Principle of data science coursework

Alberto Plebani
ap2387

1 Section a

The statistical model is presented in Equation (1), where the background follows an exponentially decaying distribution $b(M;\lambda) = \Theta(M)^1 \lambda e^{-\lambda M}$ and the signal is a Gaussian distribution with mean μ and variance σ^2.

$$p(M; f, \lambda, \mu, \sigma) = f \cdot s(M; \mu, \sigma) + (1 - f) \cdot b(M; \lambda) \tag{1}$$

a) Prove that the probability distribution is normalised in the range $M \in [-\infty, \infty]$.

Proof. $\int_{\mathbb{R}} p(M; f, \lambda, \mu, \sigma) dM = (1-f)\lambda \int_{\mathbb{R}^+} e^{-\lambda M} dM + \frac{f}{\sqrt{2\pi}\sigma} \int_{\mathbb{R}} \exp\left(\frac{-(M-\lambda)^2}{2\sigma^2}\right)$. The first integral is $1 - f$, because the integral of the exponential from 0 to ∞ is $-1/\lambda$, whereas the integral of the second function is f, because the Gaussian distribution is normalised in $\mathbb{R}$. Therefore, the sum of the two integrals is $f + 1 - f = 1$. □

b) Because $M \in [5, 5.6]$, we need to change the normalisation factor so that the total probability equals 1. In order to do so, recall the cumulative density function (cdf) $F(X) = \int_{\infty}^{X} f(X')dX'$. Given that for the exponential $F(X) = 1 - e^{-\lambda X}$ and for the Gaussian $F(X) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{X-\mu}{\sqrt{2}\sigma}\right)\right)$, we can normalise the two different distributions separately, as in Equations (2) and (3), respectively for signal and background.

$$N_S^{-1}(\lambda, \mu, \sigma; \alpha, \beta) = \frac{1}{2}\left(\text{erf}\left(\frac{\beta - \mu}{\sqrt{2}\sigma}\right) - \text{erf}\left(\frac{\alpha - \mu}{\sqrt{2}\sigma}\right)\right) \tag{2}$$

$$N_B^{-1}(\lambda, \mu, \sigma; \alpha, \beta) = e^{-\lambda\alpha} - e^{-\lambda\beta} \tag{3}$$

Therefore the resulting pdf, displayed in Equation (4), is normalised in the range $M \in [\alpha, \beta]$

$$\text{pdf}(M; \boldsymbol{\theta}, \alpha, \beta) = N_S(\lambda, \mu, \sigma; \alpha, \beta) \cdot f \cdot s(M; \mu, \sigma) + N_B(\lambda, \mu, \sigma; \alpha, \beta) \cdot (1 - f) \cdot b(M; \lambda) \tag{4}$$

c) In order to prove that the pdf is correctly normalised, I used the code `src/solve_part_c.py`. The details to run it are in the README file. I generated 10000 random values of

[1] $\Theta(x)$ is the Heaviside step function, which returns 0 for values smaller than 0, and 1 for values greater

Number of models	α	β	Mean	Variance
10000	5	5.6	0.9999	6.1×10^{-17}
10000	0	10	1	3.3×10^{-14}
10000	-5	5	0.9999	3.2×10^{-14}
10000	-5	8	1	5.6×10^{-9}

Table 1: Some example results obtained when running the code, which display that the pdf defined in Equation (4) is normalised.

$\boldsymbol{\theta} = (f, \lambda, \mu, \sigma)$ uniformly distributed in a specified range of values, and then I numerically evaluated the integralof the pdf in $[\alpha, \beta]$ with the `np.trapz` function. I tested different values of α and β, and for all values the average of the 10000 integrals was 1, with a very low variance, as displayed in Table 1.

d) The true values for the parameters $\boldsymbol{\theta}$ are $f = 0.1$, $\lambda = 0.5$, $\mu = 5.28$ and $\sigma = 0.018$. Therefore, the true PDF (Equation (4)) has the shape displayed in green in Figure 1. From the plot we can clearly see the exponential decay of the background distribution, as well as the Gaussian representing the signal distribution, with the two components being represented separately in blue and red, respectively.

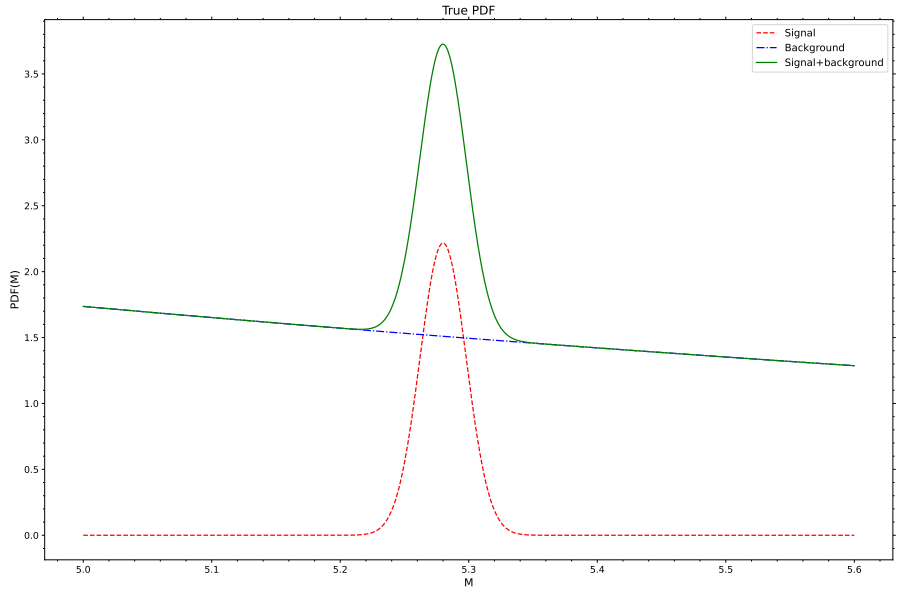

Figure 1: Plot displaying the true PDF alongside the distributions for signal and background, each rescaled by their relative fraction (10% signal, 90% background).

e) In order to generate data according to the PDF, the accept/reject (AR) method was used. This method is the more general, in that it doesn't require any knowledge of the CDF, since only the PDF is needed. In this case the CDF was analytically known, so the CDF method could have been used. Nonetheless, I decided to use the AR method

in order to give more generality to the problem, so that my class could be used also in problems where the analytical expression of the CDF is not known. The price to pay for an increased in generality is a consequential increase in the computational cost of the project, since the generation of random values requires more time than the evaluation of the CDF and its inverse function.

The AR method works as follow:

i) Generate $\vec{x} = \{x_1, \ldots, x_N\}$ random uniformly distributed points in the range $[\alpha, \beta]$

ii) Generate $\vec{y} = \{y_1, \ldots, y_N\}$ random uniformly distributed points in the range $[0, Y_{max}]$, where $Y_{max} \doteq \max_{x \in [\alpha,\beta]} [\text{pdf}(x)]$

iii) $\forall i \in [1, N]$ evaluate $pdf(x_i)$, then accept the point if $pdf(x_i) > y_i$

In this way, we are able to generate data according to the PDF.

Using this function, I generated 100k events following the true PDF (Equation (4)), and then I performed an unbinned ML fit using the `Minuit.migrad` algorithm. The algorithm returns an estimate of the free parameters $\vec{\theta}$ that minimise the negative log-Likelihood.