each measurement y_i is a random variable with a Gaussian distribution of mean $a + bx_i$ and variance 1.

In the following we assume the true values of a and b to be $a_0 = 3\,\text{mm}$ and $b_0 = 0.1$. The measured values are $y_{11} = 4.0$, $y_{12} = 3.8$, $y_{13} = 3.6$, $y_{21} = 5.2$, $y_{22} = 4.9$, $y_{23} = 4.8\,\text{mm}$.

1. Give the expression for the ML estimates $\hat{a}_{\text{MLE}}$ and $\hat{b}_{\text{MLE}}$. What is the corresponding covariance matrix? Compare the numerical value of $V(\hat{a})$ when only the 3 layers of the first station are used, or those of the second station, or all 2 stations.

2. Draw $-2\ln\mathcal{L}(a, \hat{b})$ as a function of a for the 3 first layers, the 3 last layers, and the 6 layers. Draw them together with the profile likelihood $-2\ln\mathcal{L}_{\text{prof}}(a) = -2\ln\mathcal{L}(a, \hat{\hat{b}}(a))$. You can use `TGraph` for instance.

3. An independent measurement adds the information that $\langle b\rangle = 0.1$, with a precision $\sigma_b = 0.05$. How to account for this information in the likelihood? Repeat the previous question (likelihood drawing) with the new likelihood.

4. **Data combination.** Use the BLUE method to combine the estimates $\hat{a}_{\text{MLE},1}$ and $\hat{a}_{\text{MLE},2}$. You will combine the measurement for a, considering the uncertainty on a coming from b as correlated between the two stations.

Technical tools

See `https://root.cern.ch/doc/master/`

- `TGraph`

```python
import ROOT
from array import array
x = [1,2]
y = [3,4]
```

```
g = ROOT.TGraph(len(x), array('d',x), array('d',y))
c1 = ROOT.TCanvas()
g.Draw("AL")
c1.Draw()
```

- TH1F

```
import ROOT
h = ROOT.TH1F("h","1D histo",10,0,10)
for i in range(0,100):
   h.Fill(ROOT.gRandom.Uniform(10))

c1 = ROOT.TCanvas()
h.Draw()
c1.Draw()
```

- TH2F

```
import ROOT
h2 = ROOT.TH2F("h2","2D histo",10,0,10,10,0,10)
for i in range(0,100):
   h2.Fill(ROOT.gRandom.Uniform(10),ROOT.gRandom.Uniform(10))

c1 = ROOT.TCanvas()
h2.Draw("COLZ")
c1.Draw()
```

- numpy arrays

```
import numpy

a = [0,1,2,3]

# sum of the elements of a
numpy.sum(a)

# sample mean
numpy.mean(a)

# sample variance
numpy.var(a)
```

- Numerical minimisation: `scipy.optimize.fmin`

```
from scipy import optimize

def f(x,*args):
   a = x
   b = args[0]
```

```
    return (a-b)**2

x0 = 1
b0 = 1.234
xmin = optimize.fmin(f,x0,args=(b0,),disp=False)
print(xmin)
```