Prof. Alex Rogers
Department of Computer Science
University of Oxford
21st January 2023

Bayesian Inference and Markov Chain Monte Carlo

Session 2

Contents

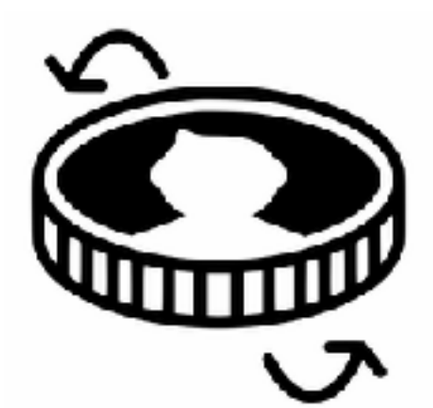
Bayesian Inference

Flipping a Fair Coin

Assume we flip a coin N times and it comes up heads n times. Is this a fair coin?

To answer this we should have some belief about the 'fairness' of the coin.

A fair coin will have a probability of coming up heads, p , such that $p = 1/2$.



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

Calculating the Likelihood

The likeliness of getting heads is determined by the binomial distribution - the probability of n successes in N Bernoulli trials.

$$P(n|p,N) = \binom{N}{n} p^n (1-p)^{N-n}$$

$$= \frac{N!}{n!\,(N-n)!} p^n (1-p)^{N-n}$$

Numerical Integration

To calculate our posterior belief we can perform numerical integration.

We assume a discrete distribution for our prior belief about p .

$$P(p|n,N) = \frac{P(n|p,N)P(p)}{\sum_p P(n|p,N)P(p)}$$

numerical_integration.ipynb

Analytic Solution

We can understand more about this particular setting by making the discrete prior distribution finer and finer until it is continuous.

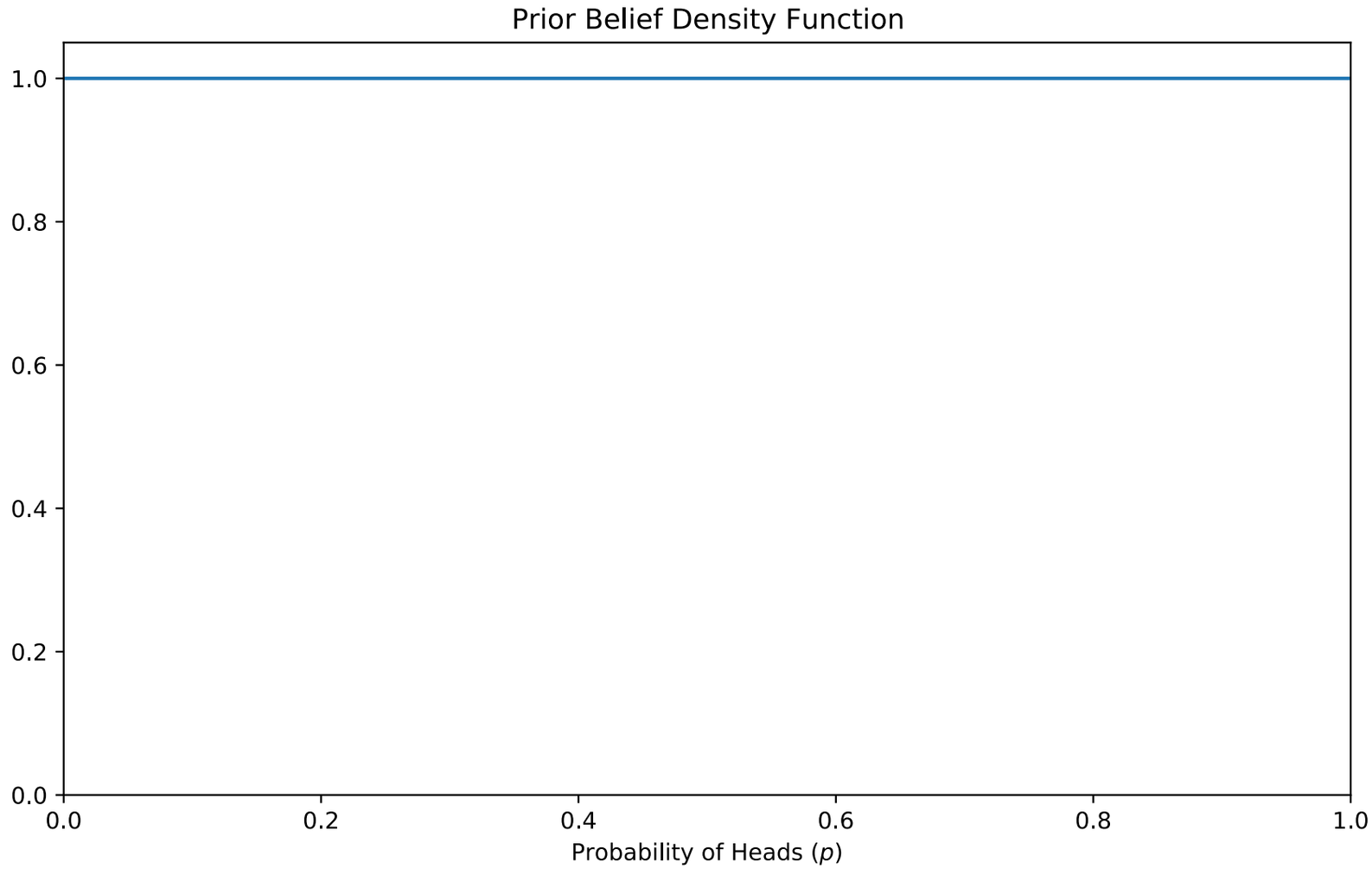
Bayes rules still holds is we use a probability density function, $\rho(p)$, rather than a discrete probability distribution, $P(p)$.

$$\rho(p|n,N) = \frac{P(n|p,N)\rho(p)}{\int_0^1 P(n|p,N)\rho(p)dp}$$

Analytic Solution

We choose a prior belief that represents an **uninformed prior.**

$$\rho(p) = 1$$



Analytic Solution

We can then expand out Bayes rule using the binomial likelihood derived earlier.

$$\rho(p|n,N) = \frac{\frac{N!}{n!(N-n)!}p^n(1-p)^{N-n}}{\int_0^1 \frac{N!}{n!\,(N-n)!}p^n(1-p)^{N-n}dp}$$

$$= \frac{p^n(1-p)^{N-n}}{\int_0^1 p^n(1-p)^{N-n}dp}$$

Analytic Solution

We can then make a change of variable such that:

$$a = n + 1$$

$$b = N - n + 1$$

We can then rewrite what we have so far in terms of these new variables:

$$\rho(p|a,b) = \frac{p^{a-1}(1-p)^{b-1}}{\int_0^1 p^{a-1}(1-p)^{b-1}dp}$$

Analytic Solution

The denominator now has a standard form and is known as a beta function:

$$B(a,b) = \int_0^1 p^{a-1}(1-p)^{b-1} dp$$

Thus, the posterior can then re-written as:

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

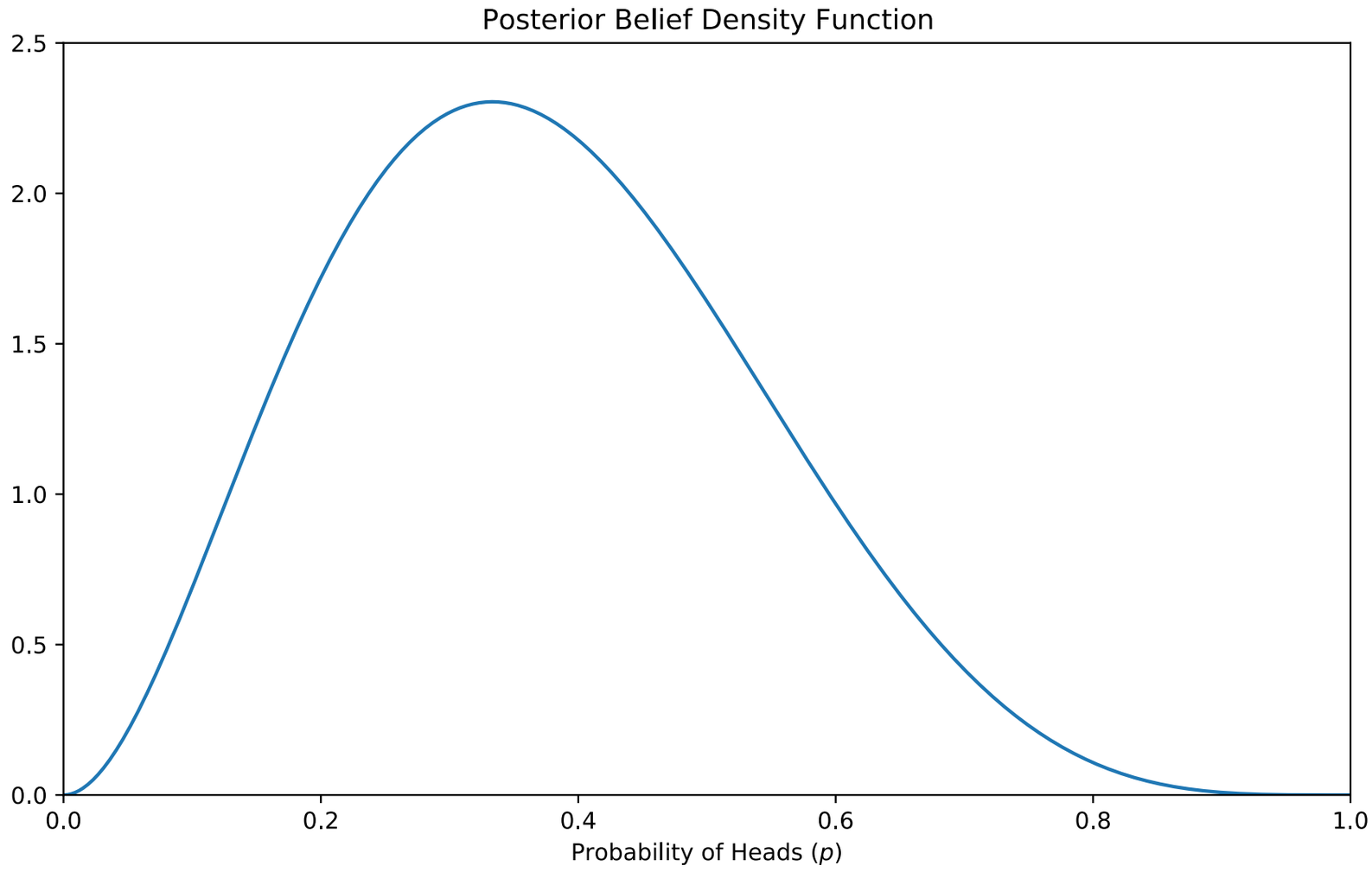
The posterior now also has a standard form and is known as a beta probability density function:

$$\rho(p|a,b) = \mathrm{Beta}(p|a,b)$$

Analytic Solution

We flip a coin 6 times and observe 2 heads such that $a = 3$ and $b = 5$.

$$\rho(p|n, N) = \text{Beta}(p|3, 5)$$



analytic_solution.ipynb

Analytic Solution

The same thing happens when the prior is a beta function; if the prior probability density function is a beta distribution and the likelihood function is a binomial distribution, then the posterior probability density function is always a beta distribution.

The beta function is a **conjugate prior** for binomial likelihoods.

The **uninformed prior** probability density function used earlier is also a beta function where $b = 1$ and $a = 1$.

A Gaussian distribution is a **conjugate prior** for Gaussian likelihoods.

Summary

We can perform numerical integration to calculate posterior beliefs.

- Easy to code.
- Scales poorly when the number of variables increases.
  - We have to explore the full range of all variables in order to calculate the denominator in Bayes rule.

We can solve Bayes rule analytically.

- This only works in particular settings with certain forms of prior and likelihood functions.
- It can be hard to do the integrations.
- Small changes to the setting mean we need to repeat all the work and derive the result again.

Sampling Methods

Sampling Methods

Often, instead of using an analytic solution, or an exhaustive numerical integration, we can generate useful information using sampling methods.

We do not need to know the exact form of the posterior distribution.

If we can generate samples from the posterior distribution we can use those to determine important properties (e.g. the mean and variance).

We use Markov chain Monte Carlo methods to do this efficiently.

Metropolis-Hastings Algorithms

The Metropolis-Hastings algorithm is a Markov chain Monte Carlo method. It efficiently generates samples from some distribution $f(x)$ which is proportional to a desired probability distribution $P(x)$.

We start at some arbitrary point x_i and use an arbitrary probability distribution $g(x'|x_i)$ to pick a candidate for the next sample.

We calculate an acceptance ratio given by:

$$\alpha = f(x')g(x'|x_i)/f(x_i)g(x_i|x')$$

We pick a random number, $u \in [0, 1]$.

If $u \leq \alpha$ add the candidate sample to the chain, $x_{i+1} = x'$.

Metropolis-Hastings Algorithms

We are trying to sample from the posterior given by:

$$\rho(p|n,N) = \frac{P(n|p,N)\rho(p)}{\int_0^1 P(n|p,N)\rho(p)dp}$$

The denominator is constant but we do not need to calculate it in order to use Metropolis-Hastings since we can use:

$$f(p) = P(n|p,N)\rho(p)$$

We can use a Gaussian to generate candidate samples $g(x'|x_i)$. This is symmetrical simplifying the process even further.

metropolis_hastings.ipynb

Probabilistic Programming

The Metropolis-Hastings algorithm is relatively simple. Yet we still need to tune parameters such as the exact distribution to use for $g(x'|x_i)$.

We also need to code the likelihood function and the prior probability distribution ourselves.

Probabilistic programming languages allow us to focus on defining the model and they provide state-of-the-art sampling methods to generate samples from the posterior distribution and contain extensive libraries of distributions to use a likelihoods and priors.

probablistic_programming.ipynb

Next Time

More Probability Distributions and PyMC3