DAML, Week 8/CP4: Data selection, p-values, hypothesis testing and significance interpretation

Christos Leonidopoulos[*]

University of Edinburgh

November 3, 2023

Abstract

We review some of the simplest statistical tools in the HEP literature used for quantifying the importance of an apparent "peak-like" morphological deviation in the data in the presence of background. The tutorial demonstrates how to calculate p-values and carry out simple hypothesis testing with the help of small `python` scripts.

1 Introduction

In searches for New Physics, we often deal with data distributions that present apparent deviations from the expected background. A common problem is the need to quantify the hypothesis that these deviations are due to an exciting new phenomenon, or are, instead, consistent with a background fluctuation. In this tutorial, we examine how simple statistical tools can be used to test the improbability of such a deviation, perform simple hypothesis testing and carry out general statistical evaluations by running toy Monte Carlo pseudo-experiments.

The workshop uses small snippets of `python` code, and the `Jupyter Notebook` web application environment [1] to display the expected code output.

[*] `Christos.Leonidopoulos@ed.ac.uk`

2 Generation of random distributions from a model

The first preliminary task that we consider is the generation of (pseudo)random "data" samples that follow a parent model distribution. We will use a first-order (linear) polynomial to model the background, and a Gaussian to model the signal. Histograms are created separately for the signal and background distributions, and added together to produce what would appear as the "data" distribution containing both signal and background. In our example we inject a relatively small number of signal events (300) into a much larger background sample (of 10,000).

CP4, problem #1 (3 points):

Create two classes, one called `Linear` (to model the background distribution), and a second one called `Gaussian` (to model the signal distribution). For both of them add an attribute `mass` array (to be initialised in the constructor) to hold the mass values that will make up the spectrum of the background and signal distributions. Add also a `next()` member function to be used for drawing random x-values (ie. corresponding to the mass) according to the parent $f(x)$ distribution. For class `Gaussian` use the built-in method `numpy.random.normal` to draw random values. For class `Linear` use the "box" method (as discussed in previous lectures):

```
# background parameters
XMIN = 0.
XMAX = 20.
intercept = 20.
slope = -1.
# signal parameters
mean = 10.
sigma = 0.5
# chose number of bins that is appropriate for the size of the statistics sample
NBINS = 100



# Evaluate method (un-normalised)
def evaluate(self, t):
   return lambda t: self.intercept + self.slope * t

def next(self):
    doLoop = True
    while(doLoop):
        # start with uniform random number in [lolimit, hilimit)
        x = numpy.random.uniform(self.lolimit, self.hilimit)
        y1 = self.evaluate(x)
        y2 = numpy.random.uniform(0, self.maxval())
        if (y2 < y1):
            filtered_x =  x
            self.mass.append(filtered_x)
            return filtered_x
```

Note how we store the accepted ("filtered") values of mass in variable `mass` array so we can access them later for plotting.

Now create class `SignalWithBackground` which will return the sum of the signal (`Signal`) and background (`Linear`) distributions weighted by the fractions of signal and background events:

CP4, problem #1 (*cont.*)

```
# Draw random number form distribution
def next(self):
    q = numpy.random.uniform()
    if( q < self.sig_fraction):
        # if here, we will draw x from signal distribution
        filtered_x = self.signal.next()
        self.mass_sig.append(filtered_x)
    else:
        # if here, we will draw x from background distribuion
        filtered_x = self.background.next()
        self.mass_bgd.append(filtered_x)

    self.mass.append(filtered_x)
    return filtered_x
```

Note how we keep track separately of drawn mass values for signal (`mass_sig`) and background (`mass_bgd`) , but also for their sum (`mass`).
Now put everything together with something like the following snippet (lots of little details omitted!), and plot the three distributions:

```
# Main code to generate and plot a single experiment
def singleToy(nevents_sig = 300, nevents_bgd = 10000):

    sig_fraction = nevents_sig/(nevents_bgd + nevents_sig)
    #Create the pdf
    pdf = SignalWithBackground(mean, sigma, sig_fraction, intercept, slope, XMIN, XMAX )

    for i in range( nevents_sig + nevents_bgd ): pdf.next()

    # retrieve the mass values for signal, background and their sum
    data = pdf.mass
    sig_data = pdf.mass_sig
    bgd_data = pdf.mass_bgd

    # plot things on same page
    myRange = (XMIN, XMAX)
    fig, axs = plt.subplots(3,1, sharex='col')

    axs[0].set_title("Signal distribution (" + str(len(sig_data)) + " entries)")
    axs[1].set_title("Background distribution (" + str(len(bgd_data)) + " entries)")
    axs[2].set_title("Total distribution (" + str(len(data)) + " entries)")
    axs[2].set_xlabel('X')

    axs[0].hist(sig_data, bins=NBINS, range=myRange)
    axs[1].hist(bgd_data, bins = NBINS)
    axs[2].hist(data, bins = NBINS)
    fig.tight_layout()
    plt.savefig('Example1.pdf')
```

The output of the script when run with these default values can be seen in Fig. 1. Even though the signal fraction is only 2.9%, there is a tantalising hint of a peak at $x = 10$. You are invited to increase the number of signal events (e.g. by a factor of three or more) and confirm how much more prominent the signal presence becomes. Conversely, reducing the number of signal events (similarly, by a factor of three or more) makes the interesting signal hint vanish in the background distribution.

We will next try to quantify the above observations. In particular, we will estimate how likely it is to get a fluctuation that is at least as “interesting” (read: extreme) as the one depicted in the bottom plot of Fig. 1 in the absence of a genuine signal.

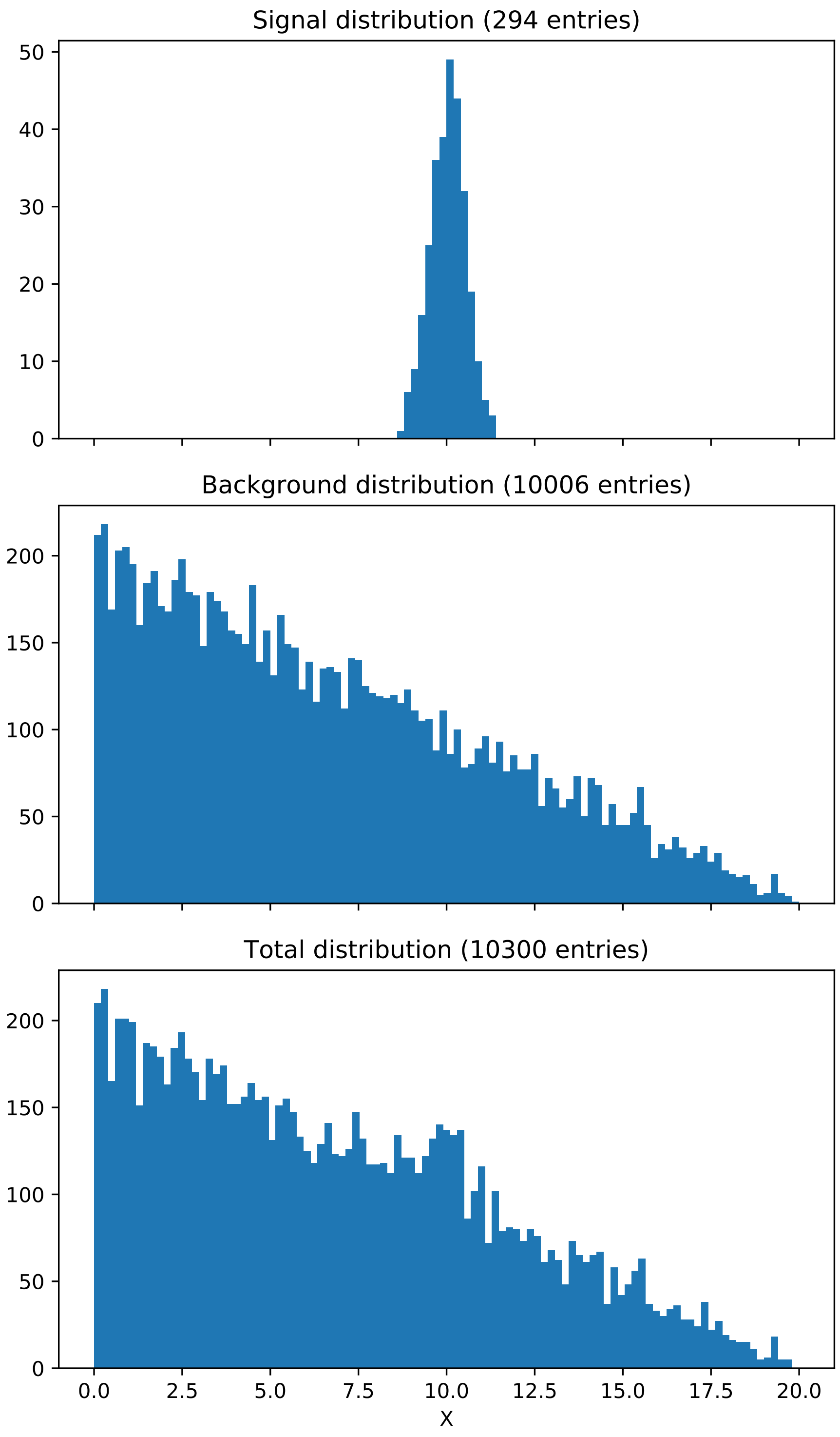

Figure 1: Output of example given in CP4, problem #1: Pseudo-random distributions drawn from a Gaussian for the signal (top) and a first-order polynomial for the background (middle). The sum of the two distributions is displayed at the bottom, with the Gaussian signal barely visible at $x = 10$.

3 p-values: quantifying background fluctuations

We now assume that we have experimentally obtained a distribution like the one at the bottom of Fig. 1 (this is our "data", combining background and signal —if any— which cannot be disentangled). The question that we are trying to answer is how likely it is that a mere background fluctuation can give us a similarly (or even more) interesting hint of a signal. The simplest way to answer this question is to carry out an event-counting experiment.

The first assumption we make is that we know nothing about the potential presence of a signal at $x = 10$. We assume, however, that we have a perfect knowledge of the background distribution, nameley its shape (a linear polynomial $p_1 \cdot x + p_0$, with $\sigma_{p_0}/p_0 \simeq \sigma_{p_1}/p_1 \simeq 0$) and absolute size (e.g. that we expect an average background population of 10,000 events between $x = 0$ and $x = 20$). For many physics processes, the observed background sample size will typically follow a Poisson distribution, namely fluctuate around a mean value λ, equal to the average (expected) number of background occurences. For large enough values of λ (here: $\lambda = 10,000$) the distribution can be approximated by a Gaussian with $\mu = \lambda$ and $\sigma = \sqrt{\mu}$. Since we know nothing about the potential presence of a signal, we can simply try to evaluate the probability that a Gaussian with $\mu = 10,000$ and $\sigma = \sqrt{\mu} = 100$ can produce a number of events, k, *at least* as high as the observed value of 10,300 (see Fig. 1, bottom).

CP4, problem #2 (1 point):

Calculate the probability that a background-only process with $\lambda = 10,000$ can fluctuate to give $k \geq 10,300$. The fastest way to do this is to employ the previously coded `Gaussian` class, by adding an `integral()` method, which can be easily implemented with method `scipy.integrate.quad`. You will need to take the ratio of two such integrals.

Calculate how large the deviation is by converting the p-value to a Z-score (or number of standard deviations). You can use the inverse error function[1], as shown in the following snippet:

```python
from scipy.special import erfinv
pvalue = ``ratio of two integrals''
n_sigmas = erfinv(1 - pvalue) * numpy.sqrt(2)
```

Now, let's try to improve our analysis. Let us assume that we know somehow that a potential signal in the data sample must be somewhere around $x = 10$. We may not know the exact value of the Gaussian mean, but we (assume that we) are pretty confident that it is not in the $x < 5$ or $x > 15$ region. In this case it makes sense to exclude these regions (as they only contain background) and focus on the $5 < x < 15$ area. This is a very rough analogy of a *selection* or *filtering* of our data sample. We always try to reduce our background (without reducing our signal). This is because the background tends to make our job of identifying the signal or carrying out related measurements more difficult.

[1]You may be tempted to calculate the Z-score by using $(10,300 - 10,000)/\sigma = 3$, however this would give you the wrong answer. This simple Z-score formula involves two-sided intervals, i.e. the 3σ deviation corresponds to the interval outside the [9,700, 10,300] region, whereas in this problem we are dealing with one-sided tails.

The next logical step is to evaluate the expected number of background events in the $5 < x < 15$ region. This can be done either analytically or numerically by using the background shape and the knowledge that the total number of expected background events in $0 < x < 20$ is 10,000. The background fraction between $x = 5$ and $x = 15$ is 50%, so the expected number of background events in this subregion is 5,000. Our problem has now been reduced to the one we dealt with earlier with the modified values $\lambda = 5{,}000$ and $k \geq 5,000 + 300 = 5,300$.

CP4, problem #3 (1 point):

First, confirm that the number of (expected) background events in the $5 < x < 15$ subregion is 50%. Then, calculate the p-value and the Z-score as in the previous problem for the updated λ and k values. You will notice that the deviation is more significant now. Why?

Note that by limiting ourselves to a narrow(er) window around the region of interest we manage to increase the sensitivity of the signal search. This is expected, since we are effectively carrying out the same simple event-counting experiment in a smaller region (with a lower background level overall) without any impact on the signal. To be able to do this, we have to make assumptions about the "region of interest" for the anticipated signal-like deviation (which may or may not be possible in general). The next step? If we are able to know the signal model precisely (e.g. that we have a Gaussian with fixed mean and sigma values) we can carry out a so-called "shape-based" analysis. This technique allows us to fully exploit the different *shapes* between signal and background, and maximally increase the sensitivity of the search. We will see how this is done in the next session.

4 Hypothesis testing: H_0 vs. H_1

As discussed in the previous sections, the distribution shown at the bottom of Fig. 1 appears to contain a small signal. We want to make a quantitative statement about how much better a model including a Gaussian signal (H_1) describes the distribution compared to the background-only linear model (H_0). Unlike the p-value description in the previous section, where a statement is made on the (im)probability that a background fluctuation can give a deviation (at least) as extreme as the one observed in the data, here the focus is on the comparison between two hypotheses: a model that contains a *specific* exotic signal (i.e. a Gaussian with fixed mean and sigma values) vs. one that contains background only. The first step of this comparison is to fit the single "data" distribution to the two models, H_1 and H_0, and evaluate the difference of the corresponding χ^2 (or log-likelihood χ^2-equivalent figure-of-merit) of the two fits.

The H_1 fit is always expected to give a more satisfactory description of the "data" distribution than the H_0 fit because of the extra fit parameter(s): the more degrees of a freedom a function has, the better the match to the data. According to Wilk's theorem [2], the difference $\chi^2(H_0) - \chi^2(H_1)$ asymptotically follows a χ^2 distribution, with the number of degrees of freedom equal to the number of extra free parameters between the H_1 and H_0 fits. This is a very useful observation, as it allows one to evaluate the statistical significance of a deviation when fitted with two different models, without the need to generate large and CPU-expensive toy MC experiments. One of the necessary conditions for the theorem to be valid is that the two hypotheses must be "nested" [3], namely that one can arrive at the simple model (H_0) from the more complex one (H_1) by setting specific values for the extra parameters (e.g. by setting the Gaussian strength to zero).

To see this in practice, we will run two fits on the spectrum with the observed deviation by using the functions that were employed to produce the signal and background distributions. We assume that we know precisely (i.e. with negligible uncertainties) the functional form for the signal, and that the background can be modelled by a linear function.

CP4, problem #4 (5 points):

In this example you should decrease the population of signal events to 150 (or ~1.5% of the total sample). Despite having a weaker signal, the shape-based analysis provides a much more powerful statistical test than mere event-counting. You should carry out two fits on the *same* "data" histogram, one assuming no signal (H_0), and a second one assuming a signal (H_1) of uknown size (but with fixed mean and sigma values). In both fits the shape of the background distribution (slope and intercept) should be free to vary. Find the difference in the (log-likelihood-equivalent) χ^2 between the H_0 and H_1 fits. Since the difference in the number of degress of freedom between the two models is just one (the size of the signal contribution), you should use this information and Wilk's theorem to calculate the p-value and the Z-score (method `scipy.stats.chi2.cdf` can be used here).

Notes:

- You should use the information from the fitting lectures to carry out the fit (preferably) with a log-likelihood maximisation (equivalent to a χ^2 minimisation). You are welcome to implement this with any method you feel comfortable with (e.g. `Minuit` or others).
- Normalisation of linear function: The linear function is supposed to represent a background distribution. It should therefore never be negative. An e.g. exponential function would always be non-negative by design, but your custom-made linear function requires some care to make sure it is well-behaved. If your fit prefers parameters that render your Linear function negative, you are doing something wrong. You should make sure that your Linear function is doing what you think it is doing (e.g. by overlaying it on top of the data distribution), and you should not try to extrapolate it to very large x-values (where it is bound to become negative eventually, since the slope is negative). Chances are that if you limit the range of the function to a well-defined region, you should be ok for its normalisation and the fit.
- If you want to implement a minimisation function from scratch one suggestion is

to use `scipy.optimize.minimize` and the χ^2-equivalent of the log-likelihood in a binned fit, given by the expression [4]:

$$\chi^2 = 2\sum_i \left[N^{(i)}_{\text{expected}} - N^{(i)}_{\text{observed}} \;+\; N^{(i)}_{\text{observed}} \log\left(\frac{N^{(i)}_{\text{observed}}}{N^{(i)}_{\text{expected}}}\right)\right]$$

where i runs over all bins, $N^{(i)}_{\text{observed}}$ is the observed number of (pseudo)data in a given bin, and $N^{(i)}_{\text{expected}}$ is the expected number of events from the theoretical model (i.e. depending on the H_0 or H_1 hypothesis each time).

Tip #1: Make sure that $N^{(i)}_{\text{expected}} > 0$. If the fit pushes this expression towards negative values, you can manually set it to e.g. `1E-3` to aid the fitting process.

Tip #2: Make sure that $N^{(i)}_{\text{expected}}$ is properly weighted to correspond to $N^{(i)}_{\text{observed}}$, i.e. that you are not using normalised-to-1 estimates.

- Wilk's theorem works with (so-called: Pearson's) χ^2 distributions. As a general rule, it is not recommended to use a χ^2 fit unless you are dealing with large statistics (which is not the case in problem #4, in particular for the high-end tail of the distribution). Instead, you are encouraged to be using a maximum likelihood estimation. The question is, then, how does the return value of your minimiser translate into something that can be used with Wilk's theorem.

  You have several options:

  - Use the output of the log-likelihood minimiser (or your custom fitter), but multiply it by a factor of 2 to turn it into a "χ^2-like" quantity. This is discussed briefly in Ref. [3]. If you need a more rigorous proof, feel free to check out Ref. [2].
  - Use the modified "χ^2-like" log-likelihood given to you in problem #4 (provided by Ref. [4]. This already includes the necessary "factor of 2", as you can see from the expression at the top of the page.

The last challenge is to confirm Wilk's theorem by generating some toy MC experiments and repeating these fits. An observed deviation equivalent to (say) about 3 standard deviations means that in the event that there is no signal, we expect such an "anomaly" in the data about a few times per mil. The relevant bit here is "in the event that there is no signal".

CP4, problem #5 (for extra credit, up to 2 points):

Run a large number of toy MC experiments by carrying out H_0 and H_1 fits but this time on spectra (i.e. the x-observable) that contain only background events. The number of background-only pseudo-experiments that you should generate depends on the p-value found in the previous problem, however it should be at least a thousand or so. You should fit the data from each pseudo-experiment i each time to both H_0 and H_1 models and produce the corresponding $\chi^2(H_0) - \chi^2(H_1)$ distribution.

Compare the tail of this distribution with the $\Delta\chi^2$ you found in the previous exercise. Is the observation consistent with Wilk's theorem?

Notes:

- One of the most important tips: Nothing better for debugging than printing and plotting.
- More general advice: you have several weeks ahead where you will be doing fits using your favourite minimiser and for which you will have to interpret the result (including occasionally applying Wilk's theorem). You are therefore strongly encouraged to make sure you understand this topic very well.

5 Conclusions

We have seen how to

- use simple `python` scripts to generate random samples following a parent model distribution,
- estimate the probability that a background fluctuation can cause a deviation similar to (or more extreme than) a hypothetical one observed in the data (p-value),
- fit a randomly generated data distribution to various functions describing background-only (H_0) and signal-plus-background (H_1) models,
- employ Wilk's theorem in a hypothesis-testing analysis, and
- run toy MC studies by running a large number of pseudo-experiments and retrieving the individual results.

For further (and more advanced) reading in the statistics literature, including real-life HEP examples, please consult Ref. [3], and the references therein.

References

[1] The `Jupyter Notebook` open-source web application, `http://jupyter.org/`

[2] S.S. Wilks, "The large-sample distribution of the likelihood ratio for testing composite hypothesis", Annals of Math. Stat. **9** (1938), 60.
Publicly available at `https://projecteuclid.org/euclid.aoms/1177732360`

[3] L. Lyons, "Methods of comparing two hypotheses" (2009), CDF note available at `http://www-cdf.fnal.gov/physics/statistics/notes/H0H1.pdf`

[4] S. Baker, and R.D. Cousins, NIM 221 (1984) 437-442