Chapter 4

Markov Chain Monte Carlo.
Metropolis Algorithm.

Ali Raisolsadat

School of Mathematical and Computational Sciences
University of Prince Edward Island

Markov Chains for Monte Carlo Estimation

We have been discussing inference using **Markov chains**.

- Concepts such as sampling and stationary distributions are central.
- For **discrete** Markov chains, dynamic programming algorithms can often be used for exact inference (e.g., Hidden Markov Models).

We can also use Markov chains for inference in more general settings.

- The most common framework is **Markov Chain Monte Carlo (MCMC)**.
- MCMC methods are used for **approximate inference**, particularly in complex Bayesian models (e.g., Bayesian logistic regression).

High-level idea of MCMC:

- We want to compute expectations with respect to a distribution $p(x)$, but we cannot generate independent samples directly from p.
- Construct a **homogeneous Markov chain** whose **stationary distribution** is $p(x)$.
- After a suitable burn-in period, use the generated samples $\{X_t\}_{t=1}^{T}$ in a Monte Carlo approximation.

Reminder: Homogeneous Markov Chains

Definition: A sequence of random variables $\{X_t\}_{t\geq 0}$ is a **Markov chain** if

$$P(X_{t+1} = j \mid X_t = i, X_{t-1}, \ldots, X_0) = P(X_{t+1} = j \mid X_t = i).$$

The process is called **homogeneous** (or time-homogeneous) if the transition probabilities do not depend on t:

$$P(X_{t+1} = j \mid X_t = i) = P_{ij}, \quad \text{for all } t.$$

Transition matrix:

$$\mathbf{P} = \begin{pmatrix} P_{11} & P_{12} & \cdots \\ P_{21} & P_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}, \quad P_{ij} \geq 0, \quad \sum_j P_{ij} = 1.$$

n-step transition: $P^{(n)} = \mathbf{P}^n$, with entries $P_{ij}^{(n)} = P(X_{t+n} = j \mid X_t = i)$.

Stationary Distribution

Definition: A probability vector $\boldsymbol{\pi}$ is a **stationary distribution** of a Markov chain if

$$\boldsymbol{\pi}^\top \mathbf{P} = \boldsymbol{\pi}^\top, \quad \sum_i \pi_i = 1,\ \pi_i \geq 0.$$

Interpretation:If $X_0 \sim \boldsymbol{\pi}$ then $X_t \sim \boldsymbol{\pi}$ for all t. The chain remains in equilibrium under $\boldsymbol{\pi}$.

Convergence: For many Markov chains

$$\lim_{t\to\infty} P(X_t = j \mid X_0 = i) = \pi_j$$

Irreducibility, Recurrence, and Ergodicity

Irreducibility: A Markov chain is **irreducible** if every state can be reached from every other state:

$$\forall i, j,\ \exists n \text{ such that } P_{ij}^{(n)} > 0$$

Recurrence: State i is **recurrent** if, starting from i, the probability of returning to i is 1:

$$P(\text{return to } i \mid X_0 = i) = 1$$

If the expected return time is finite, i is **positive recurrent**.

Ergodicity: An irreducible, aperiodic (gcd is 1 for all possible return times to a state), and positive recurrent Markov chain is called **ergodic**. For ergodic chains,

$$\lim_{t \to \infty} P(X_t = j \mid X_0 = i) = \pi_j$$

and time averages converge to expectations under π:

$$\frac{1}{T} \sum_{t=1}^{T} f(X_t) \xrightarrow{a.s.} E_\pi[f(X)]$$

Degenerate Example: “Pointless MCMC”

Consider estimating the expected value of a fair six-sided die. We know analytically that

$$E[X] = \frac{1+2+3+4+5+6}{6} = 3.5$$

Now suppose we design a “pointless” MCMC algorithm for this trivial problem:

- Start with an initial value $x_0 \in \{1, 2, 3, 4, 5, 6\}$, e.g. $x_0 = 4$.
- At each iteration t:
  - Roll the die to propose a new value $x' \sim \text{Uniform}\{1, 2, 3, 4, 5, 6\}$.
  - Generate a random number $u \sim \text{Uniform}[0, 1]$.
  - If $u < 0.5$, **accept** x', i.e. set $x_t = x'$.
  - Otherwise, **reject** x' and keep the old value, $x_t = x_{t-1}$.

This produces samples from a Markov chain with transition probabilities:

$$q(x_{t-1} \to x_t) = \frac{1}{2}\mathbf{1}(x_t = x_{t-1}) + \frac{1}{2} \cdot \frac{1}{6} = \begin{cases} \frac{7}{12}, & \text{if } x_t = x_{t-1}, \\ \frac{1}{12}, & \text{if } x_t \neq x_{t-1} \end{cases}$$

Pointless MCMC in Action

Let's simulate a few steps of the "pointless" MCMC algorithm.

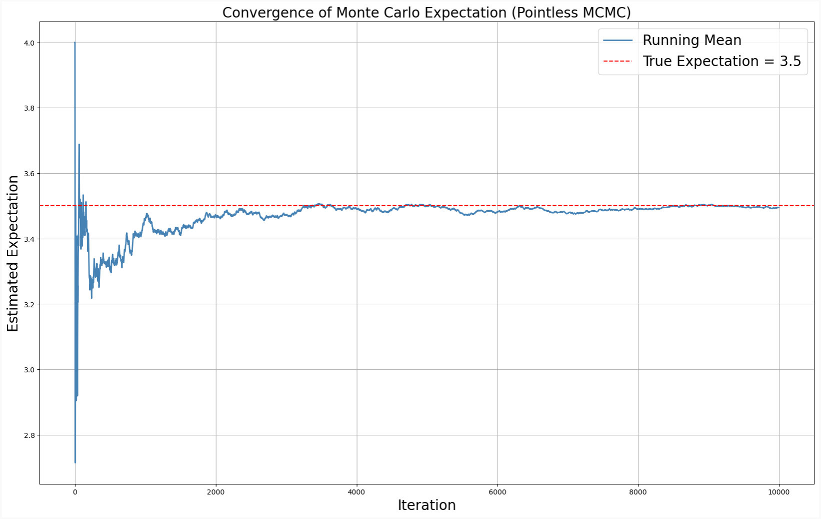
Setup: Start with $x_0 = 4$.

Step	Proposed Roll	u (Uniform(0,1))	Recorded Value (x_t)
1	6	0.234	6 (accepted)
2	3	0.612	6 (rejected)
3	2	0.523	6 (rejected)
4	3	0.125	3 (accepted)
5	2	0.433	2 (accepted)

Resulting samples:

$$x_{0:5} = (4, 6, 6, 6, 3, 2, \ldots)$$

Convergence to the Actual Expectation - Pointless MCMC in Action



Degenerate Example: “Pointless MCMC”

Observation:

- The chain sometimes repeats values due to rejections.
- Samples are **correlated**, even though the target distribution is uniform.

Key insight:

- If you run this chain long enough, you will spend roughly $\frac{1}{6}$ of the time on each outcome.
- The **stationary distribution** is uniform: if we start from a uniform state, either staying there or moving to a uniformly chosen new state keeps the distribution uniform.
- Thus, the stationary distribution of the chain is p (the uniform distribution).
- The property of constructing a chain whose stationary distribution is p is the **key idea behind all MCMC methods**.
- It is “pointless” here because we already know how to generate i.i.d. samples from p. If you can sample directly, you do *not* need MCMC.

Markov Chain Monte Carlo (MCMC)

Goal: Estimate expectations with respect to a distribution $p(x)$ when direct sampling is difficult.

Key idea:

- Construct a **Markov chain** whose **stationary distribution** is $\pi(x) = p(x)$.
  - After sufficient iterations ("burn-in"), samples $x^{(k)}$ approximately follow $p(x)$.
  - Notation: $x^{(1)}$ is the first sampled state, $x^{(2)}$ the second, ..., $x^{(n)}$ the nth.
- Use the dependent Markov chain samples in a Monte Carlo estimator:

$$\mathbb{E}_p[g(X)] \approx \frac{1}{n}\sum_{t=1}^{n} g\left(x^{(t)}\right).$$

- A generalization of the Law of Large Numbers, known as the **Ergodic Theorem**, ensures that as $n \to \infty$:

$$\frac{1}{n}\sum_{t=1}^{n} g\left(x^{(t)}\right) \xrightarrow{\text{a.s.}} \mathbb{E}_p[g(X)].$$

  - Convergence is slower than for i.i.d. samples, since the draws are **correlated**.
  - The variance of the estimator is typically larger than $\mathrm{Var}[g(X)]/n$.
- A widely used method to construct such chains is the Metropolis–Hastings algorithm.

Special Case: Metropolis Algorithm

Metropolis algorithm: Sampling from a continuous target distribution $p(x)$. We assume $p(x)$ can be evaluated up to a normalizing constant:

$$p(x) = \frac{\tilde{p}(x)}{Z}, \quad Z \text{ unknown.}$$

Algorithm:

1. Initialize $x^{(0)}$.
2. Until we get bored:
    - 2.1 Add zero-mean Gaussian noise to generate proposal $\hat{x}^{(t)}$
    $$\hat{x}^{(t)} = x^{(t-1)} + \epsilon, \quad \epsilon \sim N(0, \sigma^2)$$
    - 2.2 Generate $u \sim \text{Uniform}[0, 1]$
    - 2.3 Accept the proposal if
    $$u \le \frac{\tilde{p}(\hat{x}^{(t)})}{\tilde{p}(x^{(t-1)})} \; \frac{\text{probability of proposed}}{\text{probability of current}}$$
    and set $x^{(t)} = \hat{x}^{(t)}$.
    - 2.4 Otherwise, reject the proposal and set $x^{(t)} = x^{(t-1)}$

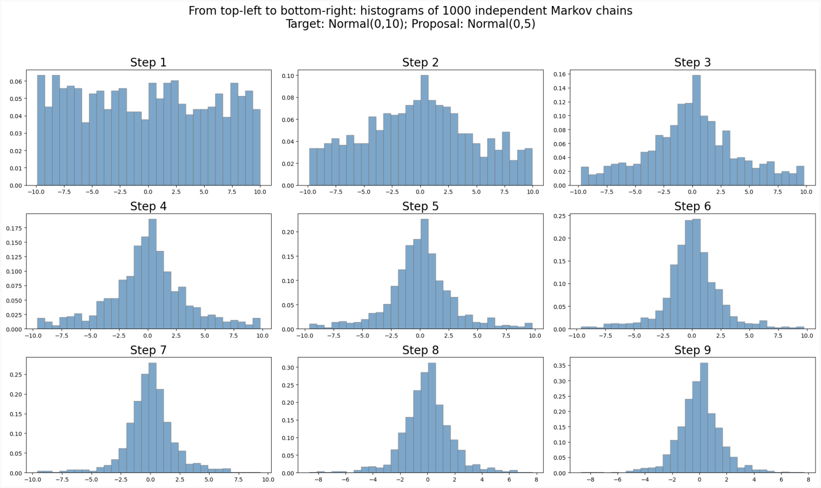
- Proposals that increase the target density are always accepted.
- Proposals that decrease the target density may be accepted or rejected.
- Under mild conditions (irreducibility, aperiodicity), the chain converges to $p(x)$, but convergence may be slow.
- Works even when the normalizing constant Z is unknown.

Metropolis Algorithm

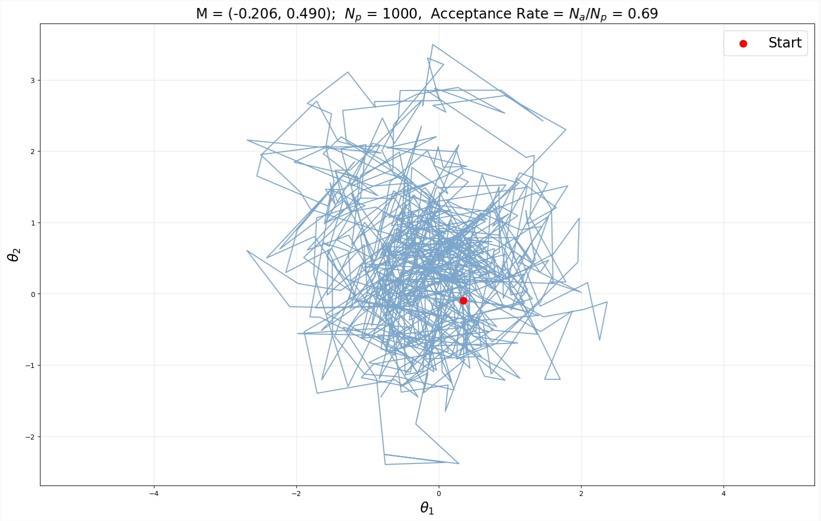
Algorithm 1 Metropolis Algorithm for Sampling from $p(x)$

```
Require: Initial state $x^{(0)}$, number of iterations $T$, proposal standard deviation $\sigma$
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:     Propose $\hat{x}^{(t)} = x^{(t-1)} + \epsilon$, with $\epsilon \sim N(0, \sigma^2)$
 4:     Generate $u \sim \text{Uniform}(0, 1)$
 5:     if $u \leq \frac{\tilde{p}(\hat{x}^{(t)})}{\tilde{p}(x^{(t-1)})}$ then
 6:         Accept the proposal: $x^{(t)} \leftarrow \hat{x}^{(t)}$
 7:     else
 8:         Reject the proposal: $x^{(t)} \leftarrow x^{(t-1)}$
 9:     end if
10: end for
```

Metropolis Algorithm - Convergence from Uniform to Target Distribution



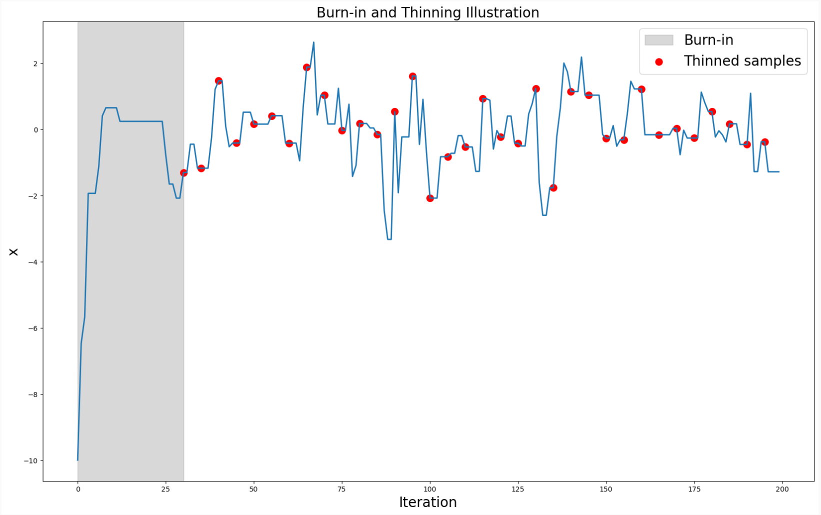
Metropolis Algorithm 2D



MCMC Implementation Issues

- In practice, we often do not *use all* the samples in our Monte Carlo estimates.
- Burn-in: discard the early samples while the chain is far from the stationary distribution.
- Thinning: keep only every k-th sample to reduce autocorrelation between consecutive samples.
- Two common approaches for applying MCMC:
  1. Run a large number of independent chains for a short time and use final states:
     - Highly parallelizable.
     - Effectively an extreme form of thinning: only one sample per chain is used.
     - Must ensure each chain has reached the stationary distribution (burn-in).
  2. Run a single chain for a long time and use states across time:
     - Less concern about burn-in if the chain is sufficiently long.
     - Thinning may be needed to reduce autocorrelation among samples.
- Diagnosing whether the chain has reached the stationary distribution can be difficult in practice.

Visualization of Burn-in and Thinning in MCMC



Homework: Burn-in and Thinning in MCMC

Problem: Consider the Metropolis algorithm for sampling from a target distribution

$$p(x) \propto \exp(-0.05x^2) \quad \text{(Normal(0,10))}.$$

1. Simulate a single Markov chain of length 200, starting from $x_0 = -10$, using a Gaussian random walk proposal with standard deviation 2.
2. Plot the chain over iterations.
3. Highlight the first 30 iterations as burn-in (shaded in gray).
4. Apply thinning by keeping every 5th sample **after burn-in** and mark these samples in red on the plot.
5. In a few sentences, explain why burn-in and thinning are used in MCMC.
6. **Optional / Bonus: Multiple Chains**
   - Run 1000 independent chains, each starting from a random value in [-10,10].
   - Apply burn-in and thinning as before.
   - Combine all thinned samples from all chains and plot a histogram.
   - Comment on how the histogram approximates the target distribution.