BE/Bi 103: Data Analysis in the Biological Sciences

Justin Bois

Caltech

Fall, 2017

4 The theory of Markov chain Monte Carlo

4.1 Why MCMC?

When doing Bayesian analysis, our goal is very often to compute a posterior distribution, $g(\theta \mid D)$, where $\theta = \{\theta^{(1)}, \theta^{(2)}, \ldots\}$ is a set of possibly many parameters. However, just having an analytical expression for the posterior is of little use if we cannot understand any properties about it. Importantly, we often want to marginalize the posterior; that is, we want to integrate over parameters we are not interested in and get simpler distributions for those we are. This is often necessary to understand all but the simplest models. Doing these marginalizations requires what David MacKay calls "macho integration," which is often impossible to do analytically.

Furthermore, we may also want to compute **expectations** out of the posterior. For example, we might want the mean, or expectation value, of parameter $\theta^{(1)}$. If we know the posterior, this is

$$E[\theta^{(1)}] = \int \mathrm{d}\theta \; \theta^{(1)} \, g(\theta \mid D). \tag{4.1}$$

Generally, we can compute the expectation of any function of the parameters, $h(\theta)$, and we often want to. This is

$$E[h(\theta)] = \int \mathrm{d}\theta \, h(\theta) \, g(\theta \mid D). \tag{4.2}$$

So, pretty much anything we want to know about the posterior requires computation of an integral.

MCMC allows us to **sample** out of an arbitrary probability distribution, which includes pretty much any posterior we could write down.[9] By sampling, we mean that we can choose values of the parameters, θ, where the probability that we choose a given value is proportional to the posterior probability. Note that each sample consists of a complete set of parameters θ; that is a sample contains a value for $\theta^{(1)}$, a value for $\theta^{(2)}$, ... We are more likely to choose samples of high probability than of low. Using MCMC, we collect a large number of these samples.

From these samples, we can trivially perform marginalizations. Say we are interested in the marginalized distribution

$$g(\theta^{(1)} \mid D) = \left(\int \mathrm{d}\theta^{(2)} \int \mathrm{d}\theta^{(3)} \cdots \right) g(\theta \mid D). \tag{4.3}$$

[9] Well, not *any*. For some cases, we may not be able to make a transition kernel that satisfies the necessary properties, which I describe in the following pages.

Given a set of MCMC samples out of $g(\theta \mid D)$, to get a set of samples out of $g(\theta^{(1)} \mid D)$, we simply ignore the values of $\theta^{(2)}$, $\theta^{(3)}$, ...! Then, given the samples of the marginalized posterior, we can plot the CDF of the marginalized posterior as an ECDF of the samples, and the PDF of the marginalized posterior as a histogram of the samples.

To compute expectations, the MCMC samples are again very convenient. Now, we just approximate the integral with an average over samples.

$$E(h(\theta)) = \int \mathrm{d}\theta \, h(\theta) \, g(\theta \mid D) \approx \frac{1}{N} \sum_{i=1}^{N} h(\theta_i), \tag{4.4}$$

where θ_i is the ith of N MCMC samples taken from the posterior.

It is now abundantly clear why the ability to generate samples from the posterior is so powerful. But generating samples that actually come from the probability distribution of interest is not a trivial matter. We will discuss how this is accomplished through MCMC.

4.2 The basic idea behind MCMC

We often draw *independent* samples from a **target distribution**. For example, we could use `np.random.uniform(0, 1, 100)` to draw 100 independent samples from a uniform distribution on the domain [0, 1]. Generating independent samples for complicated target distributions is difficult.

But the samples need not be independent! Instead, we only need that the samples be generated from a process that generates samples from the target distribution in the correct proportions. In the case of the parameter estimation problem, this distribution is the posterior distribution parametrized by θ, $g(\theta \mid D)$. For notational simplicity in what follows, since we know we are always talking about a posterior distribution, we will use $P(\theta)$ for shorthand notation for an arbitrary distribution of theta.

The approach of MCMC is to take random walks in parameter space such that the probability that a walker arrives at point θ is proportional to $P(\theta)$. This is the main concept and is important enough to repeat.

> The approach of MCMC is to take random walks in parameter space such that the probability that a walker arrives at point θ is proportional to $P(\theta)$.

If we can achieve such a walk, we can just take the walker positions as samples from the distributions. To implement this random walk, we define a **transition kernel**, $T(\theta_{i+1} \mid \theta_i)$, the probability of a walker stepping from position θ_i in parameter space to position θ_{i+1}. The transition kernel defines a **Markov chain**, which you

can think of as a random walker whose next step depends only on where the walker is right now; i.e., it has no memory.

The condition that the probability of arrival at point θ_{i+1} is proportional to $P(\theta_{i+1})$ may be stated as

$$P(\theta_{i+1}) = \int \mathrm{d}\theta_i \, T(\theta_{i+1} \mid \theta_i) \, P(\theta_i). \tag{4.5}$$

Here, we have taken θ to be continuous. Were it discete, we just replace the integral with a sum. When this relation holds, it is said that the target distribution is an **invariant distribution** or **stationary distribution** of the transition kernel. When this invariant distribution is unique, it is called a **limiting distribution**. We want to choose our transition kernel $T(\theta_{i+1} \mid \theta_i)$ such that $P(\theta)$ is limiting. This is the case if equation (4.5) holds and the chain is **ergodic**. An ergodic Markov chain has the following properties:

1. It is **aperiodic**. A periodic Markov chain can only return to a given point in parameter space after k, $2k$, $3k, \ldots$ steps, where k is the period. An aperiodic chain is not periodic.
2. It is **irreducible**, which means that any point in parameter space is accessible to the walker from any other.
3. It is **positive recurrent**, which means that the walker will surely come revisit any point in parameter space in a finite number of steps.

So, if our transition kernel satisfies this checklist and equation (4.5), it will eventually sample the posterior distribution. We will discuss how to come up with such a transition kernel in a moment; for now we focus on the important concept of "eventually" in the preceding sentence.

4.3 Tuning

Imagine for a moment that we devised a transition kernel that satisfies the above properties. Say we start a walker at position θ_0 in parameter space and it starts walking according to the transition kernel. Most likely, for those first few steps, the walker is traversing a part of parameter space that has incredibly low probability. Once it got to regions of high probability, the walker would almost never return to the region of parameter space in which it began. So, unless we sample for an incredibly long time, those first few samples visited are over-weighted. So, we need to let the walker walk for a while without keeping track of the samples so that it can arrive at the limiting distribution. This is called **tuning**, otherwise known as **burn-in** or **warm up**[10].

[10] When using NUTS with PyMC3, the tuning is a bit more than just burn-in, where we simply neglect samples. The algorithm is actively choosing stepping strategies during the tuning phase.

There is no general way to tell if a walker has reached the limiting distribution, so we do not know how many burn-in steps are necessary. There are several heuristics. For example, Gelman and coauthors proposed generating several tuning chains and computing the **Gelman-Rubin** $\hat{R}$ statistic,

$$\hat{R} = \frac{\text{variance between the chains}}{\text{mean variance within the chains}}. \tag{4.6}$$

Limiting chains have $\hat{R} \approx 1$, so you can use this as a metric for having achieved stationarity. Gelman and his coauthors in their famous book *Bayesian Data Analysis* suggest that $|1 - \hat{R}| < 0.1$ as a good rule of thumb for stationary chains.

4.4 Generating a transition kernel: The Metropolis-Hastings algorithm

The **Metropolis-Hastings algorithm** covers a widely used class of algorithms for MCMC sampling. I will first state the algorithm here, and then we will show that it satisfies the necessary conditions for the walkers to be sampling out of the target posterior distribution.

4.4.1 The algorithm/kernel

Say our walker is at position θ_i in parameter space.

1. We randomly choose a candidate position θ' to step to next from an arbitrary **proposal distribution** $K(\theta' \mid \theta_i)$.
2. We compute the **Metropolis ratio**,

$$r = \frac{P(\theta')\,K(\theta_i \mid \theta')}{P(\theta_i)\,K(\theta' \mid \theta_i)}. \tag{4.7}$$

3. If $r \geq 1$, accept the step and set $\theta_{i+1} = \theta'$. Otherwise, accept the step with probability r. If we do reject the step, set $\theta_{i+1} = \theta_i$.

The last two steps are used to define the transition kernel $T(\theta_{i+1} \mid \theta_i)$. We can define the acceptance probability of the proposal step as

$$\alpha(\theta_{i+1} \mid \theta_i) = \min(1, r) = \min\left(1, \frac{P(\theta_{i+1})\,K(\theta_i \mid \theta_{i+1})}{P(\theta_i)\,K(\theta_{i+1} \mid \theta_i)}\right). \tag{4.8}$$

Then, the transition kernel is

$$T(\theta_{i+1} \mid \theta_i) = \alpha(\theta_{i+1} \mid \theta_i)\,K(\theta_{i+1} \mid \theta_i). \tag{4.9}$$

4.4.2 Detailed balance

This algorithm seems kind of nuts! How on earth does this work? To investigate this, we consider the joint probability, $P(\theta_{i+1}, \theta_i)$, that the walker is at θ_i and θ_{i+1} at sequential steps. We can write this in terms of the transition kernel,

$$\begin{aligned}
P(\theta_{i+1}, \theta_i) &= P(\theta_i)\, T(\theta_{i+1} \mid \theta_i) \\
&= P(\theta_i)\, \alpha(\theta_{i+1} \mid \theta)\, K(\theta_{i+1} \mid \theta_i) \\
&= P(\theta_i)\, K(\theta_{i+1} \mid \theta)\, \min\left(1, \frac{P(\theta_{i+1})\, K(\theta_i \mid \theta_{i+1})}{P(\theta_i)\, K(\theta_{i+1} \mid \theta_i)}\right) \\
&= \min\left[P(\theta_i)\, K(\theta_{i+1} \mid \theta_i), P(\theta_{i+1})\, K(\theta_i \mid \theta_{i+1})\right] \\
&= P(\theta_{i+1})\, K(\theta_i \mid \theta_{i+1})\, \min\left(1, \frac{P(\theta_i)\, K(\theta_{i+1} \mid \theta_i)}{P(\theta_{i+1})\, K(\theta_i \mid \theta_{i+1})}\right) \\
&= P(\theta_{i+1})\, \alpha(\theta_i \mid \theta_{i+1})\, K(\theta_i \mid \theta_{i+1}) \\
&= P(\theta_{i+1})\, T(\theta_i \mid \theta_{i+1}). \qquad (4.10)
\end{aligned}$$

Thus, we have

$$P(\theta_i)\, T(\theta_{i+1} \mid \theta_i) = P(\theta_{i+1})\, T(\theta_i \mid \theta_{i+1}). \qquad (4.11)$$

This says that the rate of transition from θ_i to θ_{i+1} is equal to the rate of transition from θ_{i+1} to θ_i. In this case, the transition kernel is said to satisfy **detailed balance**.

Any transition kernel that satisfies detailed balance has $P(\theta)$ as an invariant distribution. This is easily shown.

$$\begin{aligned}
\int \mathrm{d}\theta_i\, P(\theta_i)\, T(\theta_{i+1} \mid \theta_i) &= \int \mathrm{d}\theta_i\, P(\theta_{i+1})\, T(\theta_i \mid \theta_{i+1}) \\
&= P(\theta_{i+1}) \left[\int \mathrm{d}\theta_i\, T(\theta_i \mid \theta_{i+1})\right] \\
&= P(\theta_{i+1}), \qquad (4.12)
\end{aligned}$$

since the bracketed term is unity because the transition kernel is a probability.

Note that all transition kernels that satisfy detailed balance have an invariant distribution. (If the chain is ergodic, this is a limiting distribution.) But not all kernels that have an invariant distribution satisfy detailed balance. So, detailed balance is a sufficient condition for a transition kernel having an invariant distribution.

4.4.3 Choosing the transition kernel

There is an art to choosing the transition kernel. The original Metropolis algorithm (1953), took $K(\theta_{i+1} \mid \theta_i) = 1$. As a rule of thumb, you want to choose a proposal distribution such that you get an acceptance rate of about 0.4. If you accept every step, the walker just wanders around and it takes a while to get to the limiting distribution. If you reject too many steps, the walkers never move, and it again takes a long time to get to the limiting distribution. There are tricks to "tune" the walkers to achieve the target acceptance rate.

Gibbs sampling, which is popular, though we will not go into the details, is a special case of a Metropolis-Hastings sampler, as is the No U-turn sampler (NUTS), which is an example of a **Hamiltonian Monte Carlo** sampler. These both result in significant performance improvements for important subclasses of problems. The sampler employed by `emcee`, the affine invariant ensemble sampler (Goodman and Weare, *J. Comp. Sci.*, **5**, 65–80, 2000), utilizes many walkers walking at the same time, sharing information between them. It is technically not a Metropolis-Hastings sampler, but many of the ideas presented in this lecture there apply for ensuring that the sampler is indeed sampling the appropriate posterior distribution.

Finally, importantly, the No U-Turn sampler and the affine invariant sample can only handle continuous variables; they cannot sample discrete variables. Depending on your problem, this could be a serious limitation.

In this class, we will use PyMC3, which uses NUTS. We will not delve into the algorithmic details, but it helps to have a feel for how the algorithm works. To educate yourself more‘ recommend Michael Betencourt's conceptual introduction to Hamiltonian Monte Carlo and this lecture by him on that topic.