Astro-statistics and Cosmology - Homework 1

Francesco Amadori

October 19, 2024

1 Homework 1

Problem: *Consider the coin tossing example, discussed in the first lecture. Simulate 1000 tosses of the coins, setting $H = 0.3$. Consider a uniform prior and update the posterior at each toss. Plot the resulting posterior after 1, 50, 100, 300, 700, 1000 tosses. Repeat the simulated experiment by setting a Gaussian prior centered in $H = 0.5$, with standard deviation $= 0.1$. Do both posteriors converge a similar distribution in the end? What does that mean? Which posterior converges faster and why?*

The first thing I made was to import properly the functions

```
import numpy as np
from numpy import array, random, linspace, pi, exp, sqrt, zeros_like, arange, argmax
from matplotlib import pyplot as plt, colormaps as cm
```

Then I defined the functions that I will use for this assignment.

```
# Likelihood calculation for this problem
def likelihood_prior_calculation(
        distribution_H, random_tossesf,  total_amount, center, std_dev
):
    # Define the amount of heads for each series considered
    heads_tosses = sum(random_tossesf[:total_amount])
    # Likelihood calculation
    likelihoodf = (
        distribution_H**heads_tosses *
        (1 - distribution_H)**(total_amount - heads_tosses)
    )
    # Gaussian Prior calculation (could have used scipy,
    # but I preferred to explicit the calculation)
    gaussian_priorf = (
            (1 / (std_dev * sqrt(2 * pi))) *
            exp(-0.5 * ((distribution_H - center) / std_dev)**2)
    )
    return likelihoodf, gaussian_priorf


# Normalize function
def normalize(arr):
    return (arr - min(arr)) / max(arr)
```

The first one ("likelihood_prior_calculation") is the one that calculates the likelihood $\mathfrak{L}$ and the gaussian prior Π_G for this problem with the formulas:

$$\mathfrak{L} = H_d^R \left(1 - H_d\right)^{N-R} \tag{1}$$

$$\Pi_G = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(-0.5\frac{H_d-\mu}{\sigma}\right)^2} \tag{2}$$

Here is a brief description of the variables involved:

- H_d ("distribution_H" in the code) is the uniform distribution of H (head tosses probability) values I chose. As I will describe later, it goes from 0 to 1 and is made of 5000 equally spaced elements to have a more precise distribution;
- R ("heads_tosses" in the code) is the amount of head tosses from the simulated data. The generation of the simulated data will be explained better later. In the code, this amount is calculated by the sum of the elements of the "random_tossesf", that is an array containing 1 if the toss is head, 0 if it is cross;
- N ("total_amount" in the close) is the total tosses considered;
- $\mathfrak{L}$ ("likelihoodf" in the code) is the likelihood calculated;
- Π_G ("gaussian_priorf" in the code) is the Gaussian prior calculated for the second part of the homework;

The second function ("normalize") is the calculation made to normalize an array between 0 and 1 to show better the results. It uses the minimum and maximum of the array to normalize it with this formula:

$$arr_{normalized} = \frac{(arr - min\,(arr))}{max\,(arr)} \tag{3}$$

The next lines of codes concern the creation of variables that will be used in the tosses simulation and the series of tosses considered for the plots.

```
# ## MAIN ## #
# Define the amount of tosses
n = 1000
# Setting the probability value for having head
# in the simulation to H=0.3
H = 0.3
# Simulate n tosses with a probability of having head H
rng = random.default_rng()
random_tosses = rng.binomial(1, H, n)
# List of amount of tosses to be considered
valid_array_of_tosses = array([1, 50, 100, 300, 700, 1000])
# Total tosses to analyze the map
array_of_tosses = arange(n + 1)
```

The variables here defined are:

- n which is the number of total tosses;
- H that is the probability of tossing head in the binomial distribution;
- *random_tosses* that is the array containing the n tosses with the probability of having head as H. I used the "numpy.random.binomial" with arguments $(1, H, n)$. The function can be resumed as: "result of flipping a coin 1 time, tested n times, with a probability of having head 0.3". This means that the output is 1 array of n elements with value 0 (cross) or 1 (head);
- *valid_array_of_tosses* that defines the number of tosses we want to analyze and plot;
- *array_of_tosses* that is a distribution of the number of tosses from 0 to 1000 to plot the convergence:

The following lines define the plots' figures and colors so I will not explain them in detail.

```
# Create the inferno colormap with len of
# array_of_tosses colors
cmap = cm.get_cmap('inferno')
# Generate equally spaced colors
```

```
num_colors = len(array_of_tosses) + 100
colors = [cmap(i / num_colors) for i in range(num_colors)]
# Figure definition
fig1, ax1 = plt.subplots(1, 1, figsize=(10, 5))
fig2, ax2 = plt.subplots(1, 1, figsize=(10, 5))
fig3, ax3 = plt.subplots(1, 1, figsize=(10, 5))
```

The next section concerns creating the uniform distribution used as parameter space and defining the variables involved for the Gaussian prior distribution creation.

```
# Generate uniformly spaced distribution
# from 0-1 for H values
start_H = 0
end_H = 1
amount_H = 5000
H_distribution = linspace(start_H, end_H, amount_H)
# Sigma definition and H prior definition for part 2
H_priors = [0.5, 0.3, 0.7]
sigma = 0.1
# Arrays creation for the MAP analysis
map_no_prior = zeros_like(array_of_tosses, dtype=float)
dict_priors = dict()
dict_priors[H_priors[0]] = zeros_like(array_of_tosses, dtype=float)
dict_priors[H_priors[1]] = zeros_like(array_of_tosses, dtype=float)
dict_priors[H_priors[2]] = zeros_like(array_of_tosses, dtype=float)
```

The variables here defined are:

- *H_distribution* that is an array of 5000 equally spaced elements with values between 0 and 1. They are used to calculate the likelihood. The more H are tested, the more the distribution will be well defined;
- *H_priors* is an array containing the center of three Gaussian priors distributions centered at: $[0.5, 0.3, 0.7]$;
- *sigma* is the standard deviation of the Gaussian prior distribution;
- *map_no_prior* which is the array containing the $\hat{H}$ values to estimate the speed of convergence for a uniform prior distribution;
- *dict_priors* is a dictionary containing the same array but for the three different priors Gaussian distributions centered at the values of *H_priors*.

The next section is the core of the script. It is a for loop that iterates over the "array_of_tosses" variables and calculates both the likelihood and the posterior using the "likelihood_prior_calculation" function. Then it calculates the normalized posterior of problem part 1 with the formula:

$$P = \mathfrak{L} \tag{4}$$

and of the problem part 2 with the formula:

$$P = \mathfrak{L}\Pi_G \tag{5}$$

As explained before, I used the function "normalize" to normalize the array results.

```
# Iteration over the amount of tosses that must be explored
for tosses in array_of_tosses:
    for H_prior in H_priors:
        # Likelihood calculation
        likelihood, gaussian_prior = likelihood_prior_calculation(
            H_distribution, random_tosses, tosses, H_prior, sigma
        )
```

```
            # Posterior (in this case is the same of the likelihood,
            # no priors defined)
            posterior_part_1 = normalize(likelihood)
            # Calculation of posterior with a gaussian prior
            # distribution with standard deviation sigma
            # centered at H_prior
            posterior_part_2 = normalize(likelihood * gaussian_prior)
            # Assignment of H_hat for the current amount of tosses
            map_no_prior[tosses] = H_distribution[argmax(posterior_part_1)]
            dict_priors[H_prior][tosses] = H_distribution[argmax(posterior_part_2)]
            # Plot the posteriors distribution only if in the selected tosses
            if tosses in valid_array_of_tosses and H_prior == 0.5:
                # Plot pt1
                ax1.plot(
                    H_distribution, posterior_part_1,
                    color=colors[tosses], label=f'{tosses} tosses', alpha=0.7
                )
                # Plot pt2
                ax2.plot(
                    H_distribution, posterior_part_2, color=colors[tosses],
                    label=f'{tosses} tosses', alpha=0.7
                )
            #
```

The next lines plot the results: the distribution of H is on the x-axis while the normalized density is on the y-axis.

```
# Add properties to the plots
ax1.set_title('Posterior Distribution after N tosses (no priors)')
ax1.set_xlabel('H (Probability of heads)')
ax1.set_ylabel('Normalized density')
ax1.axvline(H, linestyle=":", color="g", label=f"H={H}", alpha=0.7)
ax1.legend()
ax1.grid(True)
fig1.savefig("./plot_1.png")
#
ax2.set_title(
    r'Posterior Distribution after N tosses (gaussian priors '
    r'$\mu={}$, $\sigma={}$)'.format(H_priors[0], sigma)
)
ax2.set_xlabel('H (Probability of heads)')
ax2.set_ylabel('Normalized density')
ax2.axvline(H, linestyle=":", color="g", label=f"H={H}", alpha=0.7)
ax2.legend()
ax2.grid(True)
fig2.savefig("./plot_2.png")
#
ax3.set_title("MAP for faster convergence")
ax3.set_xlabel('Number of tosses')
ax3.set_ylabel(r'$\hat{H}$')
ax3.plot(
    array_of_tosses, map_no_prior, color=colors[100],
    label=f"MAP uniform prior", alpha=0.7
)
ax3.plot(
    array_of_tosses, dict_priors[H_priors[0]], color=colors[200],
```

```
    label=f"MAP gaussian prior centered at {H_priors[0]}", alpha=0.7
)
ax3.plot(
    array_of_tosses, dict_priors[H_priors[1]], color=colors[500],
    label=f"MAP gaussian prior centered at {H_priors[1]}", alpha=0.7
)
ax3.plot(
    array_of_tosses, dict_priors[H_priors[2]], color=colors[800],
    label=f"MAP gaussian prior centered at {H_priors[2]}", alpha=0.7
)
ax3.axhline(H, linestyle=":", color="g", label=f"H={H}", alpha=0.7)
ax3.legend()
ax3.grid(True)
ax3.set_xscale("log")
fig3.savefig("./plot_3.png")
```

The first two pictures below are the results of the two parts of the homework 1. Both the posteriors converge at the same result: the value that maximizes the likelihood (and so the posterior) is around $\hat{H} = 0.3$, which is the value of the probability of having a head toss used to generate the simulated data. The posterior becomes sharper and more concentrated around this value the more tosses we consider. This means that more data not only helps converge faster to the true value but also reduces uncertainty about the estimation itself. Both the uniform prior and the Gaussian prior centered at $H = 0.5$ lead to posterior distributions that converge to $H = 0.3$ after enough tosses. This is what we expect because, in general, as more data are collected, the likelihood begins to dominate over the prior, making the posterior affected more by the likelihood than the prior itself. In the Gaussian prior case, the prior pulls the posterior toward the center of the distribution if the tosses considered are not enough. This is because the prior provides incorrect initial information, and it takes more data to overcome the effects of the prior, making the convergence slower w.r.t. the uniform one. Considering a prior with the value near to the true value will lead to a faster convergence. The third picture shows the MAP (Maximum a posterior estimation) of the four different prior distributions considered from 0 to 1000 tosses. It is easy to see that the convergence is faster for a uniform prior distribution and for a Gaussian prior distribution centered at the true value.

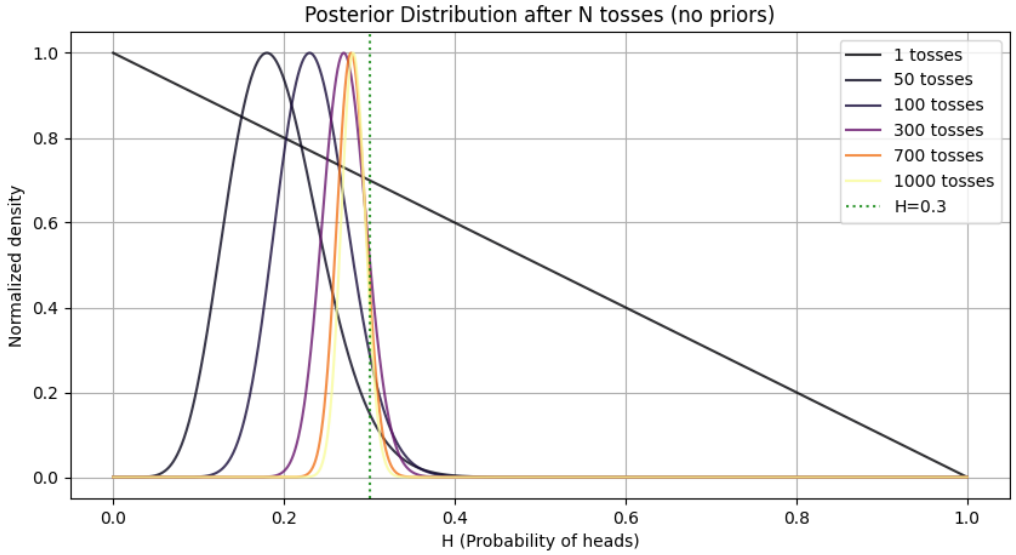

Figure 1: Normalized posterior distribution after $[1, 50, 100, 300, 700, 1000]$ tosses

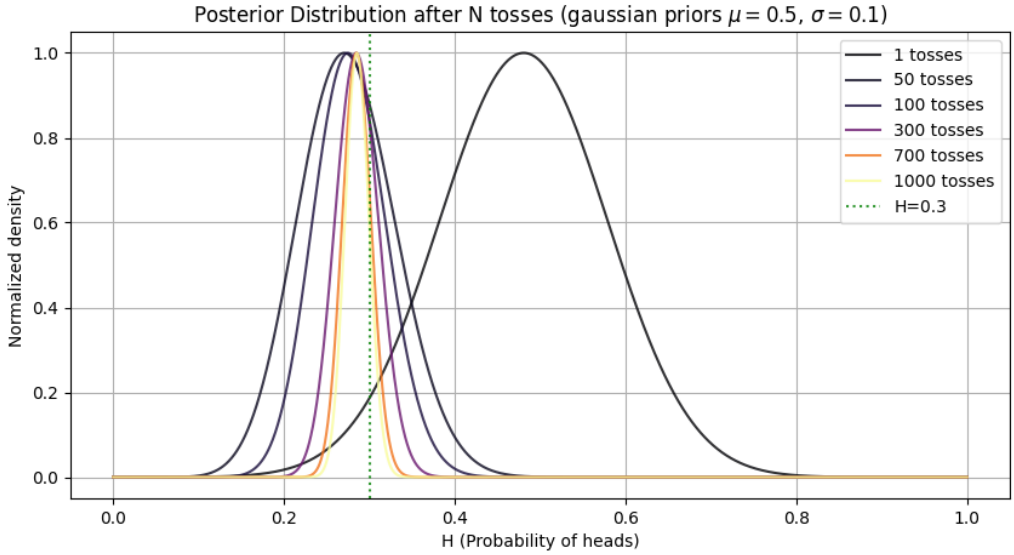

Figure 2: Normalized posterior distribution after $[1, 50, 100, 300, 700, 1000]$ tosses with a Gaussian priors centered in $H = 0.5$ and with $\sigma = 0.1$

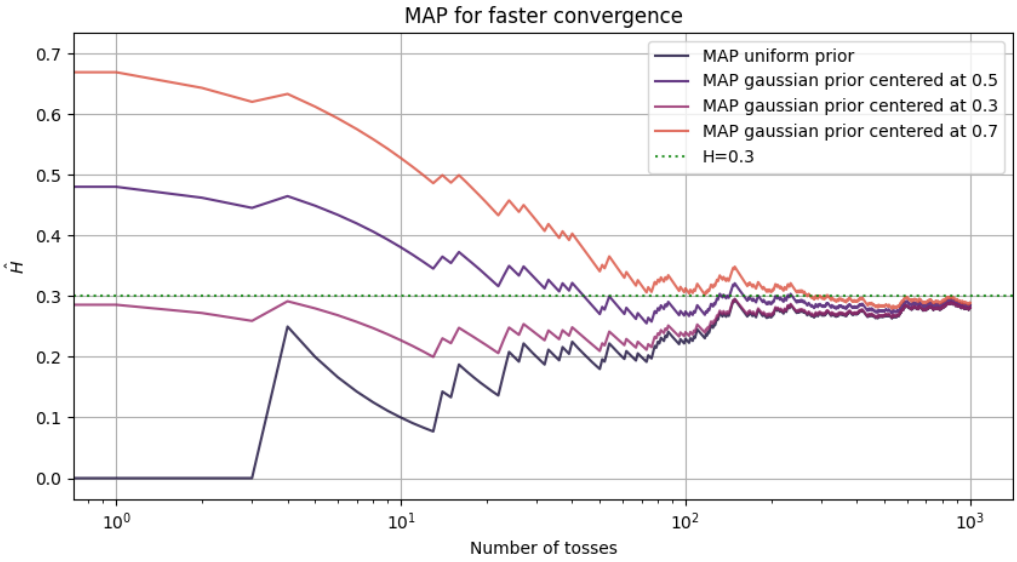

Figure 3: MAP ($\hat{H}$) values for a distribution from 0 to 1000 tosses with a Gaussian priors centered in $H = 0.5$, $H = 0.3$, $H = 0.7$ and with $\sigma = 0.1$

2 Homework 2

Problem: *Politician A makes a statement about some issue you knew nothing about before. Let's call such proposition S and assume your starting prior on S is uniform with 0.5 probability of S being either true or false. Update your probability of S being true, knowing that you trust Mr. A to tell the truth with probability prob$(A_T) = 4/5$. At this point Mr B - another politician - declares that he agrees with Mr A on S being true. You trust Mr. B much less, and believe that the probability of him to lie is prob$(B_T) = 3/4$. What is your final degree of belief in proposition S?*

To solve this problem we need to apply the Bayes Theorem that states, in general,:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} = \frac{\mathfrak{L}\Pi}{E} \tag{6}$$

The definitions are here explained (I will use only the statement S and the politician A for this example):

- $P(A|B)$ is the posterior and is the probability of event A given event B happened. In our case, this is what we want to find for the first part: the probability of statement S to be true considering that A is telling the truth, so: $P(S|A_T)$;
- $\mathfrak{L} = P(B|A)$ is the likelihood and is the probability of B given A happened. In our case, is the probability that politician A is telling the truth given the statement S as true, so: $P(A_T|S) = \frac{4}{5}$ because we trust the politician $(P(A_T|!S) = \frac{1}{5}$. **NB: the notation $P(!S)$ means the "opposite" probability of $P(S)$, so basically $P(!S) = 1 - P(S)$. I prefer to use this notation instead of $P(\overline{S}) = 1 - P(S)$ because it is easier for me to visualize the meaning. Anyway, I want to explicitly that:** $P(!S) = P\left(\overline{S}\right)$);
- $\Pi = P(A)$ is the prior, so the probability of A being true. In our case (at least in the first part) we assume that S could be true or false either because we know nothing so: $P(S) = 0.5$ and $P(!S) = 0.5$;
- $P(B) = E$ is the evidence, the probability that B is true. In our case is the probability that politician A is telling the truth without any condition, so we have:

$$P(A_T) = P(A_T|S)P(S) + P(A_T|!S)P(!S) = \frac{4}{5} \cdot 0.5 + \frac{1}{5} \cdot 0.5 = 0.5 \tag{7}$$

  This value is high because we trust politician A.

So now the probability that S is true giving A telling the truth is:

$$P(S|A_T) = \frac{P(A_T|S)\,P(S)}{P(A_T)} = \frac{\frac{4}{5} \cdot 0.5}{0.5} = 0.8 \tag{8}$$

Now politician B states the same as A but we do not trust B too much, so the probability of him telling the truth given S is $P(B_T|S) = \frac{1}{4}$. Now the probability of S is, in reality, the one derived from the calculation before:

$$P(\widetilde{S}) = P(S|A_T) = 0.8 \tag{9}$$

so $P(!\widetilde{S}) = 0.2$. In this second phase we want $P\left(\widetilde{S}|B_T\right)$ so:

$$P\left(\widetilde{S}|B_T\right) = \frac{P\left(B_T|\widetilde{S}\right)P(\widetilde{S})}{P(B_T)} = \frac{P\left(B_T|\widetilde{S}\right)P(\widetilde{S})}{P(B_T|\widetilde{S})P(\widetilde{S}) + P(B_T|!\widetilde{S})P(!\widetilde{S})} = \frac{\frac{1}{4} \cdot 0.8}{\frac{1}{4} \cdot 0.8 + \frac{3}{4} \cdot 0.2} = 0.57 \tag{10}$$

3 Homework 3

Problem: *You are tested for a dangerous disease named "Bacillum Bayesianum" (BB). You test positive to BB. You know that the general incidence of BB in the population is 1%. Moreover, you know that your test has a false negative probability of 5% (false negative: you have BB but the test scores negative), and a false positive rate also of 5% (false positive: you do not have BB, but the test scores positive). What is the probability that you have actually contracted BB?*
The same approach using the Bayes Theorem will be applied here. Here is the resume of the data:

Probability	**Meaning**
$P(B) = 0.01$	Probability of being positive
$P(!B) = 0.99$	Probability of being negative
$P(T\|B) = 0.95$	Probability of having a positive result in the test being positive
$P(T\|!B) = 0.05$	Probability of having a positive result in the test being negative
$P(!T\|B) = 0.05$	Probability of having a negative result in the test being positive
$P(!T\|!B) = 0.95$	Probability of having a negative result in the test being negative
$P(B\|T) =?$	Probability of being positive having a positive test result
$P(T) =?$	Probability of the test being positive independent from the status

Table 1: Definition of the involved probabilities

So the probability of being positive for real is:

$$P(B|T) = \frac{P(T|B)P(B)}{P(T)} = \frac{P(T|B)P(B)}{P(T|B)P(B) + P(T|!B)P(!B)} = \frac{0.95 \cdot 0.01}{0.95 \cdot 0.01 + 0.05 \cdot 0.99} = 0,161 \tag{11}$$