A Markov Chain Monte Carlo Algorithm

Kyla Pohl

January 2025

This topic is something that I've heard people say offhandedly for a long time, but I have never understood. That's come back to bite me now that I need to understand this for my research. That said, I have decided that I find this sort of stuff very interesting.

This talk will have a more probabilistic flavor than what we usually expect from this seminar. Moreover, we will speak of plane partitions without mention of the dimer model contrary to our usual ways. Most of the information for this talk was taken from Wikipedia. Chatgpt also helped my learn some stuff as well as write all the code. I spoke with David Levin last night to learn a few things as well.

Outline

- I will first define MCMC.
- Next, I will give an example via the M–H algorithm.
- This will be followed by some details about the M–H algorithm.
- Finally, I will use the M–H algorithm to generate a uniformly random plane partition which resides in a 10 by 10 by 10 cube.

Definition

Definition. A *Markov chain Monte Carlo algorithm* is a process used to draw samples from a probability distribution by performing a random walk within the space. It is especially useful when sampling directly from the distribution is difficult.

The starting point is arbitrary, but there must be sufficient "burn-in time" to reach a representative result. From wiki: "The more steps that are included, the more closely the distribution of the sample matches the actual desired distribution."

First Example

Let's sample from the standard normal distribution $N(0,1)$ via an MCMC algorithm. Note that the equation for the standard normal curve is

$$\text{targetpdf}(x) = \frac{e^{\frac{-x^2}{2}}}{\sqrt{2\pi}}.$$

Algorithm:

1. Select a starting position. For this example, let's start at the mean, 0.
2. Propose a step by picking from the $N(0, 0.1)$ distribution. If we pick, say, -0.08, we are proposing to step 0.08 to the left from our current position.

3. Decide whether to accept or reject our proposal. We define our acceptance probability to be

$$\alpha = \min\left(1, \frac{\text{targetpdf}(\textit{proposed})}{\text{targetpdf}(\textit{current})}\right).$$

4. If we accept, move to the new position. Regardless, repeat this process from step 2 until we've reached the number of steps that we want. In this case, let's do 10,000 steps.

Note that we're kind of defeating the purpose here since we're sampling from a Gaussian distribution to use an MCMC algorithm to generate a Gaussian, but nevertheless, it's a good example.

Let's run this in Python. (Switch to projector.)

This particular MCMC algorithm is called a *Metropolis–Hastings* algorithm. A M–H algorithm is characterized by relying on an acceptance probability for each step.

History of M–H

This algorithm was first proposed in 1953 by Nicholas Metropolis, Arianna W. Rosenbluth, Marshall Rosenbluth, Augusta H. Teller and Edward Teller. In 1970, W.K. Hastings generalized this idea, which is how the algorithm came to be known as M–H. The credit for the original 1953 work is disputed, with M. Rosenbluth and E. Teller each giving different accounts of what the others contributed.

Algorithm for using this Algorithm

1. Pick an arbitrary starting state x. With sufficiently many steps, it won't matter what you picked first.
2. Propose a stop from your current state to a new state y. This occurs with some probability $q(x \to y)$ from a distribution on all the legal steps from your current state.
3. Decide whether to accept or reject the proposal. Accept with probability

$$\alpha = \min\left(1, \frac{\pi(y)q(y \to x)}{\pi(x)q(x \to y)}\right)$$

   where $\pi(x)$ is the probability that you select x from the steady state of this algorithm. In other words, $\pi(x)$ is the probability that you "shake the bag" 50,000 times and pull out x.
4. The next step depends on our accept/reject choice.
   (a) If we accept the proposal, move to the new state y and repeat the algorithm from step 2, now using y as our "current state."
   (b) If we reject the proposal, this was all for naught. Try again from step 2.

Detailed Balance

This algorithm satisfies the *detailed balance equation*:

$$\pi(x)P(x \to y) = \pi(y)P(y \to x).$$

We need to describe what the pieces of this expression mean.

- $\pi(x)$ is the chance of selecting x from the stationary distribution generated by this algorithm. (Why should one exist?)
- $P(x \to y)$ is the probability of stepping from x to y, i.e.

$$P(x \to y) = \alpha(x \to y)q(x \to y).$$

Why is this equation satisfied? (Proof stolen from ChatGPT.)
Let's consider the following two cases.

1. **When** $\frac{\pi(y)q(y\to x)}{\pi(x)q(x\to y)} \leq 1$:
   - In this case, the acceptance probability is
   $$\alpha(x \to y) = \frac{\pi(y)q(y \to x)}{\pi(x)q(x \to y)}.$$
   - The left-hand side of the detailed balance condition is:
   $$\pi(x)q(x \to y)\alpha(x \to y) = \pi(x)q(x \to y)\frac{\pi(y)q(y \to x)}{\pi(x)q(x \to y)} = \pi(y)q(y \to x).$$
   - Since $\alpha(y \to x) = 1$, the right-hand side is:
   $$\pi(y)q(y \to x)\alpha(y \to x) = \pi(y)q(y \to x).$$
   - Thus, detailed balance holds in this case.
2. **When** $\frac{\pi(y)q(y\to x)}{\pi(x)q(x\to y)} > 1$:
   - Now, the acceptance probability is $\alpha(x \to y) = 1$, so the left-hand side of the condition is:
   $$\pi(x)q(x \to y)\alpha(x \to y) = \pi(x)q(x \to y).$$
   - The acceptance probability in the reverse direction is:
   $$\alpha(y \to x) = \frac{\pi(x)q(x \to y)}{\pi(y)q(y \to x)}.$$
   - The right-hand side of the detailed balance condition is:
   $$\pi(y)q(y \to x)\alpha(y \to x) = \pi(y)q(y \to x)\frac{\pi(x)q(x \to y)}{\pi(y)q(y \to x)} = \pi(x)q(x \to y).$$
   - Thus, detailed balance holds in this case as well.

Second Example

Let's create a uniformly random plane partition which remains inside the 10 by 10 by 10 cube using the M–H algorithm.

Algorithm:

1. Pick an arbitrary plane partition which sits inside the 10 by 10 by 10 cube. Say, $[[4, 3, 3, 1], [2, 1, 1], [1, 1]]$. A "step" in this algorithm will be adding or removing a box in the plane partition.
2. Make a list of all of the acceptable ways we can add or subtract a box from this pp such that
   - this remains a pp, and
   - we stay inside the cube.
3. Select an option uniformly randomly from this list.

4. Accept it with probability $\alpha = \min\left(1, \frac{q(\lambda' \to \lambda)}{q(\lambda \to \lambda')}\right)$, where

$$q(\lambda \to \lambda')$$

is the probability that λ maps to λ' when we select something uniformly randomly from the list of things we can step to from λ. In other words,

$$q(\lambda \to \lambda') = \begin{cases} \frac{1}{\#\text{ of places you can go from } \lambda} & \lambda' \text{ is a legal step from } \lambda \\ 0 & \text{else.} \end{cases}$$

5. Repeat from step 2.

Claim. *After sufficient "burn in." this will yield a uniformly random plane partition which fits inside a 10 by 10 by 10 cube.*

Perhaps this is surprising. It is to me! So why does this work? We've defined our acceptance probability very carefully!

Proof. We'd like to show that $\pi(\lambda) = \pi(\lambda')$ for two arbitrary plane partitions living in the 10 by 10 by 10 cube. By the detailed balance equation,

$$\pi(\lambda)P(\lambda \to \lambda') = \pi(\lambda')P(\lambda' \to \lambda).$$

So it suffices to show that

$$P(\lambda \to \lambda') = P(\lambda' \to \lambda).$$

Recall that

$$\begin{aligned} P(\lambda \to \lambda') = q(\lambda \to \lambda')\alpha(\lambda \to \lambda') &= q(\lambda \to \lambda') \cdot \min\left(1, \frac{q(\lambda' \to \lambda)}{q(\lambda \to \lambda')}\right) \\ &= \min\left(q(\lambda \to \lambda'), q(\lambda' \to \lambda)\right) = \cdots = P(\lambda' \to \lambda). \end{aligned}$$

□

(Show preliminary algorithm on projector.)