Principles of Data Science Coursework Report

James Hughes

December 14, 2023

Section A

Part (a)

We begin by showing that both densities s and b are properly normalised in the range $M \in [-\infty, +\infty]$. In the former case, as a first step we use a change of variables $Z = \mu + \sigma M$ such that $\frac{dM}{dZ} = \sigma$, for which the integral limits don't change:

$$\begin{aligned}\int_{-\infty}^{\infty} s(M;\mu,\sigma) &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(M-\mu)^2}{2\sigma^2}\right] dM \\ &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2}Z^2) \cdot \sigma dZ \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp(-\frac{1}{2}Z^2) dZ\end{aligned}$$

In order to prove that s is properly normalised, we simply need to show that the last expression above evaluates to 1. We do this by computing it's square, which in turn leads to an integral in two dummy variables. Below we then use the transformation to polar coordinates $(X, Y) = \rho(R, \phi) = (R\cos(\phi), R\sin(\phi))$ which has Jacobian matrix

$$\begin{pmatrix} \cos(\phi) & -R\sin(\phi) \\ \sin(\phi) & R\cos(\phi) \end{pmatrix}$$

and hence $|J_\rho(R, \phi)| = R$. Then, denoting the integral in the final expression above by I, we find:

$$\begin{aligned}
\left[\frac{1}{\sqrt{2\pi}}I\right]^2 &= \frac{1}{2\pi}\left[\int_{-\infty}^{\infty}\exp(-\frac{1}{2}X^2)dX\right]\left[\int_{-\infty}^{\infty}\exp(-\frac{1}{2}Y^2)dY\right] \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\exp(-\frac{1}{2}(X^2+Y^2))dXdY \\
&= \frac{1}{2\pi}\int_{0}^{2\pi}\int_{0}^{\infty}\exp(-\frac{1}{2}(R^2))\cdot RdRd\phi \\
&= \left[-\exp(-\frac{1}{2}R^2)\right]_{R=0}^{R=\infty} \\
&= 1.
\end{aligned}$$

For the background, we note that the density (integrand) b is zero for all $M < 0$, and show

$$\begin{aligned}
\int_{-\infty}^{\infty} b(M;\lambda) &= \int_{0}^{\infty}\lambda e^{-\lambda M}dM \\
&= \left[-e^{-\lambda M}\right]_{M=0}^{M=\infty} \\
&= 1.
\end{aligned}$$

Finally this lets us show that the probability density p given is properly normalised over $[-\infty, +\infty]$ because

$$\begin{aligned}
\int_{-\infty}^{\infty} p(M;f,\lambda,\mu,\sigma)dM &= f\int_{-\infty}^{\infty} s(M;\mu,\sigma)dM + (1-f)\int_{-\infty}^{\infty} b(M;\lambda)dM \\
&= f\cdot 1 + (1-f)\cdot 1 \\
&= 1.
\end{aligned}$$

Part (b)

In order to ensure that the fraction of signal in the restricted distribution for M remains f, we must normalise the signal and background separately over $[\alpha, \beta]$, we introduce these new restricted distributions as

$$s_r(M;\mu,\sigma) = \frac{s(M;\mu,\sigma)}{\int_\alpha^\beta s(M;\mu,\sigma)dM} \qquad b_r(M;\lambda) = \frac{b(M;\lambda)}{\int_\alpha^\beta b(M;\lambda)dM}$$

We then compute the relevant integrals:

$$\begin{aligned}\int_\alpha^\beta s(M;\mu,\sigma)dM &= \int_{-\infty}^\beta s(M;\mu,\sigma)dM - \int_{-\infty}^\alpha s(M;\mu,\sigma)dM \\ &= \frac{1}{2}\left[1+\operatorname{erf}\left(\frac{\beta-\mu}{\sigma\sqrt{2}}\right)\right] - \frac{1}{2}\left[1+\operatorname{erf}\left(\frac{\alpha-\mu}{\sigma\sqrt{2}}\right)\right] \\ &= \frac{1}{2}\operatorname{erf}\left(\frac{\beta-\mu}{\sigma\sqrt{2}}\right) - \frac{1}{2}\operatorname{erf}\left(\frac{\alpha-\mu}{\sigma\sqrt{2}}\right)\end{aligned}$$

$$\begin{aligned}\int_\alpha^\beta b(M;\lambda)dM &= \int_{-\infty}^\beta b(M;\lambda)dM + \int_{-\infty}^\alpha b(M;\lambda)dM \\ &= 1 - e^{-\lambda\beta} - (1 - e^{-\lambda\beta}) \\ &= e^{-\lambda\alpha} - e^{-\lambda\beta}\end{aligned}$$

The properly normalised probability density function for $M \in [\alpha, \beta]$ is then

$$p_r(M;\boldsymbol{\theta}) = \frac{2f}{\operatorname{erf}\left(\frac{\beta-\mu}{\sigma\sqrt{2}}\right) - \operatorname{erf}\left(\frac{\alpha-\mu}{\sigma\sqrt{2}}\right)} s(M;\mu,\sigma) + \frac{1-f}{e^{-\lambda\alpha} - e^{-\lambda\beta}} b(M;\lambda) \quad (1)$$

where $s(M;\mu,\sigma)$ and $b(M;\lambda)$ are as given on the sheet. Note that the value for $p_r(M)$ is zero when $M \notin [\alpha, \beta]$.

Part (c)

To implement the expressions for the probability density function, I created the following function in the `mixed_pdf_tools` module.

```
1     def pdf_norm_expon_mixed(x, f, la, mu, sg, alpha, beta):
2         pdf_s = norm.pdf(x, loc=mu, scale=sg)
3         pdf_b = expon.pdf(x, loc=0, scale=1 / la)
4         weight_s = (2 * f) / (
5             erf((beta - mu) / (sg * np.sqrt(2)))
```

```
6               - erf((alpha - mu) / (sg * np.sqrt(2)))
7           )
8           weight_b = (1 - f) / (np.exp(-la * alpha) - np.exp(-
    la * beta))
9           return (weight_s * pdf_s) + (weight_b * pdf_b)
```

Listing 1: Function implementing pdf for part (c).

The function takes in the argument `x`, the four parameters `f, la, mu, sg` corresponding to f, λ, μ, σ respectively, and `alpha, beta` which represent the support $[\alpha, \beta]$. The function first initialises two variables `pdf_s` and `pdf_b` which compute the signal and background parts of the density as specified by $s(.)$ and $b(.)$, without any domain restriction. These use the methods `norm.pdf` and `expon.pdf` from the `numba_stats` package. Next we compute the coefficients of these terms as specified in 1, which incorporates the signal-to-background ratio f as well as the normalisations for the interval restriction. These are stored in `weight_s` and `weight_b` Finally we return the weighted sum of the two pdf evaluations to give the mixed pdf value at `x`.

We also check this function is properly normalised in the interval $[5, 5.6]$ using the following code

```
1 def check_normalisation(pdf, lower, upper):
2     integral_value, abserr = quad(pdf, lower, upper)
3     return integral_value
```

Listing 2: Function checking the normalisation of the pdf for part (c).

This code uses the `quad` method from `scipy.integrate` to integrate any function `pdf` over the given range specified by `lower`, `upper`. Executing the script `solve_part_c.py` prints the output of this function for three different parameter settings with the interval $[5, 5.6]$; the outputs in the terminal should all read as approximately unity (within machine precision).

Part (d)

In Figure 1 we see the underlying probability density function for M, which we will sample from later.

Note that the blue and orange lines, labelled as 'density components' are the plots of the probability density functions for M for the signal and background events respectively. However, to make the plot more intuitive, they have been multiplied by the respective proportion of that type of event in the total population. This means that technically neither is a probability density

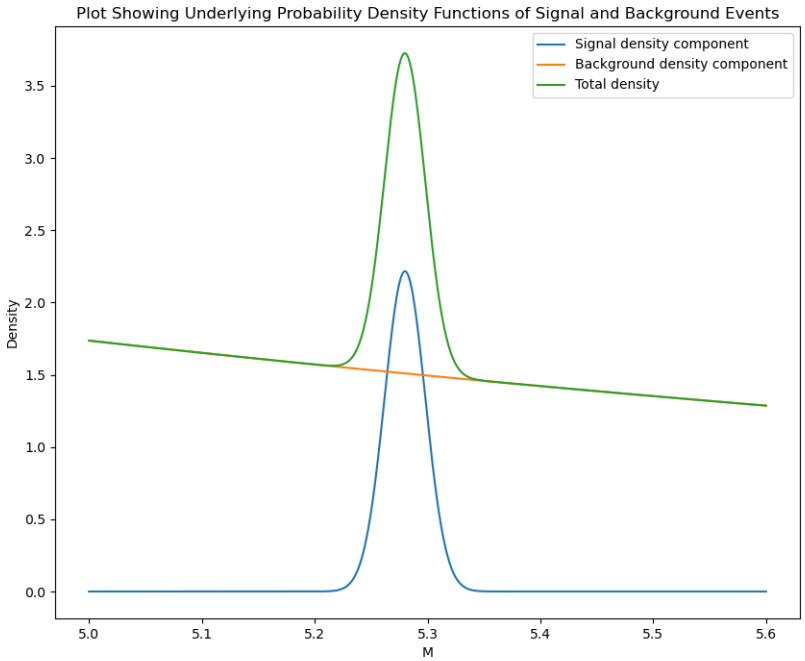

Figure 1: The true underlying distribution of M [generated in Python using Matplotlib].

function, however their sum is the true mixed density for the population. The area under either of the component functions limited to an interval smaller than $[5, 5.6]$ is precisely the probability that an event randomly sampled from the population will be of that kind, and lie within said interval.

Part (e)

In part (e), a large sample is generated from the true distribution, and then the sample is plotted alongside an estimate for the sample's underlying distribution. Originally, rejection sampling was used to create the sample, but this was found to be quite slow, a problem which was exacerbated in the simulation studies. Instead, an inverse CDF method was used to generate the sample. The inverse CDF method involves generating a sample of points

uniformly distributed in $[0, 1]$, and then transforming these by the percentage point function (the inverse of the cumulative density function). This achieves a sample which should theoretically be distributed according to the original probability density function.

This can easily be seen by letting X have distribution $U(0, 1)$, and transforming $Y := F^{-1}(X)$ where F is the cdf of the distribution we wish to sample from. Then clearly Y has the desired distribution because for any $c \in (-\infty, +\infty)$,

$$\begin{aligned} P(Y \in (-\infty, c)) &= P(F^{-1}(X) \in (-\infty, c)) \\ &= P(X \in (0, F(c))) && F \text{ is monotonic and increasing} \\ &= F(c) && X \text{ is uniformly distributed} \end{aligned}$$

Hence Y has cdf F.

In this case the cumulative distribution for the true density p is intractable, and thus we have to estimate it by sampling it at high granularity using the below block of code.

```
# Compute CDF values across interval [5.0, 5.6].
cdf_mixed_approx_sb = cdf_norm_expon_mixed(
    input_space, 0.1, 0.5, 5.28, 0.018, 5.0, 5.6
)
quantiles_sb = np.linspace(0.0, 1.0, int(10**GRANULAR))
cdf_mixed_inv_sb_01 = np.zeros(int(10**GRANULAR))
# For each quantile in [0.0, 1.0], find how far into the
    range [5.0, 5.6] you
# need to go in order for the cdf to equal the quantile.
for quantile_idx, quantile in enumerate(quantiles_sb):
    cdf_mixed_inv_sb_01[quantile_idx] = (10 ** (-GRANULAR)) *
    np.sum(
        cdf_mixed_approx_sb < quantiles_sb[quantile_idx]
    )
# Transform into the interval [5.0, 5.6].
CDF_MIXED_INV_APPROX_SB = 5.0 + 0.6 * cdf_mixed_inv_sb_01
```

Listing 3: Code block producing array of values for inverse CDF of p (e).

We then model the density function of the sample parametrically using a model of the same form as the true distribution p. We estimate the four parameters by using a binned maximum likelihood procedure: first separating

the data into 100 bins, and then find maximum likelihood estimates for the parameters based on the binned data. This is done using the `iminuit` Python package, which outputs the Hesse error matrix for the estimates as well as the fitted values.

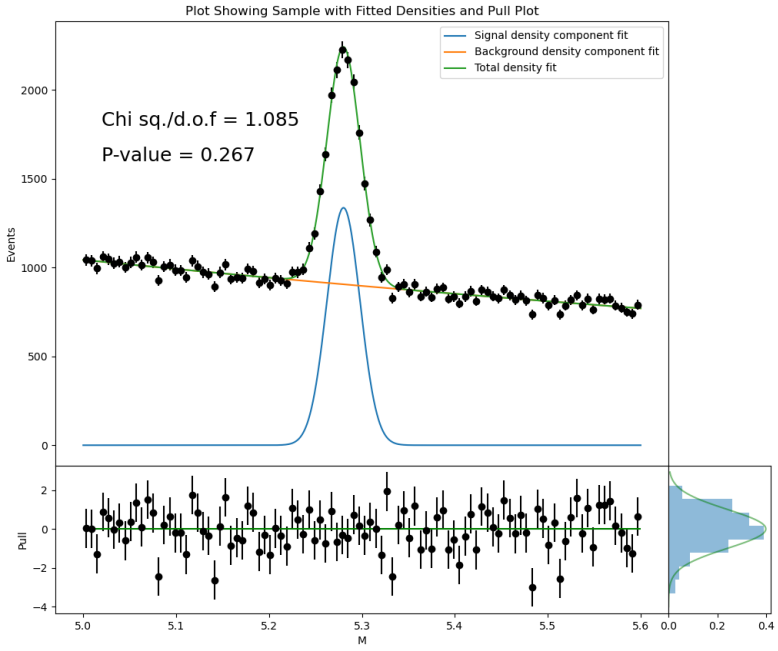

Figure 2: The true underlying distribution of M [generated in Python using Matplotlib].

Section B