Principle of data science coursework

Alberto Plebani
ap2387

1 Preliminary information

I run the entire code on a Ubuntu-22.04 WSL on my Huawei MateBook 16 laptop with Windows 11. To run the code I used Anaconda 23.9.0. I have allocated 8 GBs of RAM to the WSL, half of the total laptop's available RAM. The laptop has an AMD Ryzen 5 5600 Hz CPU with Radeon Graphics.

- Part c took 0.2 seconds to run, provided the plots aren't displayed. If the –plots is applied, the time rises up to 4.9 seconds.
- Part d took 0.48 seconds to run without the –plots flag, whereas it takes 2.5 seconds to run with the flag.
- Part e takes 0.8 seconds and 2.8 seconds without and with the flag, respectively.
- Part f takes 13 minutes if the –fit option is selected, whereas if the data is already saved and just needs to be loaded it just takes 0.35 seconds to generate the plots.
- Similarly, part g takes roughly 32 minutes to run with the –fit flag on, whereas it takes only 1.7 seconds to generate the plots.

Therefore, running all the steps require less than an hour.

2 Section a

In this section I will be presenting the work done on parts a-e.

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 $M \in [-\infty, \infty]$.

[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.

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 $f + 1 - f = 1$. $\square$

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) $F(X) = \int\limits_{\infty}^{X} f(X')dX'$. 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 $\boldsymbol{\theta} = (f, \lambda, \mu, \sigma)$ uniformly distributed in a specified range of values, and then I numerically evaluated the integral of 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.

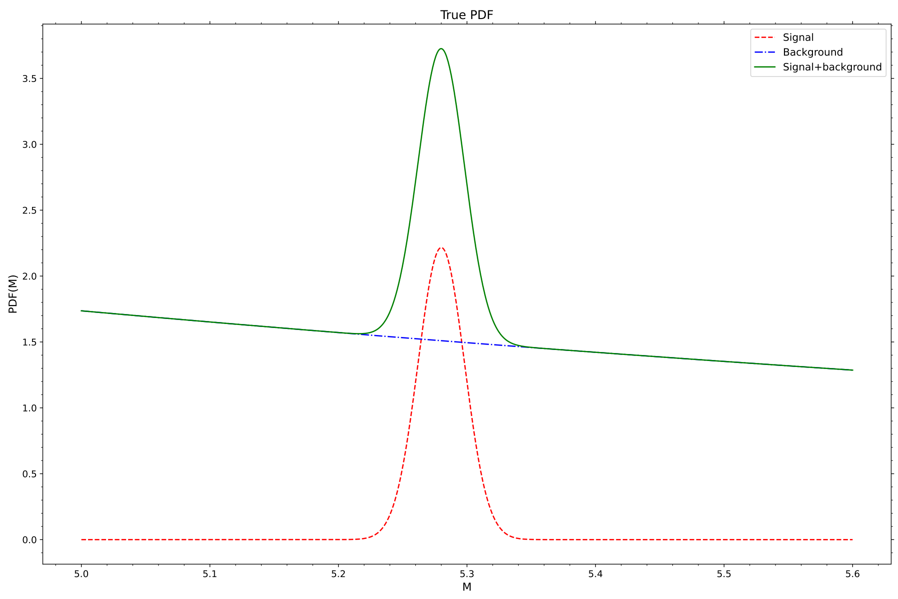

Figure 1: Plot displaying the true PDF alongside the distributions for signal and background, each rescaled by their relative fraction (10% signal, 90% background).

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. The values obtained are displayed in Table 2, and we can see how all the estimates are in agreement with the true values. The ML estimate for the PDF can be seen in Figure 2. As expected we can see that the orange and green distributions are very similar, once again ensuring the fairness of the fit.

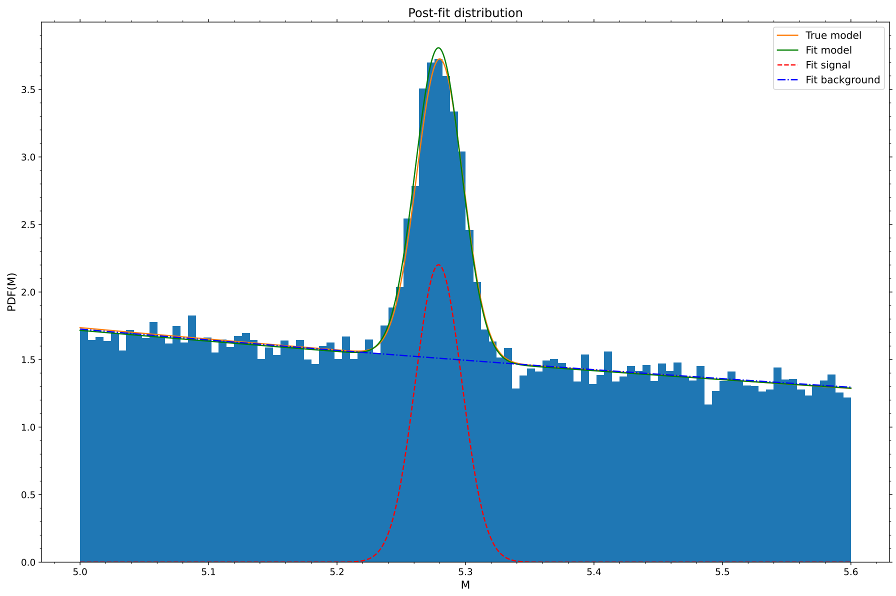

Figure 2: Plot displaying the generated data alongside the true distribution (orange) and the distribution of the best-fit model for signal (red), background (blue) and combined (green).

Parameter	MLE	Uncertainty	True value
f	0.0965	0.0024	0.1
μ	5.2799	0.0005	5.28
σ	0.0180	0.0005	0.018
λ	0.495	0.029	0.5

Table 2: MLE for the four parameters. We can see that for all parameters there is a good agreement between the MLE and the true value.

3 Section b

In this section I will be presenting the work done on parts f and g

f) The goal of this part is to estimate how much data is needed to discover the signal 90% of the times. For "discovery" we consider the HEP discovery limit, meaning 5σ of a normal distribution away from the null hypothesis, which requires a p-value smaller than 2.9×10^{-7}.

For null hypothesis (H_0) we consider the background-only hypothesis, meaning we set $f = 0$, having only an exponentially decaying distribution with 1 free-floating parameter, λ. The alternate hypothesis (H_1) is the hypothesis that both signal and background are present in our data. For this hypothesis we have 4 free-floating parameters, which are f, μ, σ and λ, respectively the fraction of signal, the central value of the signal Gaussian distribution with its variance, and the decay constant λ for the background distribution.

As a test statistics I used the likelihood ratio, following Neyman-Pearson lemma

(Equation (5))

$$T = -2\ln\left(\frac{L(x|H_0)}{L(x|H_1)}\right) \tag{5}$$

This test statistics is more powerful than the χ^2, and it holds the property, thanks to Wilks' theorem, of being asymptotically distributed as a χ^2 distribution with m degrees of freedom, where m is the difference between the number of free parameters between denominator and numerator. Therefore, it is trivial to evaluate the p-value for this hypothesis, which is given by Equation (6), where F is the CDF for the χ^2 distribution[2].

$$p = 1 - F_{\chi^2}(T; dof) \tag{6}$$

The significance Z of the test, meaning how many Gaussian standard deviations away we are from the hypothesis H_0, can be obtained using the percent point function (PPF, F^{-1}), which is the inverse of the CDF function, as per Equation (7).

$$Z = F^{-1}_{\chi^2}(1-p) \tag{7}$$

Therefore, the goal of this exercise is to evaluate the size of the generated data for which the discovery rate is above 90%. In order to do, I have performed the following steps

i) Select number of toy experiments. I have chosen 1000, meaning we want at least 900[3] experiments that give a significance greater than 5σ

ii) Select dataset size. The dataset size tells how many points are generated with the AR method for each toy experiment. I have tested different values, ranging from 50 to 2000, and then zooming in the values that gave a discovery rate close to 90%. In the final version of the code not all these values are displayed, and only the significant ones are kept in order to speed up the code, which already takes ~ 13 minutes to run.

iii) For each dataset size:

- Evaluate the test statistics T (Equation (5)). In order to do so, the negative-log likelihoods for both the null hypothesis and the alternate hypothesis are minimised, and T is evaluated as the difference between the two minima. If either one or the two fits fails, meaning it is not converging, the T value is not saved. In order to minimise the number of fails, the `iterate` option of `Minuit.migrad` was set to 100. This significantly slows the code down, but it reduces the number of failed fits, thus giving us more statistics.

[2] $F(x;k) = \dfrac{\gamma\left(\frac{k}{2}, \frac{x}{2}\right)}{\Gamma\left(\frac{k}{2}\right)}$, where k is the number of degrees of freedom (dof)

[3] This is not technically correct, since sometimes the fit fails, and therefore the failed values need to be removed from the counting. The actual number of toy experiments that give us a discovery rate above 90% is then $0.9 \times (1000 - N_{\text{failed}})$

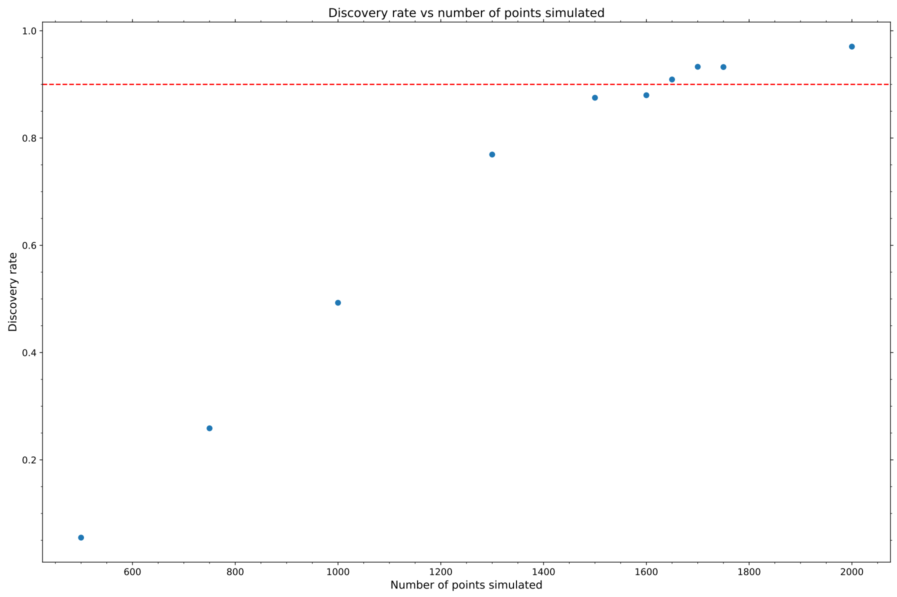

Figure 3: Discovery rate for each dataset size, in the 500-2000 range.

- Evaluate p-value. Using Equation (6), the p-value is evaluated, using the χ^2 approximation with $m = 3$ degrees of freedom, where $m = N_{free}^{H1} - N_{free}^{H0} = 4 - 1 = 3$, where N_{free}^{H} refers to the number of free-floating parameters of the hypothesis H.
- Evaluate the significance, using Equation (7).
- Evaluate the discovery rate, as the ratio between number of converging fits with $Z \geq 5\sigma$ and the number of converging fits.

The plot displaying the discovery rate for each dataset size can be found in Figure 3. We can see how there are some fluctuations, due to some fits failing in reaching a convergence. Nonetheless, it is clear that the greater the size of the dataset, the higher the discovery rate will be, because with more data points it will be easier to fit the correct function. The fits that fail are always for the alternate hypothesis, and this number ranges from 145 with 500 data points to 19 for 2000.

Overall, a 90% discovery rate is obtained when the size is of the order or 1600 points, with the discovery rate for 1650 points being 0.91.

g) Finally, for this part of the coursework the PDF in Equation (4) was modified, with a second Gaussian signal being added. The PDF then takes the form in Equation (8), where $s_i(M; \mu_i, \sigma)$ are two Gaussian centred in μ_i with same variance σ^2 representing the two signal distributions, each carrying a fraction f_i of the total PDF.

$$\text{pdf}(M; \boldsymbol{\theta}, \alpha, \beta) = f_1 \cdot s_1(M; \mu_1, \sigma) + f_2 \cdot s_2(M; \mu_2, \sigma) + (1 - f_1 - f_2) \cdot b(M; \lambda) \qquad (8)$$

The same procedure applied to the PDF in Equation (4) was carried out here, with each component being normalised separately by dividing by the CDF in the range $[\alpha, \beta]$, so

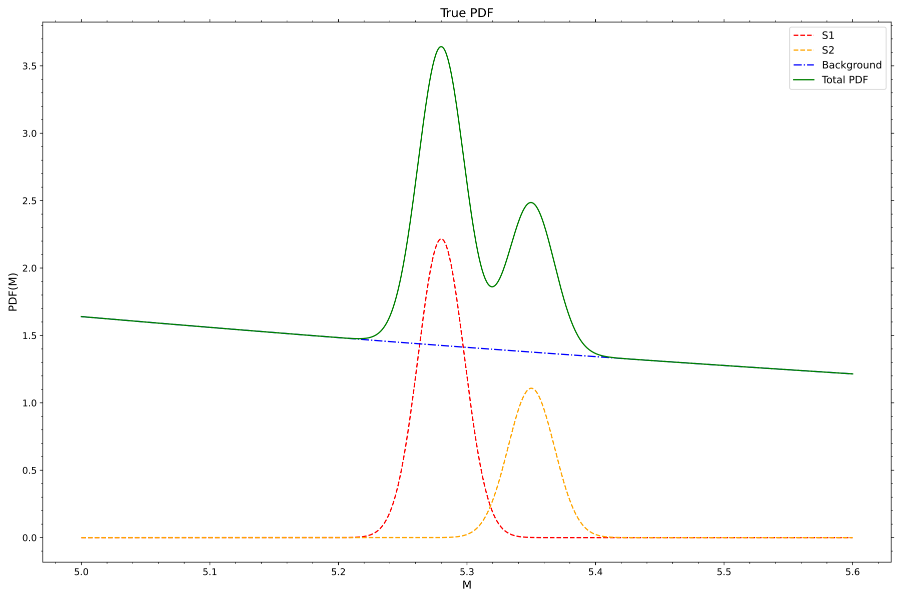

Figure 4: Plot displaying the true PDF alongside the distributions of the two signals and the background, each rescaled by their relative fraction (10%, 5%) and 85%.

Parameter	MLE	Uncertainty	True value
f_1	0.1017	0.0023	0.1
f_2	0.0499	0.0019	0.05
μ_1	5.2796	0.0005	5.28
μ_2	5.3495	0.0009	5.35
σ	0.0186	0.0004	0.018
λ	0.487	0.030	0.5

Table 3: MLE for the siz free-floating parameters. We can see that for all parameters there is a good agreement between the MLE and the true value.

that the integral of the PDF is $f_1 + f_2 + (1 - f_1 - f_2) = 1$.

Setting as true values for the PDF $f_1 = 0.1$, $f_2 = 0.05$, $\mu_1 = 5.28$, $\mu_2 = 5.35$, $\sigma = 0.018$ and $\lambda = 0.5$, the true PDF of Equation (8) has the shape represented in Figure 4, where we can clearly see two separate peaks, with a very small overlap only at the tails. Before testing the discovery rate, I checked a single fit for this model to assure the fairness of the AR method, and I obtained the MLE displayed in Table 3. Once again, we have a good agreement for all the six MLE, as we can also see from the plot in Figure 5. Finally, I moved on to the evaluation of the discovery rate. This time, the null-hypothesis H_0 is that we have only one signal, therefore either f_1 or f_2 must be 0, whereas the alternate hypothesis is that $f_1,\ f_2 \neq 0$, meaning we have two different signals. The fit for the null-hypothesis was then made to the PDF in Equation (4), because it is a PDF with only one signal, whereas for H_1 the PDF in Equation (8) was used.

The same procedure of exercise f) was used, except this time the dataset size range

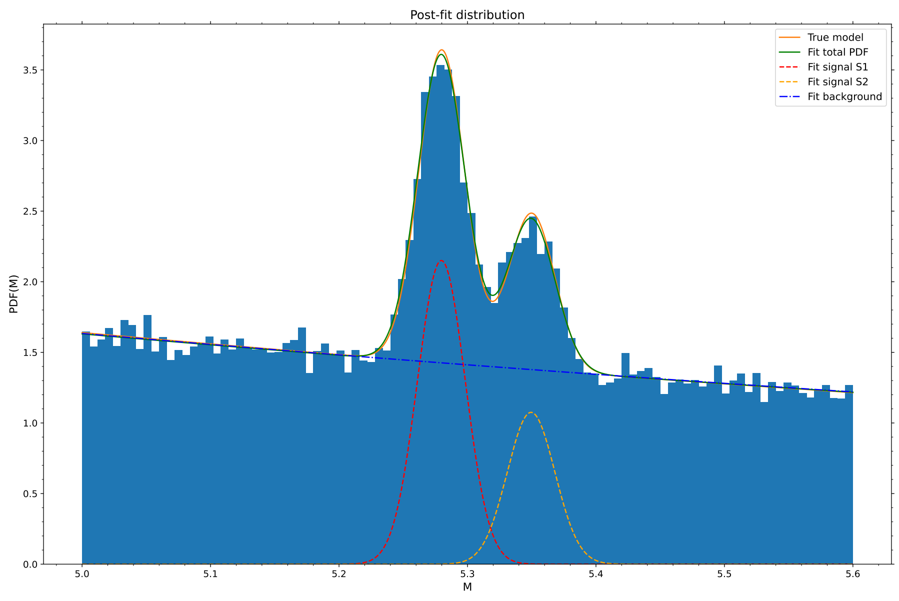

Figure 5: Plot displaying the generated data alongside the true distribution (orange) and the distribution of the best-fit model for the two signals (red and yellow), background (blue) and combined (green)

was increased in order to reach a discovery rate of 90%, because this time it was harder to discard H_0 due to the two signals being close. The plot displaying the different values of discovery rate can be found in Figure 6. We observe a similar pattern to what was observed in part f) (Figure 3), with the only difference being in the numbers being much greater. In detail, for 6000 generated points the discovery rate is 0.89%, meaning that more than 6000 events must be generated in order to have a 90% discovery rate. Once again, this number takes into account the failed fits, which ranged from 104 for 2000 data points to 19 for 10000 points, with the majority of fails being from H_0 fits.

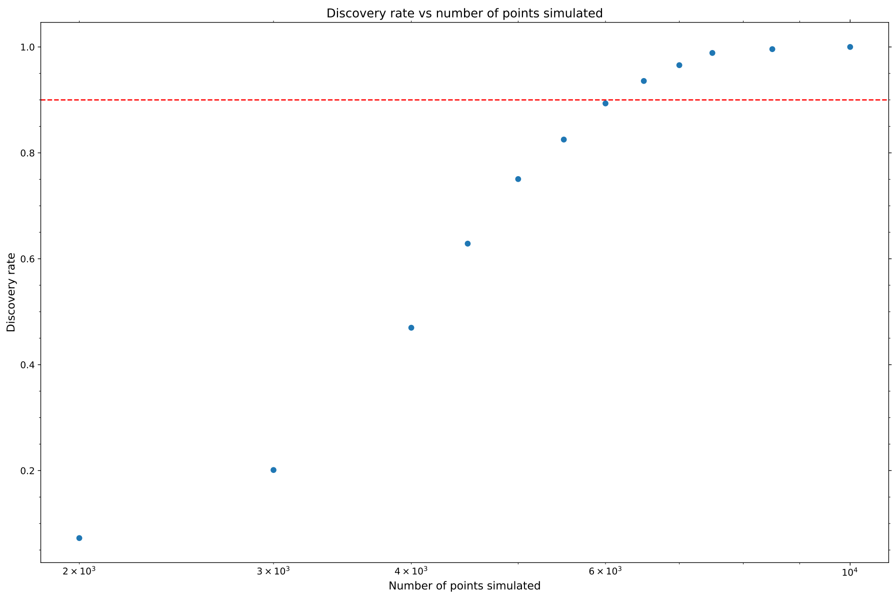

Figure 6: Discovery rate for each dataset size, in the 2000-10000 range.