Chapter 4

Markov Chain Monte Carlo. Metropolis-Hastings Algorithm.

Ali Raisolsadat

School of Mathematical and Computational Sciences
University of Prince Edward Island

Metropolis–Hastings

The Metropolis algorithm is a special case of the more general Metropolis–Hastings algorithm.

- Metropolis-Hastings uses a general proposal distribution

$$q(\hat{x}^{(t+1)} \mid x^{(t)}) = q(x^{(t)} \to \hat{x}^{(t+1)})$$

- In the Metropolis algorithm, q is typically a symmetric Gaussian centered at $x^{(t)}$.

MH accepts the proposed state $\hat{x}^{(t)}$ with probability

$$\frac{\tilde{p}(\hat{x}^{(t)})}{\tilde{p}(x^{(t-1)})} \cdot \frac{q(x^{(t-1)} \to \hat{x}^{(t)})}{q(\hat{x}^{(t)} \to x^{(t-1)})}$$

The ratio

$$\frac{q(x^{(t-1)} \to \hat{x}^{(t)})}{q(\hat{x}^{(t)} \to x^{(t-1)})}$$

ensures detailed balance when the proposal is asymmetric: if a move is easier to propose one way than the reverse, it becomes harder to accept.

Under mild conditions (e.g., $q(x \to x') > 0$ for all states), the Markov chain converges to the target distribution. However, the rate of convergence can vary significantly depending on the choice of q.

Metropolis–Hastings Example: Rolling a Die with Coins

Suppose we want to sample from a fair 6-sided die:

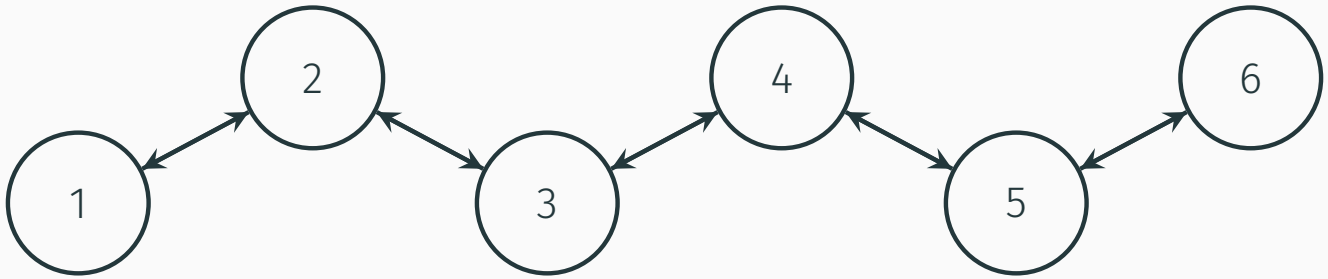
$$P(X = c) = \frac{1}{6} \qquad c \in \{1, \ldots, 6\}$$

but we have no die or computer. We only have coins. We also want to avoid rejection sampling.

Consider the following random-walk proposal on $\{1, \ldots, 6\}$:

- If $x = 1$, always propose 2
- If $x = 2$, propose 1 or 3 with probability 1/2 each
- If $x = 3$, propose 2 or 4 with probability 1/2 each
- If $x = 4$, propose 3 or 5 with probability 1/2 each
- If $x = 5$, propose 4 or 6 with probability 1/2 each
- If $x = 6$, always propose 5

What are we doing? flip a coin; move up on heads (if possible), move down on tails (if possible). This induces a random walk on the state graph.



Metropolis–Hastings Example: Rolling a Die with Coins

"Roll a die with a coin" by using the random-walk proposal q in M-H:

$$q(1 \to 2) = 1, \quad q(2 \to 1) = \tfrac{1}{2}, \quad q(2 \to 3) = \tfrac{1}{2}, \ \ldots, \ q(6 \to 5) = 1$$

If $x = 3$ and we propose $\hat{x} = 2$, we always accept. The acceptance check is

$$u < \frac{p(2)}{p(3)} \cdot \frac{q(2 \to 3)}{q(3 \to 2)} = \frac{1/6}{1/6} \cdot \frac{1/2}{1/2} = 1$$

The same holds for any x in the "middle" states $\{2, 3, 4, 5\}$.

If $x = 2$ and we propose $\hat{x} = 1$, we also always accept:

$$u < \frac{p(1)}{p(2)} \cdot \frac{q(1 \to 2)}{q(2 \to 1)} = \frac{1/6}{1/6} \cdot \frac{1}{1/2} = 2$$

If x is at an endpoint (1 or 6), then the acceptance probability becomes $\frac{1}{2}$:

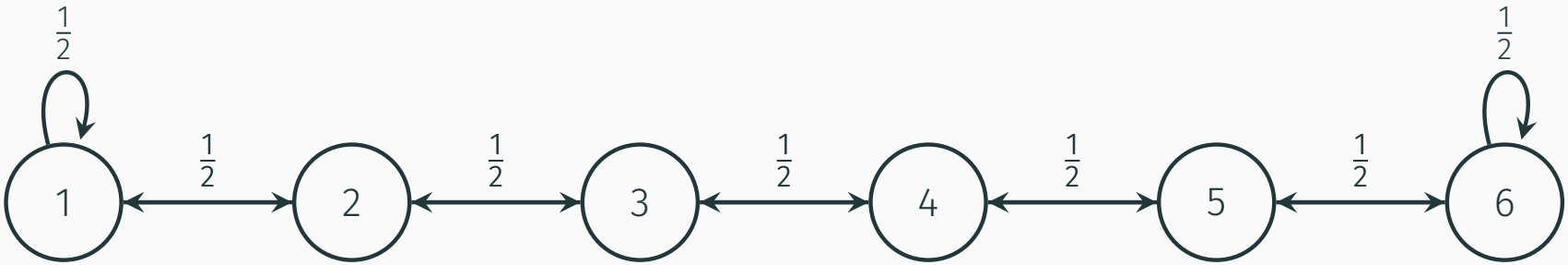
$$u < \frac{p(2)}{p(1)} \cdot \frac{q(2 \to 1)}{q(1 \to 2)} = \frac{1/6}{1/6} \cdot \frac{1/2}{1} = \frac{1}{2}$$

Metropolis–Hastings Example: Rolling a Die with Coins

Metropolis–Hastings adjusts the raw random-walk proposal to ensure the correct stationary distribution:

- At the endpoints (1 or 6), MH forces you to stay put half of the time.
- This accounts for the fact that 1 and 6 each have only one neighbour; which means they are not visited as often by the random walk.

This could be viewed as random surfer in a different graph:



In general, MH provides the modification that makes the random walk have the correct stationary probabilities, regardless of the (reasonable) proposal distribution q you start with.

Metropolis–Hastings Algorithm

Metropolis–Hastings algorithm: Sampling from a target distribution $p(x)$ using a general proposal $q(\hat{x} \mid x)$. We assume $p(x)$ is known up to a normalizing constant:

$$p(x) = \frac{\tilde{p}(x)}{Z} \qquad Z \text{ unknown}$$

Algorithm:

1. Initialize $x^{(0)}$
2. Until we get bored:
   - 2.1 Generate a proposal from a general proposal distribution
   $$\hat{x}^{(t)} \sim q(\hat{x} \mid x^{(t-1)})$$
   - 2.2 Generate $u \sim \text{Uniform}[0, 1]$
   - 2.3 Accept the proposal if
   $$u \leq \frac{\tilde{p}(\hat{x}^{(t)})}{\tilde{p}(x^{(t-1)})} \cdot \frac{q(x^{(t-1)} \mid \hat{x}^{(t)})}{q(\hat{x}^{(t)} \mid x^{(t-1)})}$$
   and set $x^{(t)} = \hat{x}^{(t)}$.
   - 2.4 Otherwise, reject the proposal and set $x^{(t)} = x^{(t-1)}$

- If the proposal is symmetric (e.g., Gaussian noise), the ratio of q's cancels, recovering the Metropolis algorithm.
- Proposals that increase $\tilde{p}(x)$ are always accepted.
- Proposals that decrease $\tilde{p}(x)$ are accepted with some probability.
- Works even when the normalizing constant Z is unknown.

Algorithm 1 Metropolis–Hastings Algorithm for Sampling from $p(x)$

Require: Initial state $x^{(0)}$, number of iterations T, proposal kernel $q(\hat{x} \mid x)$
Ensure: Samples $\{x^{(t)}\}_{t=1}^{T}$ approximately distributed according to $p(x)$

1: Set $x^{(0)}$ as the starting state
2: **for** $t = 1$ **to** T **do**
3: Draw proposal $\hat{x}^{(t)} \sim q(\cdot \mid x^{(t-1)})$
4: Generate $u \sim \text{Uniform}(0, 1)$
5: Compute acceptance probability

$$\alpha = \min\left(1, \frac{\tilde{p}(\hat{x}^{(t)})}{\tilde{p}(x^{(t-1)})} \frac{q(x^{(t-1)} \mid \hat{x}^{(t)})}{q(\hat{x}^{(t)} \mid x^{(t-1)})}\right)$$

6: **if** $u \leq \alpha$ **then**
7: Accept: $x^{(t)} \leftarrow \hat{x}^{(t)}$
8: **else**
9: Reject: $x^{(t)} \leftarrow x^{(t-1)}$
10: **end if**
11: **end for**

Metropolis–Hastings Example

Problem: Consider the Metropolis–Hastings algorithm for sampling from the target distribution

$$p(x) \propto \exp\left(-\frac{(x-2)^2}{2}\right) \quad \text{(Normal(2,1))}$$

We will use an asymmetric proposal distribution so that the full MH acceptance ratio must be used.

Simulate a single Markov chain of length 250, starting from $x_0 = 0$. Use the proposal

$$q(\hat{x} \mid x) = \begin{cases} \text{Normal}(x + 0.5,\ 0.5^2) & \text{with probability } 0.6 \\ \text{Normal}(x - 0.5,\ 0.5^2) & \text{with probability } 0.4 \end{cases}$$

This creates an *asymmetric proposal*, so the acceptance probability is

$$\alpha = \min\left(1,\ \frac{\tilde{p}(\hat{x})}{\tilde{p}(x)} \cdot \frac{q(x \mid \hat{x})}{q(\hat{x} \mid x)}\right)$$

Final Remarks on Metropolis–Hastings

The practical performance of the Metropolis–Hastings algorithm depends strongly on the choice of the proposal distribution.

Burn-in remains an important step, since early iterations may not represent the target distribution.

Thinning can be used to reduce autocorrelation or storage costs but with current CPU and GPU capabilities it is often better to keep all samples and thin later.

Finally, Metropolis–Hastings is part of a broader family of MCMC methods. There are other algorithms such as **Gibbs sampling**, **Hamiltonian Monte Carlo**, and **adaptive MCMC** build on the same core ideas and provide more efficient exploration in certain settings.

Homework: Burn-in and Thinning in Metropolis–Hastings

Problem: Consider the Metropolis–Hastings algorithm for sampling from the target distribution

$$p(x) \propto \exp\left(-\frac{x^2}{8}\right) \quad \text{(Normal(0,4))}$$

We will use an asymmetric proposal distribution so that the full MH acceptance ratio must be used.

1. Simulate a single Markov chain of length 200, starting from $x_0 = 5$. Use the proposal

$$q(\hat{x} \mid x) = \begin{cases} \text{Normal}(x+1,\, 1^2) & \text{with prob. } 0.7 \\ \text{Normal}(x-1,\, 1^2) & \text{with prob. } 0.3 \end{cases}$$

   This makes the proposal *asymmetric*, so the acceptance probability is

$$\alpha = \min\left(1,\ \frac{\tilde{p}(\hat{x})}{\tilde{p}(x)} \cdot \frac{q(x \mid \hat{x})}{q(\hat{x} \mid x)}\right)$$

2. Plot the resulting chain over iterations.
3. Highlight the first 40 iterations as burn-in (shade them in gray).
4. Apply thinning by retaining every 5th sample **after burn-in**, and mark the retained samples in red.