TP3 — Inference bayesienne (corrige)

Source : TP3/Bayesian-inference_corrige.ipynb — PDF compile (code execute).

1. Simple coin flip exercise

You have a coin that when flipped ends up head with probability p and ends up tail with probability $1−p$. The value of p is unknown.

Flipping the coin $\bf{n}$ times you end up with $\bf{k}$ heads. The objective of this exercise is to estimate the probability distribution of p using this data.

Credits:

• Panos Ipeirotis, https://www.behind-the-enemy-lines.com/2008/01/are-you-bayesian-or-frequentist-or.html

Import librairies

In [1] :

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt

import scipy.special as sc
from scipy.stats import beta # beta probability distribution function
```

Parameters of the exercise

In [2] :

```
k,n = 100,140 # number of heads and trials
```

> 1. What could be a coarse estimation of p?

Bayesian treatment: mathematical approach

We will see that, for this particular exercice, it is actually straightforward to access p. Here, p is not a value, it's a distribution and to compute it we start from the Bayes rule:

$$ P(p|data) = \frac{P(data | p) P(p)}{P(data)}, $$

where $P(data | p)$ is given by the Binomial distribution which determines the probability of observing k "heads" in n flip when the probability of success (head result) is given by p:

$$ P(data|p) = P(k|p,n) = \binom nk p^k (1-p)^{n-k}, \,\,\, \text{with the binomial coefficient: } \binom nk = \frac{n!}{k! (n-k)!}. $$

Now, for the prior distribution $P(p)$, a very convenient choice for this scenario (also known as the conjugate prior of the binomial) is the Beta distribution, $Beta(p|a,b)$ defined as:

$$ Beta(p|a,b) = \frac{1}{B(a,b)} p^{a-1}(1-p)^{b-1} $$

where B is the beta function (not to be mistaken with the beta distribution !) and is a normalization constant to ensure that the probability integrates to 1. A distribution $Beta(p|a,b)$ has a mean of $\frac{a}{a+b}$.

> 2. With some simple calculations, show that the posterior is of the same distribution family as the prior (Beta distribution) and that the posterior distribution comes down to:

$$ P(p|data) = Beta(p|k+a, n - k + b) $$

Thus the posterior of this problem depends of the data (k and n) and the two parameters (a and b) that defines the prior !

1.1. Prior and posterior distributions

The Beta distribution, for parameters a and b, is given by the function `beta.pdf(x,a,b)`

> 3. Plot the shape of the prior for different values of a and b.

> 4. Why could it be a good choice to choose a symmetric ($a=b$) prior?

1.1.a) Model 1: uniform prior

> 5. For Model 1, choose an uniform prior ($a_1=b_1=1$) and plot the corresponding posterior distribution.

> 6. Derive the MAP estimator and MMSE estimator (i.e. averaged p) for model 1. Is it consistent with your coarse estimation of p? How do they behave with the amount of data (k and n)?

1.1.b) Model 2: peaked prior

> 7. Now, for Model 2, plot the posterior distribution for a 'peaked' prior (e.g. $a_2=b_2=10$) and comment in comparison to Model 1.

> 8. Derive the MAP estimator and MMSE estimator (i.e. averaged p) for Model 2. Comment in comparison to Model 1.

1.1.c) Model 3: Very peaked prior

> 9. For Model 3, plot the posterior distribution for a 'peaked' prior (e.g. $a_3=b_3=100$) and comment in comparison to Models 1 and 2.

> 10. Derive the MAP estimator and MMSE estimator (i.e. averaged p) for model 1. Comment in comparison to Models 1 and 2.

1.2. Bayes factor

For comparing the plausibility of two models, we can resort to Bayes factors.

The Bayes factor is the ratio of the marginal likelihoods of the two models, that is, the likelihoods of two statistical models integrated over the prior probabilities of their parameters. Here instead of comparing models, we can now also compare hypotheses encoded in the form of priors.

The marginal likelihood is given by: $$ P(data) = \int P(data | p)P(p)dp $$

The Bayes factor K for comparing the plausibility of two models is: $$ K = \frac{P(data | \text{Model}_1)}{P(data | \text{Model}_2)} = \frac{\int P(data | p, \text{Model}_1)P(p | \text{Model}_1)dp}{\int P(data | p, \text{Model}_2)P(p | \text{Model}_2)dp}. $$

A value of K > 1 means that Model1 is more strongly supported by the data under consideration than Model2. The relative strength of a Bayes factor can be determined by consulting the interpretation tables.

For the binomial model (with a conjugate Beta prior) the Bayes factor is easily calculated:

$$ K = \frac{P(data | \text{Model}_1)}{P(data | \text{Model}_2)} = \frac{B(k+a_1,n-k+b_1)}{B(a_1,b_1)} \times \frac{B(a_2,b_2)}{B(k+a_2,n-k+b_2)}. $$

Here B is the beta function.

> 11. Calculate the Bayes factors corresponding to the comparison of the three models used above (uniform prior, 'peaked' prior and 'very peakd' prior). Hint: to ease numerical calculation it is better to consider the logarithm of the beta function `sc.betaln` (instead of the beta function `sc.beta` that can have very low values). Conclude on which model to choose.

1.3. Making predictions

In Bayesian inference we can use our probabilistic statements about the parameter from the posterior for making predictions. This is usually refered to as posterior predictive distribution.

This is the distribution of unobserved data, conditioned on observed data and prior. It follows the idea as the marginal likelihood discussed above (which can be seen as a prior predictive distribution), namely that we want to evaluate new data on our fitted model.

We can write the posterior predictive distribution for some new unseen data x, as the marginal of a distribution conditioned on observed data and prior:

$$ P(x |data) = \int P(x|p)P(p |data)dp $$

where $data$ refers to our training data, and x to some new unseen data.

> 12. For the beta-binomial model, compute $P(j | m,k,n,a,b)$ where j refers to the new number of Heads, m to the new number of total flips, k to the previous number of Heads, n to the previous number of total flips, and a and b to our prior hyperparameters.

$$ P(j | m,k,n,a,b) = \int P(j|m,p)P(p |k,n,a,b)dp= \binom{m}{j} \frac{B(k+j+a,n-k+m-j+b)} {B(a+k,b+n-k)} $$

> 13. Calculate the probability of flipping 2 heads out of 2 new flips with the model you selected as the best in the previous question > > 14. Calculate the probability of flipping 3 heads out of 5 new flips with the model you selected as the best in the previous question

> 15. What can you conclude about the quality of these predictions? How will the prediction probability behave with increasing number of new flips? Check by producing the plot of the prediction probability of any j with values of m between 1 and 200 for instance.

2. The lighthouse problem

The following is a classic estimation problem called the "lighthouse problem". The figure shows a set of receivers distributed at coordinates x_k along the shore and a lighthouse located at some position (α,β) offshore. The idea is that the coastline is equipped with a continuous strip of photodetectors. The lighthouse flashes the shore N times at some random angle θ_k. The strip of photodetectors registers the k^{th} flash position x_k, but the angle θ_k of the flash is unknown. Furthermore, the lighthouse beam is laser-like so there is no smearing along the strip of photodetectors. In other words, the lighthouse is actually more of a disco-ball in a dark nightclub.

The problem is how to estimate α given we already have β.

Credits:

• Data Analysis: Bayesian tutorial. D.S.Sivia.



Given the nature of the lighthouse emission it seems reasonable to assign an uniform density function to the angle θ_k, such that it is uniformly distributed between $\pm \pi/2$: $$ P(\theta_k|\alpha, \beta) = \frac{1}{\pi} $$

Now, what we really want is the density function for x_k which will tell us the probability that the k^{th} flash will be recorded at position $ x_k $. From basic trigonometry, we have the following:

$$ \beta \tan(\theta_k) = x_k - \alpha $$

Thus, after a transformation of variables, we obtain the following:

$$ P(x_k|\alpha, \beta) = \frac{\beta}{\pi \left[\beta ^2 +(x_k-\alpha)^2 \right]}. $$

This distribution is known as the [Cauchy distribution](https://en.wikipedia.org/wiki/Cauchy_distribution).

2.1. Data generation

First we'll generate some toy data that will be used in this exercise. We assume that the 'true' values of the parameters that we want to infer are $\alpha = \beta = 1$.

> 16. Run the cell below to generate N=250 values of the angle θ_k uniformely distributed between $-\pi/2$ and $+\pi/2$ and to calculate the corresponding x_k values. > > 17. Make a histogram of the θ_k and x_k values > > 18. Define the function $P(x_k|\alpha, \beta)$ and draw it for fixed $\alpha = \beta = 1$ values.

2.2. Bayesian inference

Given a set of observed measurements $\{x_k\}$ we can infer the position along the shore of the lighthouse, α, by the posterior pdf $P(\alpha|\{x_k\},\beta)$. Here, to simplify the problem, we assume that the distance to the coast, β, is known. Writing down the Bayes theorem we have:

$$ P(\alpha|\{x_k\},\beta) \propto P(\{x_k\}|\alpha,\beta) P(\alpha|\beta). $$

As the knowledge of β, without the data, tells us nothing new about the position α, let us assign a single uniform pdf to the prior:

$$ P(\alpha|\beta) = P(\alpha) \begin{cases} A & \alpha_\text{min} \leq \alpha \leq \alpha_\text{max} \\ 0 & \text{otherwise} \end{cases} $$

where A is a constant and α_min and α_max represent the limit of the coastline.

$P(\{x_k\}|\alpha,\beta)$ is the likelihood of the data that is just the product of the probabilities for obtaining the N individual observations x_k:

$$ P(\{x_k\}|\alpha,\beta) = \prod_{k=1}^N P(x_k|\alpha,\beta). $$

From the equations above one can write the logarithm of the posterior distribution as:

$$\mathcal{L}(\alpha)= \log P(\alpha|\{x_k\},\beta) = \text{constant}- \sum_{k=1}^N \ln \left[\beta^2 + (x_k-\alpha)^2 \right] $$

The MAP estimate of the position α is given by the maximum of the posterior pdf. Taking the first derivative gives us the equation we would have to solve:

$$ \frac{d \mathcal{L}}{d \alpha} = 2 \sum_{k=1}^N \frac{x_k-\alpha}{\beta^2+(x_k-\alpha)^2}=0. $$

Unfortunately, there is no easy way to solve for the optimal α for this equation. However, nothing can prevent us from addressing this problem numerically.

For this we will use brute force to evaluate $\mathcal{L}(\alpha)$ for a range of possible values α:

> 19. Implement the function $\mathcal{L}(\alpha)$. We can ignore the constant in this expression as we will deal with it below, > > 20. Calculate $\mathcal{L}(\alpha)$ with the available data and for a grid of values in the range $3 \leq \alpha \leq 3$. Then subtract the maximum value of $\mathcal{L}(\alpha)$ to $\mathcal{L}(\alpha)$ > > 21. Finally plot the posterior distribution $P(\alpha|\{x_k\},\beta) = \exp(\mathcal{L}(\alpha))$ as a function of α. > > 22. Print out the maximum posterior estimate

In [3] :

```
# Donnees phare (cellule setup — le sujet d'origine utilisait x_samples/N/beta sans les
definir)
np.random.seed(0)
N = 250
beta = 1.0
def generate_data(N, alpha=1.0, beta=1.0):
theta = np.random.uniform(-np.pi/2, np.pi/2, size=N)
x = alpha + beta*np.tan(theta)
return theta, x
theta_samples, x_samples = generate_data(N, alpha=1.0, beta=1.0)
```

In [4] :

```
        # 19)
        def logposterior(alpha,x_samples,beta):
        return -1 * np.sum(np.log(np.power(x_samples-alpha,2)+beta**2))

        # 20)
        aa = np.linspace(-3,3,100)
        llog = []
        for a in aa:
        llog += [logposterior(a,x_samples,beta)]

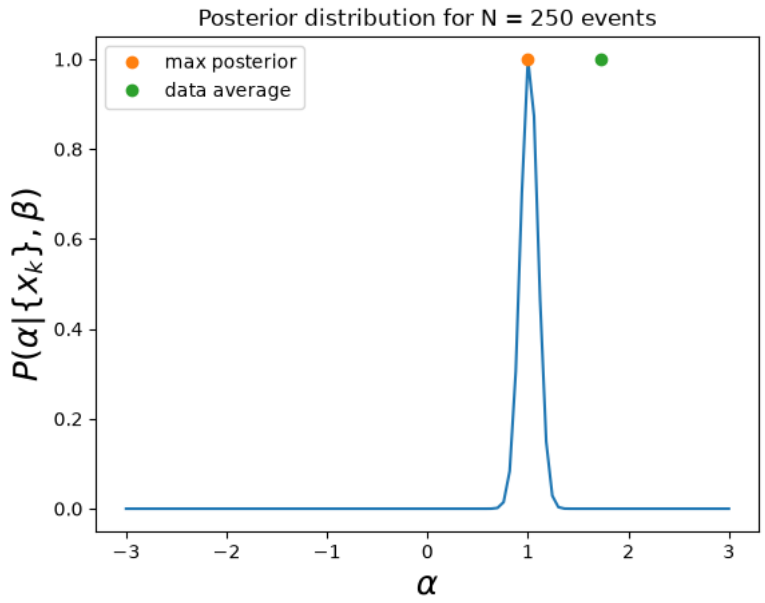
        llog = np.array(llog)
        llog -= np.max(llog) # Subtract max value

        # 21)
        posterior = np.exp(llog)
        plt.plot(aa,posterior)
        plt.xlabel(r'$\alpha$',fontsize=18)
        plt.ylabel(r'$P(\alpha|\{x_k\},\beta)$',fontsize=18)
        plt.title('Posterior distribution for N = %d events' % N)

        # 22)
        ymax = np.max(posterior)
        xpos = np.where(posterior == ymax)
        alpha_max = aa[xpos]
        plt.plot(alpha_max,ymax,'o',label='max posterior')
        plt.plot(x_samples.mean(),ymax,'o',label='data average')
        plt.legend()

        print('maximum posterior estimate = %.3f' % alpha_max[0])
```

```
maximum posterior estimate = 1.000
```




```
<Figure size 640x480 with 1 Axes>
```

2.3. Comparing maximum posterior estimate and plain averaging

The best estimator for a mean μ of a normal distribution is simply given by the sample average. As in the limit of large numbers all distributions should behave as a normal distribution (this is the Central Limit Theorem) why didn't we just take the average of the $\lbrace x_k \rbrace$ and use that to estimate α?

We will see here that in the present case, this is a bad idea!

> 23. Run the full procedure above 10 times. At each trial generate a new sample, then compare the maximum posterior estimate with the sample average. Do these result seem to agree ? > > 24. Now let's explore this

problem by performing a large number of trials (say 100), Make a plot showing, as a function of the trial number, the posterior maximum estimate and the sample average with its standard deviation. What is your conclusion ?

In [5] :

```
        # 23)

        Ntrial = 10

        def runtrial(do_plot=True):

        # generate new data
        theta_samples, x_samples = generate_data(N)

        # Calculate posterior
        aa = np.linspace(-3,3,100)
        llog = []
        for a in aa:
        llog += [logposterior(a,x_samples,beta)]

        llog = np.array(llog)
        llog -= np.max(llog) # Subtract max value

        posterior = np.exp(llog)

        ymax = np.max(posterior)
        xpos = np.where(posterior == ymax)
        alpha_max = aa[xpos]

        sample_mean = x_samples.mean() # mean
        sample_mean_error = x_samples.std() / np.sqrt(N)

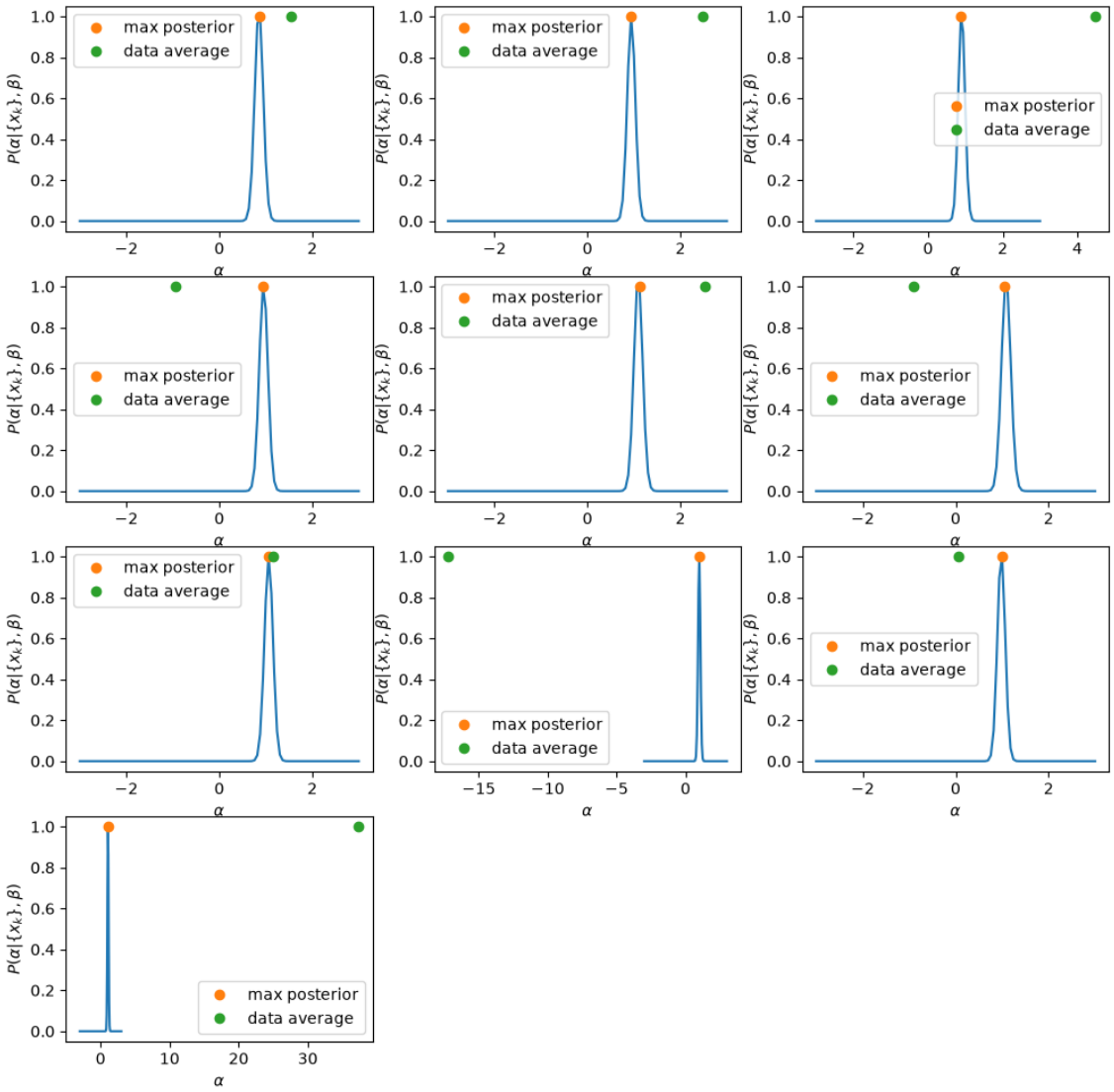
        if do_plot:
        plt.plot(aa,posterior)
        plt.xlabel(r'$\alpha$')
        plt.ylabel(r'$P(\alpha|\{x_k\},\beta)$')
        #plt.title('Posterior distribution for N = %d events' % N)

        plt.plot(alpha_max,ymax,'o',label='max posterior')
        plt.plot(x_samples.mean(),ymax,'o',label='data average')
        plt.legend()

        return alpha_max[0], sample_mean, sample_mean_error, np.median(x_samples)

        plt.figure(figsize=(12,12))
        for i in range(Ntrial):
        print('Trial %d' % (i+1))
        plt.subplot(4,3,i+1)
        alpha_max, sample_mean, sample_mean_error, sample_median = runtrial(True)
        print('maximum posterior estimate = %.3f' % alpha_max)
        print('alpha using average = %.3f +- %.3f' % (sample_mean,sample_mean_error))
```

```
Trial 1
maximum posterior estimate = 0.879
alpha using average = 1.544 +- 0.461
Trial 2
maximum posterior estimate = 0.939
alpha using average = 2.490 +- 1.981
Trial 3
maximum posterior estimate = 0.879
alpha using average = 4.475 +- 7.114
Trial 4
maximum posterior estimate = 0.939
alpha using average = -0.949 +- 2.198
Trial 5
maximum posterior estimate = 1.121
alpha using average = 2.529 +- 0.816
Trial 6
maximum posterior estimate = 1.061
alpha using average = -0.909 +- 3.683
Trial 7
maximum posterior estimate = 1.061
alpha using average = 1.165 +- 0.855
Trial 8
maximum posterior estimate = 1.000
alpha using average = -17.228 +- 17.262
Trial 9
maximum posterior estimate = 1.000
alpha using average = 0.072 +- 1.519
Trial 10
maximum posterior estimate = 1.061
alpha using average = 37.385 +- 35.317
```




```
<Figure size 1200x1200 with 10 Axes>
```

In [6] :

```
        # 24)

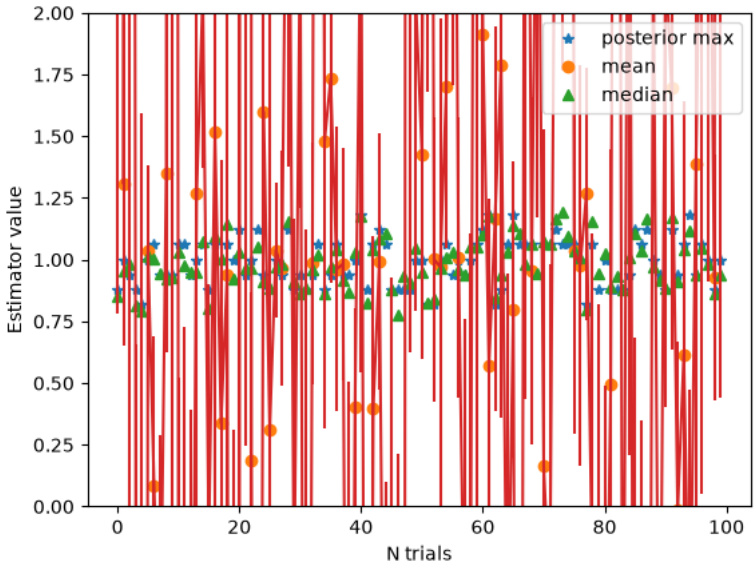
        Ntrial = 100

        posterior_max = []
        mean = []
        mean_error = []
        median = []

        for i in range(Ntrial):
        alpha_max, sample_mean, sample_mean_error, sample_median = runtrial(False)
        posterior_max += [alpha_max]
        mean += [sample_mean]
        mean_error += [sample_mean_error]
        median += [sample_median]

        plt.plot(range(Ntrial), posterior_max, '*',label='posterior max')
        plt.plot(range(Ntrial), mean, 'o',label='mean')
        plt.plot(range(Ntrial), median, '^',label='median')
        plt.errorbar(range(Ntrial), mean, yerr=mean_error)
        plt.ylim(0,2)
        plt.xlabel('N trials')
        plt.ylabel('Estimator value')
        plt.legend()
```

```
<matplotlib.legend.Legend at 0x28743b403d0>
```




```
<Figure size 640x480 with 1 Axes>
```

In [7] :

```
        print('Mean of posterior max values: %.3f +- %.3f'
        %(np.array(posterior_max).mean(),np.array(posterior_max).std()))
        print('Mean of median values: %.3f +- %.3f'
        %(np.array(median).mean(),np.array(median).std()))
        print('Mean of average values: %.3f +- %.3f'
        %(np.array(mean).mean(),np.array(mean).std()))
```

```
Mean of posterior max values: 0.989 +- 0.093
Mean of median values: 0.985 +- 0.100
Mean of average values: 2.208 +- 8.426
```

Conclusion: you should observe that the estimate using the average has enormous variance. This is a consequence of the fact that the Cauchy distribution is very peculiar as it has no defined mean or variance! Therefore the Central Limit Theorem is not applicable to the Cauchy distribution.

The morale of this exercise is that the sample mean is not alway a good number to use, and in this particular case it is best to use more refined knowledge of the posterior pdf to determine what is best. Such knowledge can be gained by outputting samples from such distributions by Monte Carlo methods.

> 25. What other estimate than the averaged sample value could be used to estimate α? Add this new estimator to your answer to the previous questions 23 and 24

In [8] :

```
```

CORRIGE — 1. Pile ou face (Q1–Q15)

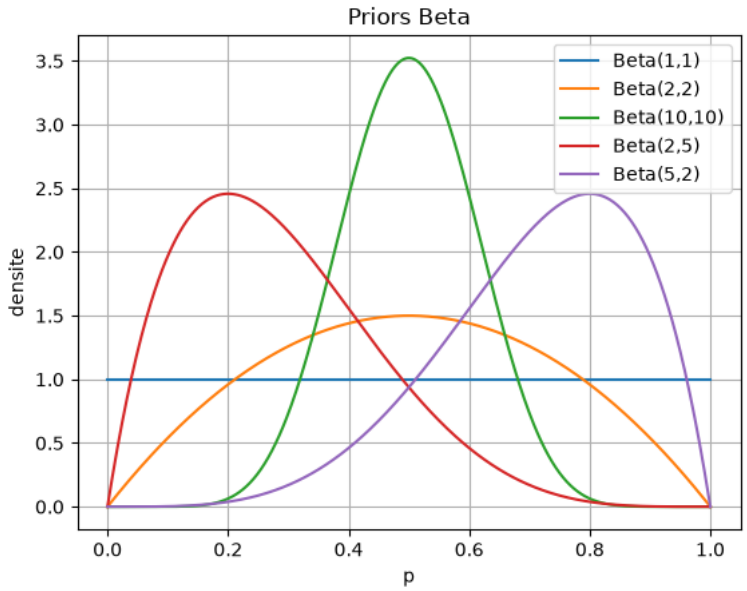
In [9] :

```
        # Q1 : estimation grossiere frequentiste
        k, n = 100, 140
        p_hat = k / n
        print(f"Q1 : p_hat = k/n = {p_hat:.4f}")
        # Q2 (math) : posterior Beta(k+a, n-k+b) car vraisemblance Binomiale x prior Beta
        # P(p|data) ∝ p^k(1-p)^{n-k} * p^{a-1}(1-p)^{b-1} = p^{k+a-1}(1-p)^{n-k+b-1}.
```

```
Q1 : p_hat = k/n = 0.7143
```

In [10] :

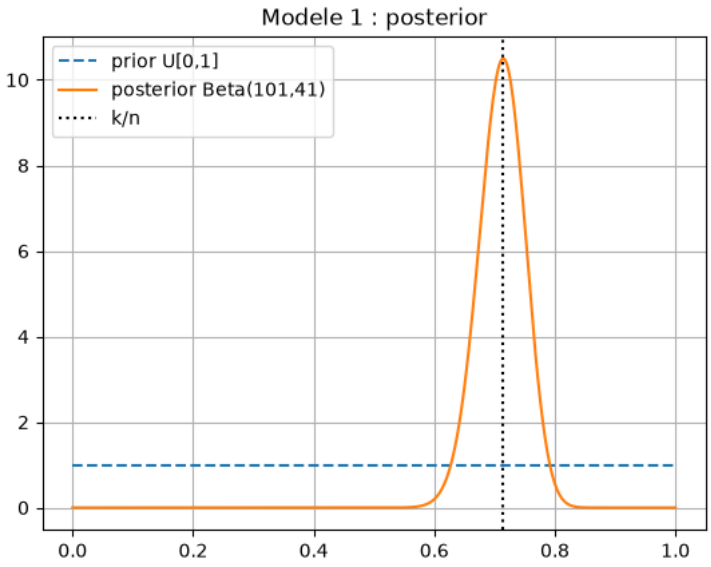
```
        # Q3 : formes du prior Beta pour plusieurs (a,b)
        import numpy as np
        import matplotlib.pyplot as plt
        from scipy.stats import beta
        xs = np.linspace(0, 1, 500)
        for (a, b) in [(1,1),(2,2),(10,10),(2,5),(5,2)]:
        plt.plot(xs, beta.pdf(xs, a, b), label=f"Beta({a},{b})")
        plt.title("Priors Beta"); plt.xlabel("p"); plt.ylabel("densite"); plt.legend();
        plt.grid(True); plt.show()
        # Q4 (math) : prior symetrique a=b centre en 1/2, non-informatif sur le biais piece,
        # invariant par echange pile/face, moyenne 1/2.
```




```
<Figure size 640x480 with 1 Axes>
```

In [11] :

```
        # Q5-Q6 : Modele 1, prior uniforme a1=b1=1 -> posterior Beta(101,41)
        a1=b1=1
        ap1, bp1 = k+a1, n-k+b1
        xs = np.linspace(0,1,500)
        plt.plot(xs, beta.pdf(xs, a1, b1), "--", label="prior U[0,1]")
        plt.plot(xs, beta.pdf(xs, ap1, bp1), label=f"posterior Beta({ap1},{bp1})")
        plt.axvline(k/n, color="k", linestyle=":", label="k/n")
        plt.legend(); plt.grid(True); plt.title("Modele 1 : posterior"); plt.show()
        MAP1 = (ap1-1)/(ap1+bp1-2); MMSE1 = ap1/(ap1+bp1)
        print(f"Q6 Modele1 : MAP={MAP1:.4f}, MMSE={MMSE1:.4f}, k/n={k/n:.4f} -> coherent.")
        print("Quand n grand, MAP et MMSE -> k/n (le prior se neglige).")
```

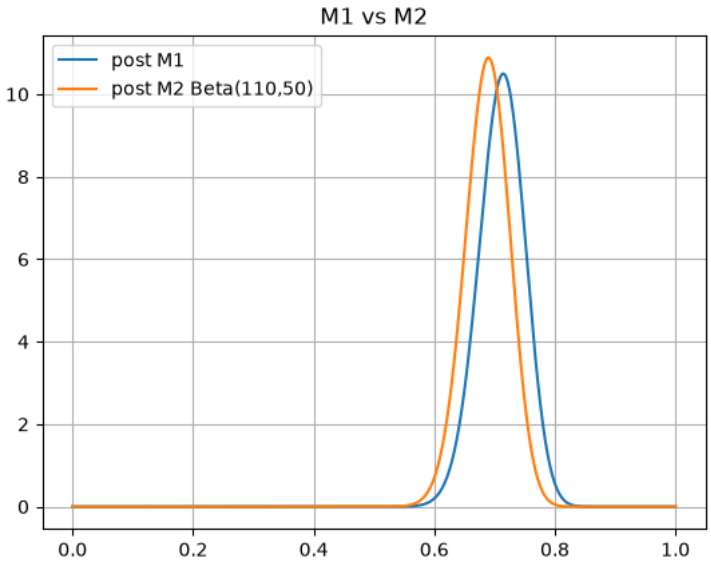



```
<Figure size 640x480 with 1 Axes>
```

```
Q6 Modele1 : MAP=0.7143, MMSE=0.7113, k/n=0.7143 -> coherent.
Quand n grand, MAP et MMSE -> k/n (le prior se neglige).
```

In [12] :

```
        # Q7-Q8 : Modele 2, prior pique a2=b2=10 -> posterior Beta(110,50)
        a2=b2=10
        ap2, bp2 = k+a2, n-k+b2
        xs = np.linspace(0,1,500)
        plt.plot(xs, beta.pdf(xs, ap1, bp1), label="post M1")
        plt.plot(xs, beta.pdf(xs, ap2, bp2), label=f"post M2 Beta({ap2},{bp2})")
        plt.legend(); plt.grid(True); plt.title("M1 vs M2"); plt.show()
        MAP2=(ap2-1)/(ap2+bp2-2); MMSE2=ap2/(ap2+bp2)
        print(f"Q8 Modele2 : MAP={MAP2:.4f}, MMSE={MMSE2:.4f} : legerement tire vers 1/2, variance
        plus faible que M1.")
```

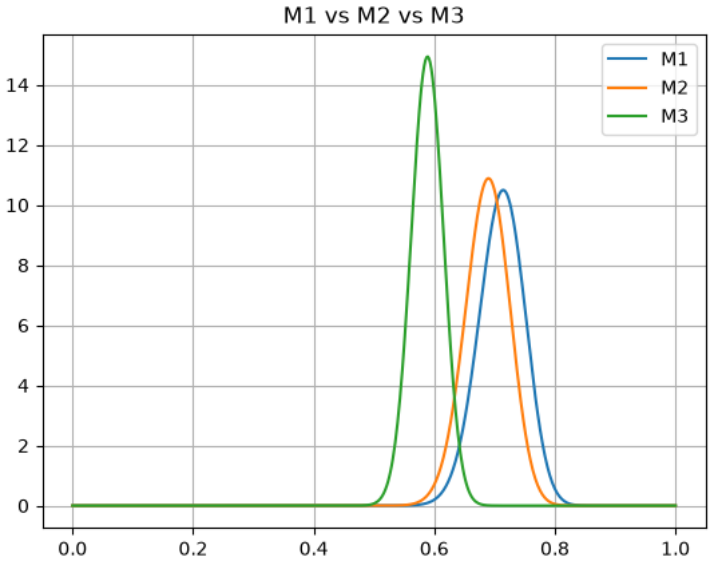



```
<Figure size 640x480 with 1 Axes>
```

```
Q8 Modele2 : MAP=0.6899, MMSE=0.6875 : legerement tire vers 1/2, variance plus faible que M1.
```

In [13] :

```
# Q9-Q10 : Modele 3, tres pique a3=b3=100 -> posterior Beta(200,140)
a3=b3=100
ap3, bp3 = k+a3, n-k+b3
xs = np.linspace(0,1,500)
for (ap,bp,lab) in [(ap1,bp1,"M1"),(ap2,bp2,"M2"),(ap3,bp3,"M3")]:
plt.plot(xs, beta.pdf(xs, ap, bp), label=lab)
plt.legend(); plt.grid(True); plt.title("M1 vs M2 vs M3"); plt.show()
MAP3=(ap3-1)/(ap3+bp3-2); MMSE3=ap3/(ap3+bp3)
print(f"Q10 Modele3 : MAP={MAP3:.4f}, MMSE={MMSE3:.4f} : fortement tire vers 1/2, le prior
domine les donnees.")
```




```
<Figure size 640x480 with 1 Axes>
```

```
Q10 Modele3 : MAP=0.5888, MMSE=0.5882 : fortement tire vers 1/2, le prior domine les donnees.
```

In [14] :

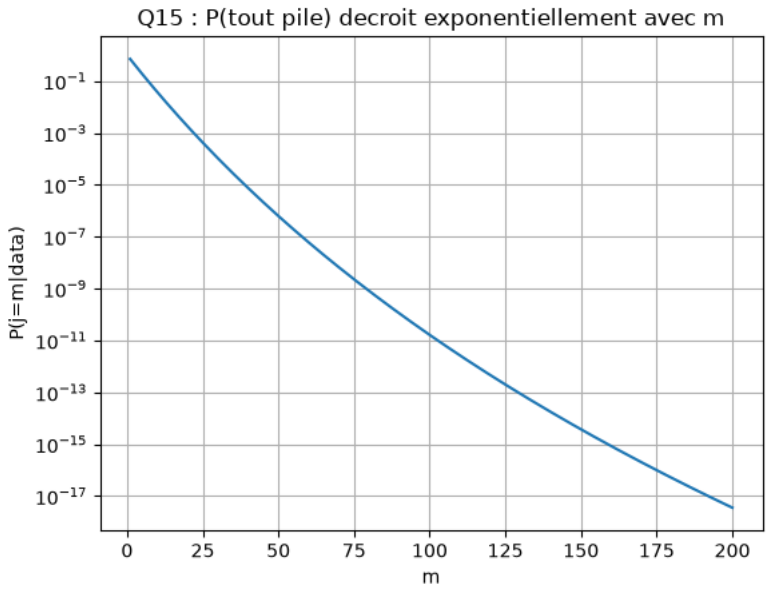
```
# Q11 : facteurs de Bayes via betaln
import scipy.special as sc
def log_marginal(k,n,a,b):
from scipy.special import gammaln
return float(gammaln(n+1)-gammaln(k+1)-gammaln(n-k+1) + sc.betaln(k+a, n-k+b) -
sc.betaln(a,b))
l1=log_marginal(k,n,a1,b1); l2=log_marginal(k,n,a2,b2); l3=log_marginal(k,n,a3,b3)
print(f"log P(data|M1)={l1:.2f}, M2={l2:.2f}, M3={l3:.2f}")
print(f"K12=exp(l1-l2)={np.exp(l1-l2):.2f}, K13={np.exp(l1-l3):.2f},
K23={np.exp(l2-l3):.2f}")
print("Conclusion : M1 (uniforme) largement prefere : les donnees (p~0.71) contredisent
les priors piques en 0.5.")
best=(a1,b1,ap1,bp1)
```

```
log P(data|M1)=-4.95, M2=-5.33, M3=-10.81
K12=exp(l1-l2)=1.46, K13=350.58, K23=240.62
Conclusion : M1 (uniforme) largement prefere : les donnees (p~0.71) contredisent les priors piques
en 0.5.
```

In [15] :

```
        # Q12-Q14 : predictive beta-binomiale
        from math import comb
        def pred(j,m,k,n,a,b):
        return comb(m,j)*np.exp(sc.betaln(k+j+a, n-k+m-j+b)-sc.betaln(a+k,b+n-k))
        # meilleur modele = M1
        a,b=k+a1,n-k+b1 # posterior params (101,41) ; a prior=a1
        print(f"Q13 P(2/2|data,M1)={pred(2,2,k,n,a1,b1):.4f}")
        print(f"Q14 P(3/5|data,M1)={pred(3,5,k,n,a1,b1):.4f}")
        # Q15 : comportement quand m grand
        ms=np.arange(1,201)
        # proba predictive du mode j=m (tout pile) et de j~0.7m ?
        p_all_heads=[pred(m,m,k,n,a1,b1) for m in ms]
        plt.semilogy(ms,p_all_heads); plt.grid(True)
        plt.xlabel("m"); plt.ylabel("P(j=m|data)"); plt.title("Q15 : P(tout pile) decroit
        exponentiellement avec m"); plt.show()
        print("Q15 : la predictive s'etale (variance croit avec m) ; la proba d'une sequence
        extreme -> 0.")
```

```
Q13 P(2/2|data,M1)=0.5073
Q14 P(3/5|data,M1)=0.2952
```



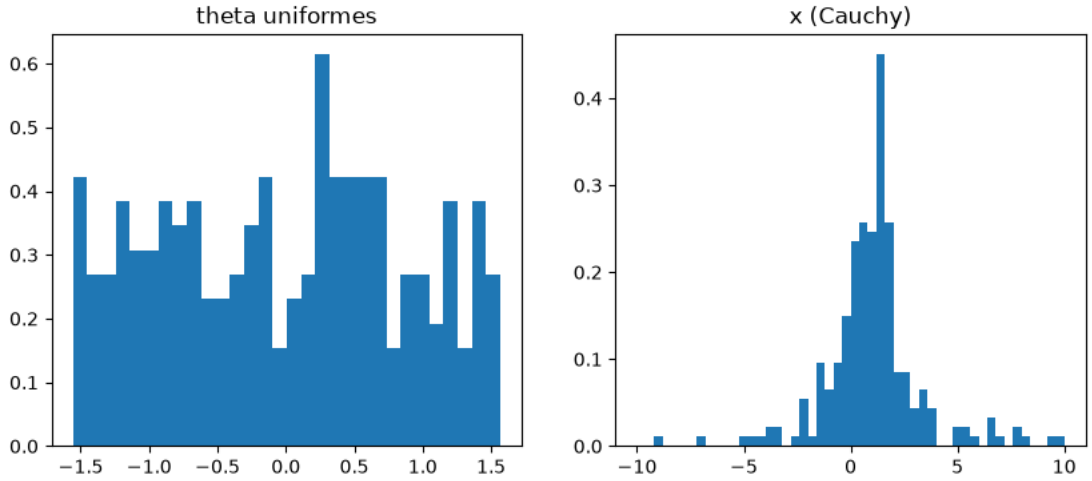

```
<Figure size 640x480 with 1 Axes>
```

```
Q15 : la predictive s'etale (variance croit avec m) ; la proba d'une sequence extreme -> 0.
```

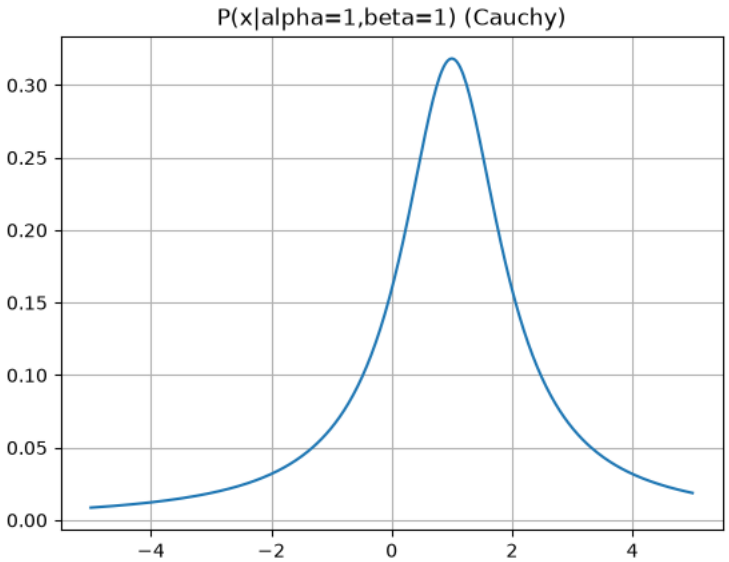
CORRIGE — 2. Phare (données manquantes Q16–Q18, Q25)

In [16] :

```
        # Q16-18 : generation des donnees (manquait dans le notebook) + P(x|alpha,beta)
        np.random.seed(0)
        N = 250
        beta_true = 1.0
        alpha_true = 1.0
        def generate_data(N, alpha=1.0, beta=1.0):
        theta = np.random.uniform(-np.pi/2, np.pi/2, size=N)
        x = alpha + beta*np.tan(theta)
        return theta, x
        theta_samples, x_samples = generate_data(N, alpha_true, beta_true)
        beta = beta_true
        # Q17 histogrammes
        fig,ax=plt.subplots(1,2,figsize=(10,4))
        ax[0].hist(theta_samples,bins=30,density=True); ax[0].set_title("theta uniformes")
        ax[1].hist(x_samples,bins=50,range=(-10,10),density=True); ax[1].set_title("x (Cauchy)")
        plt.show()
        # Q18 densite Cauchy
        def cauchy_pdf(x,alpha,beta):
        return beta/(np.pi*(beta**2+(x-alpha)**2))
        xg=np.linspace(-5,5,400)
        plt.plot(xg,cauchy_pdf(xg,1,1)); plt.title("P(x|alpha=1,beta=1) (Cauchy)");
        plt.grid(True); plt.show()
        print(f"Q16 : N={N} genere, moyenne empirique={x_samples.mean():.3f} (instable !),
        mediane={np.median(x_samples):.3f}")
```




```
<Figure size 1000x400 with 2 Axes>
```




```
<Figure size 640x480 with 1 Axes>
```

```
Q16 : N=250 genere, moyenne empirique=1.722 (instable !), mediane=1.062
```

In [17] :

```
        # Q25 : autre estimateur que la moyenne -> la MEDIANE (robuste pour Cauchy).
        # Pour une Cauchy, la mediane estime alpha de facon convergente, contrairement a la
        moyenne (pas d'esperance, pas de TCL).
        print("Q25 : utiliser la mediane empirique (ou le MAP). Deja compare en Q23-24 : mediane ~
        MAP ~ 1.0, moyenne tres volatile.")
```

```
Q25 : utiliser la mediane empirique (ou le MAP). Deja compare en Q23-24 : mediane ~ MAP ~ 1.0,
moyenne tres volatile.
```