Problems2

October 27, 2022

1 Problem 2 - The Calorimeter Resolution (Solutions)

A calorimeter is a generic term for a device that measures the energy of some object. Traditionally this was done by measuring small temperature changes in some material due to the energy deposited by the object, but in particle physics experiments, since the energies are very small, the devices use sophisticated processes to measure the energies of incoming particles.

Designing a high performing calorimeter is crucial in the design of a good high energy particle physics detector. Having a good energy resolution is a key design challenge.

In this question, you will need to read in a dataset and estimate the energy resolution as a function of energy from the data.

In the file `problem_2_data.csv` is a dataset of recorded energy measurements of high energy electrons from a calorimeter. This is calibration data for the experiment so the true energy `e_true` is known for every electron. The recorded energy `e_measured` is provided by the calorimeter. We want to measure the resolution of the calorimeter using `e_measured - e_true` as a function of the true energy.

You can use a Jupyter notebook to help you structure your answers. You should clearly label any plots and present results clearly - **marks will be lost for poor quality figures or unclear results!** Some useful python modules are imported below.

1.1 Part 1

First, read in the data from `problem_2_data.csv`. You could read this in using Pandas or convert the data into some other format if you prefer. The file contains rows with different beam energy (`e_true`) values. These should be the following energies $e_{true} = 10, 15, 20, 25, 30, 35, 40, 45, 50$ GeV.

a) For each `e_true` value draw draw a histogram of $e_{diff} = e_{measured} - e_{true}$.

b) We can assume that due to the resolution of the calorimeter, e_{diff} will be distributed as a Gaussian distribution $\phi(\mu_{e_{true}}, \sigma_{e_{true}})$, with parameters that are different for different values of e_{true}. For each value of e_{true} find the maximum likelihood estimators $\hat{\mu}_{e_{true}}$ and $\hat{\sigma}_{e_{true}}$. You can do this numerically using `scipy.optimize.minimize` or any other method. HINT: remember that to find the maximum likelihood estimator, you first need to write the likelihood function $L(\mu_{e_{true}}, \sigma_{e_{true}})$ for a particular dataset and then maximise it. Remember, it's nearly always easier to minimize $-\ln(L)$.

c) Plot the resulting Gaussian distribution $\phi(\hat{\mu}_{e_{true}}, \hat{\sigma}_{e_{true}})$ on top of the histograms from i). Remember, for this to look right, you're histograms should be drawn as *densities* (e.g. you

can specify `density=True` in the plotting of the histogram). Also, plot the ratio of maximum likelihood estimator to the value of e_{true} i.e plot $\frac{\hat{\sigma}_{e_{true}}}{e_{true}}$ as a function of e_{true}.

1.2 Part 2

There are a number of different contributing factors to the resolution of a calorimeter but in calorimeters like the one at the CMS experiment, one large contribution is from the limited amount of scintillation light (number of photons) which is called *stochastic noise*. The resolution can be parameterised as

$$\sigma_{e_{true}} = S\sqrt{e_{true}}$$

where S is called the *stochastic term*. With the data, we can try to estimate S.

a) Write a new likelihood function $L(S, \vec{\mu}_{e_{true}})$ *combined* across the data from different e_{true} values. Remember that the total likelihood is the product over the individual likelihoods (or the sum if using $-\ln(L)$. The vector of values $\vec{\mu}_{e_{true}}$ are still free parameters but this time you should replace $\sigma_{e_{true}}$ in the Gaussian pdfs with $\sigma_{e_{true}} = S\sqrt{e_{true}}$. Calculate the maximum likelihood estimator $\hat{S}$. HINT: You will probably find it helpful to initialise the minimization with a value around $S \approx 0.03$. You might also want to impose boundaries on the minimizer with the `boundaries=[[a1,b1],[a2,b2],...]` keyword on the call to `minimize`.

b) What is the standard deviation on your estimator $\hat{S}$? To obtain this, you can use a bootstrapping method on the original dataset and calculate the sample standard deviation of the bootstrap samples - this may be quite slow so you should use no more than 100 bootstrap samples.

c) Finally make a plot of the function $\frac{\hat{S}\sqrt{e_{true}}}{e_{true}}$ overlaid on the individual fits you obtained in Part 1. How well do the points agree with your curve?

HINT: For this problem set you can use whichever software you like but I reccomend using the following setup in python with the `iminuit` minimizer package.

[ ]:
```
%matplotlib inline

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.stats import norm
from iminuit import minimize
from scipy.interpolate import interp1d

plt.rcParams.update({'font.size': 10})
```

[ ]:
```
E_points = range(10,55,5)
n_points = len(E_points)
print(list(E_points))
```

[ ]:
```
df = pd.read_csv('problem_2_data.csv')
df.head()
```