Computer Intensive Statistics and Applications
Chapter 1 and 2: Random Variable Generation

Shaobo Jin

Department of Mathematics

Statistical Model

In statistical inference,

- the possible observations are from a sample space $\mathcal{X}$,
- observation x is a realization of a random variable/vector X.

Statistical inference is concerned with methods of using the observations to obtain information about the distribution of X.

A statistical model is a class of possible probability measures $\mathcal{P}$ on the sample space $\mathcal{X}$, parameterized by parameter(s) θ:

$$\mathcal{P} = \{\mathrm{P}_\theta : \theta \in \Theta\},$$

where Θ is the parameter space.

Simulate Data

A sample $X = (X_1, ..., X_n)$ is a collection of independent random variables where X_i is distributed according to a distribution $\mathrm{P}_{i,\theta}$.

- We can view data as "simulated" from P_θ and we would like to know how data are generated by P_θ.
- This is similar to conducting many experiments in order to collect data.

Once we can simulate data from P_θ, we will be able to perform various simulation-based inference:

- estimate unknown quantity (e.g., Monte Carlo methods),
- quantify uncertainty (e.g., bootstrap).

Some Examples

Some examples that you should be able to solve by the end of the course:

- approximate expectations $\mathrm{E}[X] = \int x f(x)\, dx$, if the closed form expression is difficult/impossible to find,
- approximate integrals $\int f(x, y)\, dx$,
- simulate a stochastic process,
- approximate distributions of functions of random variables $h(X)$,

among others.

The focus of our course is neither 100% programming nor 100% math. We will show you

- how simulations are performed,
- why simulations work or do not work.

Uniform Distribution

We start with the fundamental component: random number generation.

Definition (Definition 1.2)

A uniform pseudo-random number generator is an algorithm which, starting from an initial value u_0 and a transformation D, produces a sequence with $u_i = D^i(u_0)$ in $[0, 1]$. For all n, $(u_1, ..., u_n)$ reproduce the behavior of an i.i.d. sample of uniform random variables when compared through a set of tests.

Congruential Generators

To generate Uniform $[0, 1]$, we can use the classic congruential generator (a, b, M) on $\{0, ..., M-1\}$.

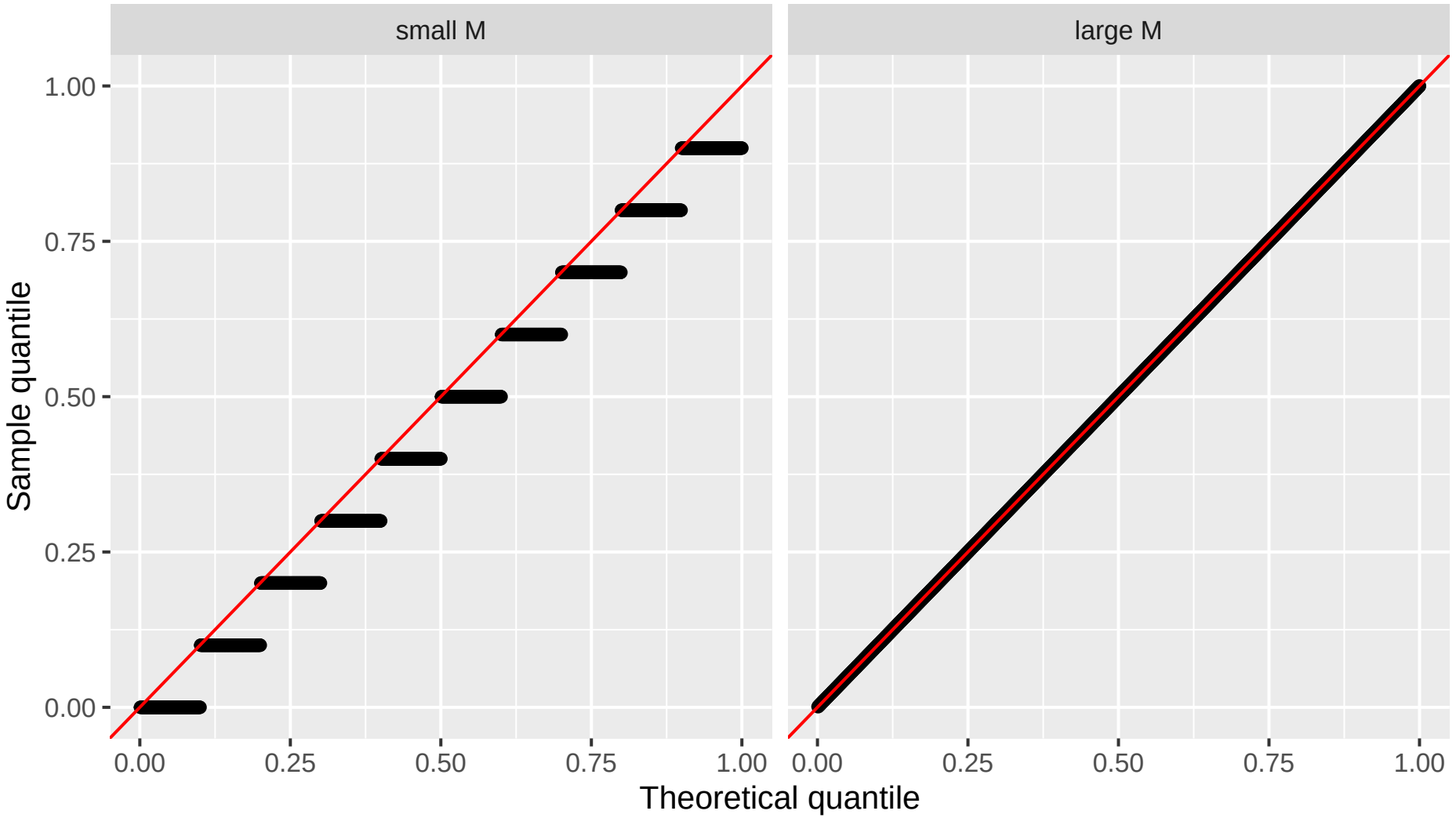
Algorithm 1: Generate pseudo random numbers from U $[0, 1]$

1 Choose a large $M > 0$ and (a, b) from $\{0, ..., M-1\}$;
2 Choose a seed x_0 ;
3 **for** $i = 1$ *in* $1 : n$ **do**
4 $\quad x_i = (ax_{i-1} + b) \mod M$;
5 **end**
6 Dividing the sequence by M gives a sequence that are regarded as random numbers from Uniform $[0, 1]$.

The operator $y = z \mod M$ means that, for an integer z, there is an integer k such that $y = kM + z$ but $0 \leq y < M$.

- If z is positive, then it is just the remainder.

Illustration of QQ plot



Expectation and Variance

Suppose that we generated $(x_1, ..., x_M)$ from a congruential generator (a, b, M). Then, the ordered outcome must be

$$\begin{bmatrix} 0 & \frac{1}{M} & \cdots & \frac{M-1}{M} \end{bmatrix}.$$

The sample mean is

$$\frac{1}{M}\sum_{i=1}^{M} x_i = \frac{1}{M}\sum_{j=0}^{M-1} \frac{j}{M} = \frac{M-1}{2M} \to \frac{1}{2},$$

and the sample variance with denominator M is

$$\frac{1}{M}\sum_{i=1}^{M} x_i^2 - \left(\frac{1}{M}\sum_{i=1}^{M} x_i\right)^2 = \frac{M^2-1}{12M^2} \to \frac{1}{12},$$

as M increases.

Quantile

In this course, we will mostly assume that we can generate Uniform $[0, 1]$ and study how we can generate random numbers of other distributions.

Definition (Definition 1.1)

For a distribution function F on $\mathbb{R}$, the generalized inverse F^- of F is the function defined by

$$F^-(u) = \inf\{x;\ F(x) \geq u\} \quad \text{for } 0 \leq u \leq 1.$$

The generalized inverse is often called the quantile function, denoted by Q.

Example

Let $X \sim \text{Exp}(\lambda)$ with mean $1/\lambda$. Find F^-.

Quantile Method

Lemma (Lemma 1.1)

If $U \sim$ *Uniform* $[0,1]$, *then the random variable* $F^{-}(U)$ *has distribution* F.

This lemma means that as long as we can generate $U \sim \text{Uniform}\,[0,1]$, we can obtain $X \sim F$ by letting $X = F^{-}(U)$.

Example

For exponential distribution, we can generate data via

$$X = -\frac{\log(1-U)}{\lambda}.$$

Transformation Method

The quantile method is a special case of the transformation method that means that, if we are interested in Y and we know that $Y = h(X)$, then we can generate random numbers for X and $h(X)$ will be the random numbers for Y.

Example

Box-Muller algorithm for normal distribution: $U_1 \sim \text{Uniform}[0,1]$, $U_2 \sim \text{Uniform}[0,1]$, and U_1 is independent of U_2. Then,

$$\begin{aligned} X_1 &= \sqrt{-2\log(U_1)}\cos(2\pi U_2) \sim N(0,1), \\ X_2 &= \sqrt{-2\log(U_1)}\sin(2\pi U_2) \sim N(0,1), \end{aligned}$$

and X_1 is independent of X_2.

Conditional Method

Suppose that we can partition X into $X = (X_1, X_2, ..., X_d)$ such that

$$p(x_1, ..., x_d) = p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_1, x_2) \cdots p(x_d \mid x_1, ..., x_{d-1}).$$

Then, we can generate X_1, $X_2 \mid X_1$, $X_3 \mid (X_1, X_2)$, ..., $X_d \mid (X_1, ..., X_{d-1})$ in order to generate X.

Example

Suppose that $Y \sim \text{Bernoulli}(\theta)$, where $\log\left(\frac{\theta}{1-\theta}\right) = a + bX$ with $X \sim N(0,1)$. To generate (X, Y), we generate

$$\begin{aligned} X &\sim N(0,1), \\ Y \mid X = x &\sim \text{Bernoulli}\left(\frac{\exp\{a+bx\}}{1+\exp\{a+bx\}}\right). \end{aligned}$$

Dominance

Suppose that we want to generate a sample with target density $p(x)$ but we know that it is easier to generate random numbers from another instrumental density $g(x)$. Let M be a constant such that

$$p(x) \leq Mg(x), \quad \text{for all } x.$$

Example

Let $X \sim \text{Beta}(a, b)$ with density $p(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}$, where $x \in (0, 1)$. We focus on $a > 1$ and $b > 1$, and let $g(x)$ be the uniform density.

Rejection Algorithm

Algorithm 2: Rejection algorithm

Choose a seed (omitted in all following algorithms) ;
for $i = 1$ *in* $1 : n$ **do**
  while *new trial is needed* **do**
    Draw $X = x$ from $g(\cdot)$;
    Compute the ratio $r(x) = \frac{p(x)}{Mg(x)}$;
    Draw $U = u$ from Uniform $[0, 1]$;
    If $u > r(x)$, start a new trial. Otherwise, let $x^{(t+1)} = x$;
  end
end

The algorithm works even when x is multidimensional.

But why does the rejection algorithm work?

Effect of M

- The probability of accepting a candidate is

$$\mathrm{P}\left(U \leq r\left(X\right)\right) \quad = \quad \frac{1}{M}.$$

  For large M, the proportion of useless draws that are discarded is large.
- But if M is not big enough, then we cannot guarantee $p\left(x\right) \leq Mg\left(x\right)$ for all x.

A Variant

Sometimes it is not easy to get the exact expression of $p(x)$, but it is easy to get $f(x)$ where $p(x) = cf(x)$.

- That is, the normalizing constant c is often hard to obtain, such as in a posterior distribution.

Instead of using $p(x)$ (hard to get) in the accept-reject algorithm, we can use $f(x)$ (easy to get) instead.

Algorithm 3: Rejection algorithm

```
for i = 1 in 1 : n do
    while new trial is needed do
        Draw X = x from g(·) ;
        Compute the ratio r(x) = f(x) / (M g(x)) ;
        Draw U = u from Uniform[0, 1] ;
        If u > r(x), start a new trial. Otherwise, let x^(t+1) = x ;
    end
end
```

Another Variant: Envelope Accept-Reject Method

Suppose that $p(x)$ can be computed by can be time consuming. We can find an extra lower bound $g_l(x)$ such that

$$g_l(x) \leq p(x) \leq Mg(x), \quad \text{for all } x.$$

- In the accept-reject algorithm, we accept the candidate value if

$$u \leq \frac{p(x)}{Mg(x)}.$$

- In the envelop accept-reject algorithm, we first check if

$$u \leq \frac{g_l(x)}{Mg(x)}.$$

Envelope Accept-Reject Method

Algorithm 4: Envelope accept-reject algorithm

```
for i = 1 in 1 : n do
    while new trial is needed do
        Draw X = x from g(·) ;
        Draw U ~ Uniform[0, 1] ;
        if U ≤ g_l(x) / (M g(x)) then
            Accept X ;
        else
            if U ≤ p(x) / (M g(x)) then
                Accept X;
            else
                Start a new trial ;
            end
        end
    end
end
```

Markov Property and Markov Chain

Another idea of generating random number is through a Markov chain.

Definition

Let $X_i \in \Omega$ for all i, where the set Ω is the state space. The random variables satisfy the Markov property if

$$\mathrm{P}\left(X_{i+1} \in A \mid X_j = x_j, 0 \leq j \leq i\right) \quad = \quad \mathrm{P}\left(X_{i+1} \in A \mid X_i = x_i\right).$$

Definition

A Markov chain is a sequence of random variables $X_i \in \Omega$ that satisfy the Markov property.

Transition Kernel

Consider a sequence of discrete random variables $\{X_n\}$. The Markov property means that

$$\mathrm{P}\left(X_n = y \mid X_{n-1} = x, X_{n-2}, \cdots X_0\right) \quad = \quad \mathrm{P}\left(X_n = y \mid X_{n-1} = x\right).$$

In this case, a transition kernel is a matrix $\boldsymbol{K}$ with elements $\mathrm{P}\left(X_n = y \mid X_{n-1} = x\right)$.

Consider a sequence of continuous random variables $\{X_n\}$. The Markov property means that

$$\mathrm{P}\left(X_n \in A \mid X_{n-1} = x, X_{n-2}, \cdots X_0\right) \quad = \quad \int_{y \in A} K\left(x, y\right) dy,$$

$$p\left(X_n = y \mid X_{n-1} = x, X_{n-2}, \cdots X_0\right) \quad = \quad p\left(X_n = y \mid X_{n-1} = x\right) = K\left(x, y\right),$$

where the transition kernel $K\left(x, y\right)$ is the conditional density of Y given $X = x$.

Discrete Marginal Distribution: Time 1

Take the sequence of discrete random variables $\{X_n\}$ as an example. Suppose that each X_n takes values from 1, ..., M. Then,

$$\mathrm{P}\left(X_1 = j\right) \quad = \quad \sum_{i=1}^{M} \mathrm{P}\left(X_0 = i\right) \mathrm{P}\left(X_1 = j \mid X_0 = i\right).$$

Expressing it as a matrix, we have

$$\boldsymbol{p}_1 \quad = \quad \boldsymbol{p}_0 \boldsymbol{K},$$

where $\boldsymbol{p}_k$ is a row vector with tth element $\mathrm{P}\left(X_k = t\right)$, and the transition kernel $\boldsymbol{K}$ is a $M \times M$ matrix with (i, j)th element $\mathrm{P}\left(X_1 = j \mid X_0 = i\right)$.

Discrete Marginal Distribution: Time n

If we consider a homogenous Markov chain, where

$$\mathrm{P}\left(X_i = y \mid X_{i-1} = x\right) \quad = \quad \mathrm{P}\left(X_1 = y \mid X_0 = x\right)$$

for all i, then

1. At time $t = 2$,

$$\begin{aligned} \mathrm{P}\left(X_2 = j\right) \quad &= \quad \sum_{i=1}^{M} \mathrm{P}\left(X_1 = i\right) \mathrm{P}\left(X_2 = j \mid X_1 = i\right) \\ \Rightarrow \quad \boldsymbol{p}_2 \quad &= \quad \boldsymbol{p}_1 \boldsymbol{K} = \boldsymbol{p}_0 \boldsymbol{K}^2. \end{aligned}$$

2. In general,

$$\begin{aligned} \mathrm{P}\left(X_n = j\right) \quad &= \quad \sum_{i=1}^{M} \mathrm{P}\left(X_{n-1} = i\right) \mathrm{P}\left(X_n = j \mid X_{n-1} = i\right) \\ \Rightarrow \quad \boldsymbol{p}_n \quad &= \quad \boldsymbol{p}_{n-1} \boldsymbol{K} = \boldsymbol{p}_0 \boldsymbol{K}^n. \end{aligned}$$

Stationary Distribution

Definition

The distribution π on Ω is a stationary distribution (or invariant distribution) of the Markov chain with the transition kernel K, if

$$\pi(y) = \sum_{x \in \Omega} \pi(x) K(x, y), \quad \text{discrete case,}$$

where $\pi(x) = \mathrm{P}(X = x)$, or

$$\pi(y) = \int_{x \in \Omega} \pi(x) K(x, y) \, dx, \quad \text{continuous case,}$$

where $\pi(x) = p(x)$.

- The stationary distribution means that if the initial state $X_0 \sim \pi$, then $X_n \sim \pi$ for all $n \geq 0$, the same distribution.

Stationary Distribution: Existence

The stationary distribution does not always exists. Our goal in MCMC is to design the algorithm such that the stationary distribution is the distribution that we want to sample random numbers from.

Example

1. The stationary distribution of $\boldsymbol{K} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$ is $\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$.
2. But if we let $\Omega = \mathbb{Z}$ and consider

$$\begin{aligned} \mathrm{P}\left(X_n = i - 1 \mid X_{n-1} = i\right) &= \frac{1}{4}, \\ \mathrm{P}\left(X_n = i + 1 \mid X_{n-1} = i\right) &= \frac{3}{4}, \end{aligned}$$

then the stationary distribution does not exist.

Irreducible Markov Chain

From now on, we consider the finite-state Markov chain for simplicity.

- The states x and y communicate if there exists an integer $n \geq 0$ such that $\mathrm{P}(X_n = x \mid X_0 = y) > 0$ and $\mathrm{P}(X_n = y \mid X_0 = x) > 0$.
- The transition matrix $\boldsymbol{K}$ is irreducible if x communicates with y whenever $x, y \in \Omega$.

Example

$$\boldsymbol{K} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \end{bmatrix}$$ is reducible and $$\boldsymbol{K} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.25 & 0.25 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0.25 & 0 & 0.25 & 0.5 \end{bmatrix}$$ is irreducible.

Periodic

For a transition matrix, the state w is periodic with period $t > 1$ if

1. $\mathrm{P}(X_n = w \mid X_0 = w) > 0$ only holds for $n = tm$ with integer $m > 0$ and
2. t is the largest number of $\{2, 3, ...\}$ for which 1) holds.

The state w is aperiodic if $\mathrm{P}(X_n = w \mid X_0 = w) > 0$ for all n.

Example

$$\boldsymbol{K} = \begin{bmatrix} 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \end{bmatrix} \text{ has period } t = 2.$$

Long-Run Property: Finite-State Markov Chain

Theorem

Consider a finite-state Markov chain. Suppose that the transition matrix $\boldsymbol{K}$ *is irreducible, aperiodic, and has stationary distribution* π. *Then, for all starting state* $w_0 \in \Omega$,

$$\lim_{n\to\infty} P\left(X_n = w \mid X_0 = w_0\right) \quad = \quad \pi\left(w\right).$$

- The stationary distribution is reached after large enough iterations. When we simulate random numbers using a Markov chain, we need to discard the early iterations. The discarded ones are called the burn-in period.
- Samples obtained after the burn-in period can be considered as samples from π.

Choose the Transition Kernel

Our goal is to simulate data from some distribution with density $p(x)$. Now the question becomes how to choose K such that the stationary distribution is $\pi(x) = p(x)$.

Fact

If $p(x)$ and $K(x, y)$ satisfies the detailed balance condition, i.e,

$$K(x, y)\, p(x) \quad = \quad K(y, x)\, p(y),$$

for any $x, y \in \Omega$, then $p(x)$ is the stationary distribution of the Markov chain with the transition kernel K.

Proposal Distribution

When we simulate random numbers from a Markov chain, we need a proposal distribution

$$T(x, y) = p(Y = y \mid X = x).$$

Find a proposal distribution $T(x, y)$ that satisfies the detailed balance condition is difficult.

- So with probability $A(x, y)$ we let $X_{n+1} = y$ (accept), and probability $1 - A(x, y)$ we let $X_{n+1} = x$ (reject).
- For $X_{n+1} \neq x$, the transition is

$$K(x, y) = T(x, y) A(x, y).$$

Hence, we should seek A such that the detailed balance condition is fulfilled.

Detailed Balance: Symmetric Proposal

Start with the case where the proposal distribution is symmetric, i.e., $T(x, y) = T(y, x)$.

Example

$Y \mid X = x \sim N\left(x, \sigma^2\right)$ is symmetric, since

$$T(x, y) = \frac{1}{\sqrt{2\sigma^2}} \exp\left\{-\frac{(x-y)^2}{2\sigma^2}\right\}$$

The transition kernel is

$$K(x, y) = T(x, y) A(x, y) + [1 - A(x, y)] \delta_x(y),$$

where $\delta_x(y) = 1$ if $x = y$, and 0 otherwise. For $y \neq x$, the detailed balance condition reduces to

$$\pi(x) A(x, y) = \pi(y) A(y, x).$$

Deriving $A(x, y)$: Symmetric Proposal

The detailed balance condition is fulfilled, if we choose

$$A(x, y) = \lambda(x, y)\,\pi(y), \quad A(y, x) = \lambda(x, y)\,\pi(x),$$

such that $\lambda(x, y) = \lambda(y, x)$. Since $A(\cdot, \cdot)$ is a probability, we must have

$$\lambda(x, y) \leq \min\left\{\frac{1}{\pi(y)}, \frac{1}{\pi(x)}\right\}.$$

The maximum of $A(\cdot, \cdot)$ with respect to λ is attained at

$$\lambda(x, y) = \min\left\{\frac{1}{\pi(y)}, \frac{1}{\pi(x)}\right\}.$$

Hence,

$$A(x, y) = \lambda(x, y)\,\pi(y) = \min\left\{1, \frac{\pi(y)}{\pi(x)}\right\}.$$

Metropolis Algorithm

Algorithm 5: Metropolis Algorithm

Choose an initial state $x^{(0)}$;

for $t = 1$ *in* $1 : n$ **do**

  Sample a candidate y from $T\left(x^{(t)}, y\right)$;

  Calculate the ratio $R\left(x^{(t)}, y\right) = \frac{\pi(y)}{\pi\left(x^{(t)}\right)}$;

  Draw $U \sim \mathrm{U}[0,1]$;

  Update

$$x^{(t+1)} = \begin{cases} y, & \text{if } U \leq R\left(x^{(t)}, y\right), \\ x^{(t)}, & \text{otherwise.} \end{cases}$$

end

Metropolis Algorithm: Example

Since we have $R\left(x^{(t)}, y\right) = \frac{\pi(y)}{\pi\left(x^{(t)}\right)}$ in the Metropolis algorithm, we only need to know $\pi\left(\right)$ up to a normalizing constant.

Example

Suppose that we want to sample random numbers from a distribution with density

$$p(x) \quad = \quad c \exp\left(-x^2\right)\left[2 + \sin(5x) + \sin(2x)\right],$$

where c is a normalizing constant.

Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm allows asymmetric proposal distributions, as long as $T(x, y) > 0$ if and only if $T(y, x) > 0$.

Algorithm 6: Metropolis-Hastings Algorithm

Choose an initial state $x^{(0)}$;
for $t = 1$ *in* $1 : n$ **do**
    Sample a candidate y from $T\left(x^{(t)}, y\right)$;
    Calculate the ratio $R\left(x^{(t)}, y\right) = \frac{\pi(y)T\left(y, x^{(t)}\right)}{\pi\left(x^{(t)}\right)T\left(x^{(t)}, y\right)}$;
    Draw $U \sim \mathrm{U}[0, 1]$;
    Update

$$x^{(t+1)} = \begin{cases} y, & \text{if } U \leq R\left(x^{(t)}, y\right), \\ x^{(t)}, & \text{otherwise.} \end{cases}$$

end

Examples of MCMC Algorithms

Many different MCMC algorithms differ mainly in how the candidate y is sampled.

- In the random-walk Metropolis algorithm, $y = x^{(t)} + \epsilon$, where ϵ is sampled from some distribution, e.g., Uniform $[-a, a]$, Normal, etc.
- In multiplicative random walk, $y = x^{(t)}\epsilon$.
- In independence sampler, y is sampled from $g(y)$ that does not depend on $x^{(t)}$.
- In hit and run, we first sample a direction $\rho^{(t)}$ and then a random length $\lambda^{(t)}$. Then $y = x^{(t)} + \lambda^{(t)}\rho^{(t)}$.
- The Langevin Metropolis-Hastings algorithm explores the shape of the target distribution by $y = x^{(t)} + d^{(t)} + \sigma\epsilon$, where $\epsilon \sim N(0, I)$ and

$$d^{(t)} = \frac{\sigma^2}{2}\frac{\partial \log \pi\left(x^{(t)}\right)}{\partial x}.$$

Gibbs Sampler: Conditioning

It can be the case that it is not easy to sample from the joint distribution of $X \in \mathbb{R}^d$. But it is easy to sample from the conditional distributions.

- Suppose that $X = (X_1, ..., X_p)$, where $X_i \in \mathbb{R}^{d_i}$.
- Let $\pi_{i|-i}(X_i \mid X_{-i})$ be the conditional distribution of X_i given X_{-i}, where $X_{-i} = \begin{pmatrix} X_1 & \cdots & X_{i-1} & X_{i+1} & \cdots & X_p \end{pmatrix}$.

Algorithm 7: Gibbs Sampler

Choose an initial state $x^{(0)}$;
for $t = 1$ *in* $1 : n$ **do**
  for $i = 1$ *in* $1 : p$ **do**
    Draw $X_i^{(t+1)} \sim \pi_{i|-i}\left(X_i \mid X_1^{(t+1)}, ..., X_{i-1}^{(t+1)}, X_{i+1}^{(t)}, ..., X_p^{(t)}\right)$;
  end
end

Gibbs Sampler: Example

Example

Suppose that we can generate data from $N\left(\mu, \sigma^2\right)$. For a bivariate normal distribution $\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}\right)$, we know that

$$X_2 \mid X_1 = x_1 \quad \sim \quad N\left(\mu_2 + \rho\sigma_1^{-1}\sigma_2\left(x_1 - \mu_1\right), \sigma_2^2\left(1 - \rho^2\right)\right).$$

Sample random numbers from the bivariate normal with Gibbs sampler.

Why Does Gibbs Sampler Work?

The transition kernel $K(x, y)$ is

$$\pi_{1|-1}(y_1 \mid x_2, ..., x_p)\, \pi_{2|-2}(y_2 \mid y_1, x_3, ..., x_p) \cdots \pi_{p|-p}(y_p \mid y_1, ..., y_{p-1}).$$

If we consider the continuous random variables, the transition kernel satisfies

$$\pi(y) = \int K(x, y)\, \pi(x)\, dx,$$

where $\pi(x)$ is the marginal distribution. Here $\pi()$ is not treated as a generic symbol.

Gibbs Sampler: Another Example

Gibbs sampler simulates random number from "correct" conditional distribution. But it can work poorly if the correlations between components are high, especially when the conditional distribution has a low variance so that the step size becomes small. The same issue can occur to Metropolis-Hastings as well.

Example

Suppose that we can generate data from $N\left(\mu, \sigma^2\right)$. For a bivariate normal distribution $\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}\right)$, we know that

$$X_2 \mid X_1 = x_1 \quad \sim \quad N\left(\mu_2 + \rho\sigma_1^{-1}\sigma_2\left(x_1 - \mu_1\right), \sigma_2^2\left(1-\rho^2\right)\right).$$

Sample random numbers from the bivariate normal with Gibbs sampler when $\rho = 0.99$.

Burn-In Period

The stationary distribution is reached after large enough iterations.

- If the iterations have not proceeded long enough, the simulated numbers may be unrepresentative of the target distribution.

To diminish the influence of the starting values, we can discard the early simulations, known as the burn-in.

- There is no golden standard on how long the burn-in period should be.

Serial Correlation

It is obvious that X_{n+1} and X_n are not independent draws. Inference from autocorrelated draws is generally less precise than from the same number of independent draws.

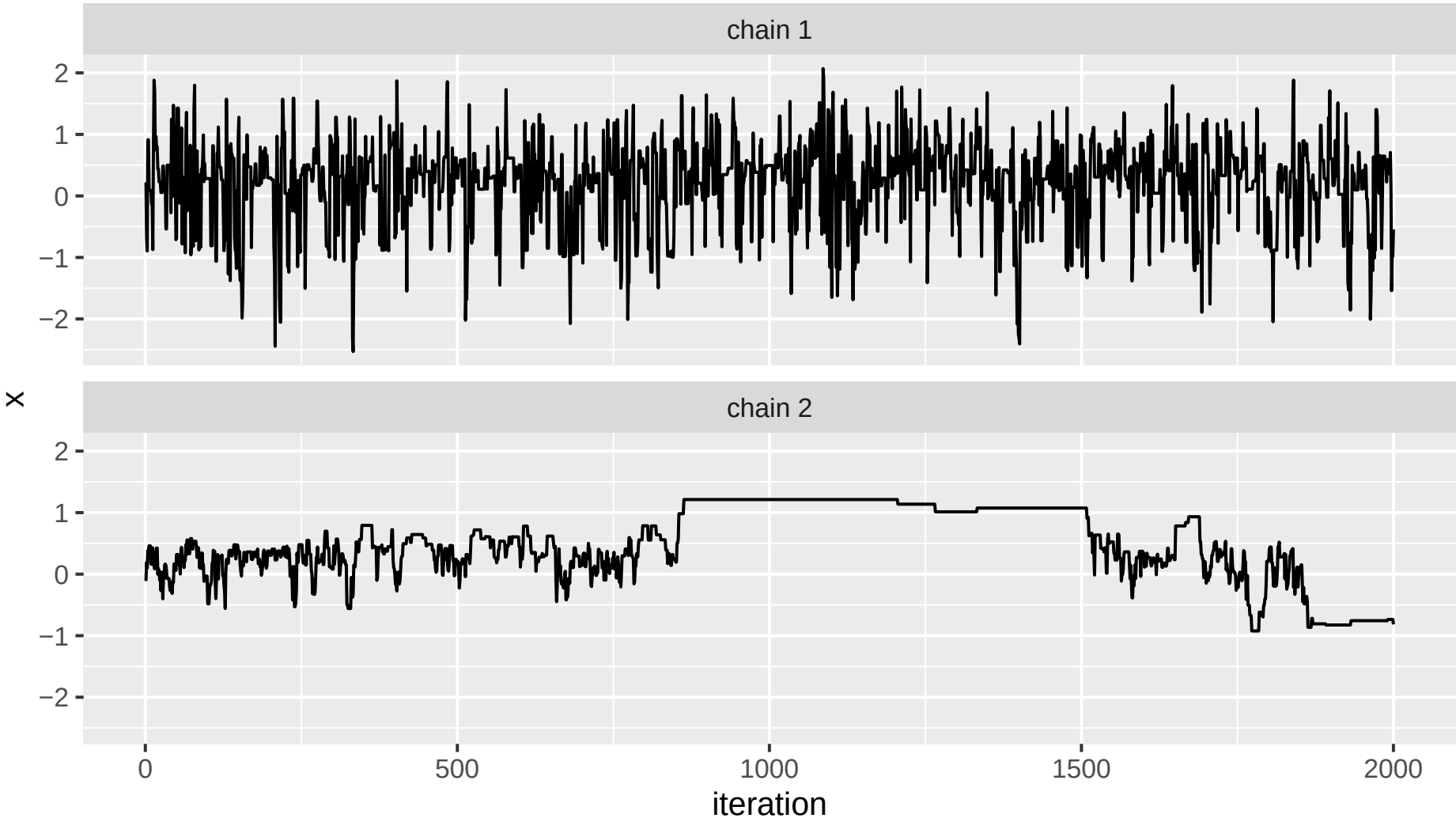
- However, such serial correlation is not necessarily a problem.
- Remember that, at convergence, we reach the stationary distribution.

Some prefer thinning the sequence by only keeping every kth draw from a sequence in order to reduce serial correlation.

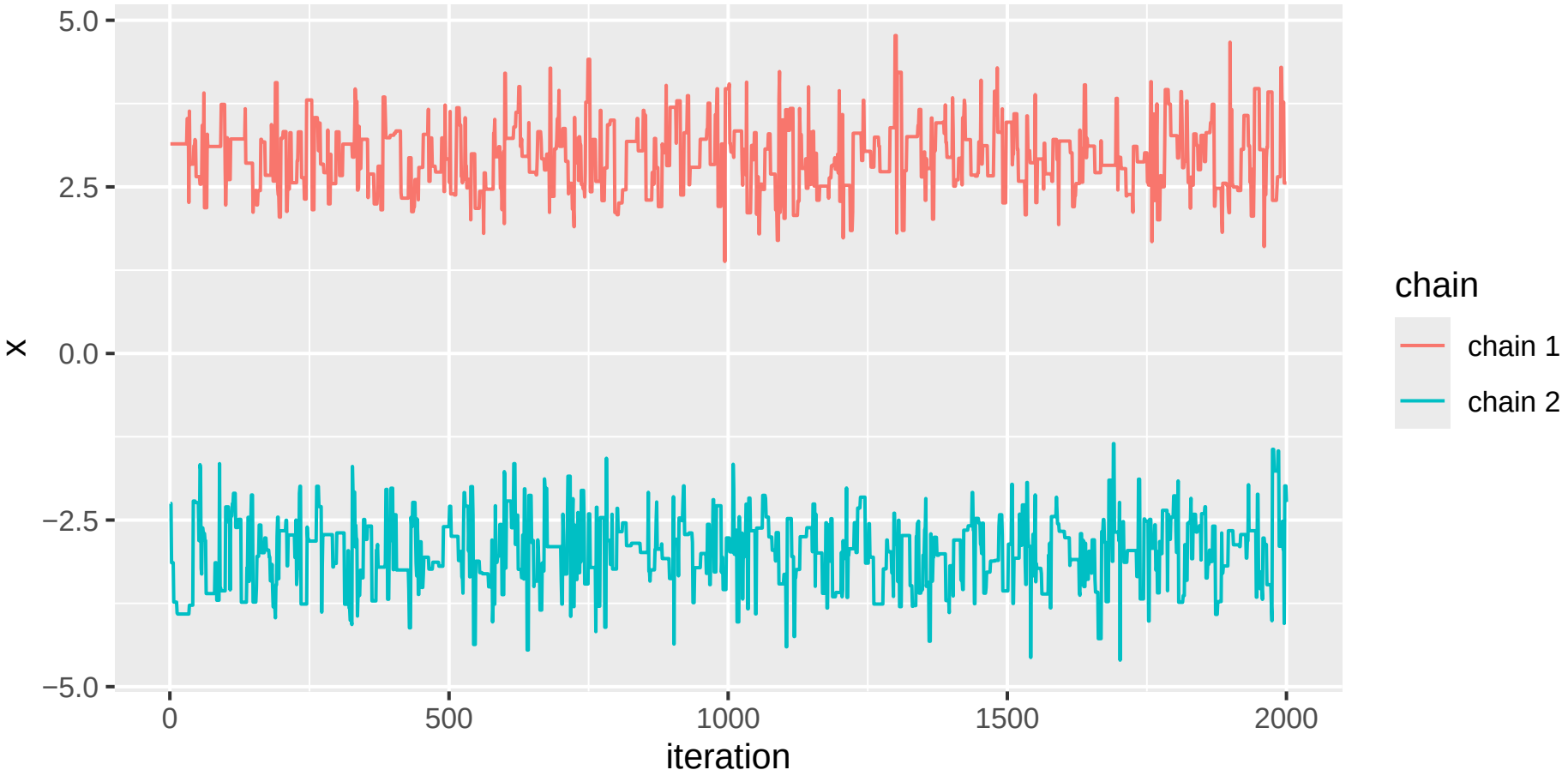
- But whether or not the Markov chain is thinned, it can be used for inferences, provided that it has reached convergence.

Mixing

We want the Markov chain to show good mixing.



Several Markov Chains From Same Model



One suggestion is to generate several independent Markov chains, starting from widely separated places.

Gelman-Rubin $\hat{R}$ Statistic: Variation

One way to assess convergence is the Gelman-Rubin $\hat{R}$ statistic. Suppose that we have simulated m chains each with n iterations after the burn-in period. Say we have a univariate statistic $y_{ij} = f(x_{ij})$, where x_{ij} is the ith value in the jth chain.

- The variation within the chains is measured by

$$W = \frac{1}{m}\sum_{j=1}^{m}\left[\frac{1}{n-1}\sum_{i=1}^{n}(y_{ij}-\bar{y}_{\cdot j})^2\right],$$

  where $\bar{y}_{\cdot j}$ is the average of $\{y_{ij}\}$.
- The variation between the chains is measured by

$$B = \frac{n}{m-1}\sum_{j=1}^{m}(\bar{y}_{\cdot j}-\bar{y}_{\cdot\cdot})^2,$$

  where $\bar{y}_{\cdot\cdot}$ is the average of all $\bar{y}_{\cdot j}$.

Gelman-Rubin $\hat{R}$ Statistic: Expression

Consider

$$\hat{V} = \frac{n-1}{n}W + \frac{1}{n}B.$$

- If the Markov chains have reached stationary as $n \to \infty$, then $\hat{V}$ is unbiased for $\mathrm{Var}(Y)$.
- $\mathrm{E}[W]$ approaches $\mathrm{Var}(Y)$ when $n \to \infty$.

The Gelman-Rubin $\hat{R}$ statistic is then

$$\hat{R} = \sqrt{\frac{\hat{V}}{W}},$$

which declines to 1 as $n \to \infty$.

- It is suggested that we keep simulating the Markov chain until $\hat{R} < 1.1$, or even 1.01.

Variants of Gelman-Rubin $\hat{R}$

Several different versions of $\hat{R}$ have been proposed.

1. One suggestion is to change $\hat{V}$ to

$$\hat{V} = \frac{n-1}{n}W + \frac{1}{n}\left(1+\frac{1}{m}\right)B.$$

   to account for the possibility that $\hat{V}$ is too low.

2. Another suggestion is to split each chain into two parts, yields $2m$ chains of length $n/2$ each. Then compute the $\hat{R}$, pretending that we have simulated $2m$ chains of length $n/2$.
   - This can be useful to detect the case where each chain does not reach stationary but the chains cover a common distribution, e.g, two chains exhibit an X-shape.

Geweke Diagnositic

Suppose that we have simulated a Markov chain $\{x_i\}$. The idea is that, the average of the first $100p_1\%$ observations should be close to the last $100p_2\%$ observations, where $p_1 + p_2 < 1$ to ensure that these two segments do not overlap.

Let $\bar{X}_i$ be the average of the ith segment. Then, the Geweke method computes

$$Z = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\text{Var}\left(\bar{X}_1\right) + \text{Var}\left(\bar{X}_2\right)}},$$

which converges to $N(0,1)$, as $n \to \infty$.

Hamiltonian Monte Carlo

- The Metropolis algorithm and the Gibbs sampler often move too slowly through the target distribution when the dimension of the target distribution is high.
- Hamiltonian Monte Carlo (HMC) moves much quicker through the target distribution.
  - For each component in the target distribution, HMC adds a momentum variable and the proposal distribution largely depends on the momentum variable.
  - Both the component in the target distribution and the momentum are updated in the MCMC algorithm.

Hamiltonian Dynamics

The idea of HMC originates from the Hamiltonian dynamics in physics.

- The state of a system consists of the position $x \in \mathbb{R}^d$ and the momentum $\phi \in \mathbb{R}^d$ of same dimension.
- The Hamiltonian is a function of x and ϕ, denoted by $H(x, \phi)$.
- The position and the momentum can change over time t. The change is described by the Hamilton's equations:

$$\frac{dx_i}{dt} = \frac{\partial H}{\partial \phi_i}, \quad \text{and} \quad \frac{d\phi_i}{dt} = -\frac{\partial H}{\partial x_i},$$

for $i = 1, ..., d$.

Potential and Kinetic Energy

For HMC, the Hamiltonian is usually

$$H(x, \phi) = U(x) + V(\phi),$$

where $U(x) = -\log \pi(x)$ is called the potential energy and $V(\phi)$ is called the kinetic energy.

- We want to sample from $\pi(x)$. Hence, ϕ is artificial.
- We often let $\phi \sim N(0, M)$, independent of X, for a prespecified covariance matrix M, and $V(\phi)$ the negative log density of ϕ.
- The Hamilton's equations become

$$\frac{dx}{dt} = M^{-1}\phi, \quad \text{and} \quad \frac{d\phi}{dt} = \frac{\partial \log \pi(x)}{\partial x},$$

arranged as column vectors.

Augmentation

Since X and ϕ are independent, their joint density is

$$\begin{aligned} p(x,\phi) = \pi(x)\,p(\phi) &= \exp\{-U(x) - V(\phi)\} \\ &= \exp\{-H(x,\phi)\}. \end{aligned}$$

We have augmented the problem from sampling X from $\pi(x)$ to sampling (X,ϕ) form $\exp\{-H(x,\phi)\}$.

1. We first sample ϕ from $N(0,M)$, independent of current x.
   - Since $\phi \sim N(0,M)$, we already sample ϕ from the desired distribution.
2. We then sample X, where the new state is proposed by Hamiltonian dynamics by solving the differential equations.

Solve Differential Equation

To solve the differential equations, we consider an approximation known as the leapfrog method. For some stepsize $\epsilon > 0$, we perform half-step updates as

$$\begin{aligned}
\phi\left(t+\frac{\epsilon}{2}\right) &= \phi(t)+\frac{\epsilon}{2}\frac{\partial\phi(t)}{\partial t}=\phi(t)+\frac{\epsilon}{2}\frac{\partial\log\pi(x(t))}{\partial x},\\
x(t+\epsilon) &= x(t)+\epsilon\frac{\partial x\left(t+\frac{\epsilon}{2}\right)}{\partial t}=x(t)+\epsilon M^{-1}\phi\left(t+\frac{\epsilon}{2}\right),\\
\phi(t+\epsilon) &= \phi\left(t+\frac{\epsilon}{2}\right)+\frac{\epsilon}{2}\frac{\partial\phi(t+\epsilon)}{\partial t}=\phi\left(t+\frac{\epsilon}{2}\right)+\frac{\epsilon}{2}\frac{\partial\log\pi(x(t+\epsilon))}{\partial x}.
\end{aligned}$$

Starting from $t=0$, we can get a trajectory at times $\{\epsilon, 2\epsilon, ..., L\epsilon\}$, and approximate the values for $x(L\epsilon)$ and $\phi(L\epsilon)$.

Leapfrog Method to Sample X

Suppose that the current state is (x, ϕ).

1. Update ϕ with a half-step update by

$$\phi \leftarrow \phi + \frac{\epsilon}{2} \frac{\partial \log \pi(x)}{\partial x}.$$

2. For $\ell = 1, ..., L-1$,
   1. Update the position: $x \leftarrow x + \epsilon M^{-1} \phi$.
   2. Update the momentum:

$$\phi \leftarrow \phi + \epsilon \frac{\partial \log \pi(x)}{\partial x}.$$

3. Make one last update on the position: $x \leftarrow x + \epsilon M^{-1} \phi$.
4. Make one last half-step update of the momentum

$$\phi \leftarrow \phi + \frac{\epsilon}{2} \frac{\partial \log \pi(x)}{\partial x}.$$

Metropolis Step

Suppose that the state after such L updates is (x^*, ϕ^*). We negate the momentum and the new proposal state is $(x^*, -\phi^*)$.

- We determine whether to accept the proposal using the Metropolis algorithm, where the acceptance probability is

$$A\left((x,\phi),(x^*,-\phi^*)\right) = \min\left\{1, \frac{\exp\left\{-H\left(x^*,-\phi^*\right)\right\}}{\exp\left\{-H\left(x,\phi\right)\right\}}\right\}.$$

- If the proposed state is accepted, then we accept x^* as a new state for x, but don't care about ϕ^*.
- No matter we accept or reject the proposal, we will draw a new momentum in the next iteration, independent of previous momentum.

Properties of HMC

Some crucial properties of the Hamiltonian dynamics for MCMC updates include

1. deterministic updates. The Hamiltonian dynamics is deterministic. After running the leapfrog loop L times, we always move the initial state (x_0, ϕ_0) to the same proposal (x^*, ϕ^*).
2. reversible. The mapping from the state at time t, denoted by $(x(t), \phi(t))$, to the state at time $t+s$, denoted by $(x(t+s), \phi(t+s))$, is one-to-one and has an inverse mapping. If we negate the momentum, we will come back from $(x(t+s), -\phi(t+s))$ to $(x(t), -\phi(t))$.

Tuning Parameters

Tuning of HMC can occur in several places such as

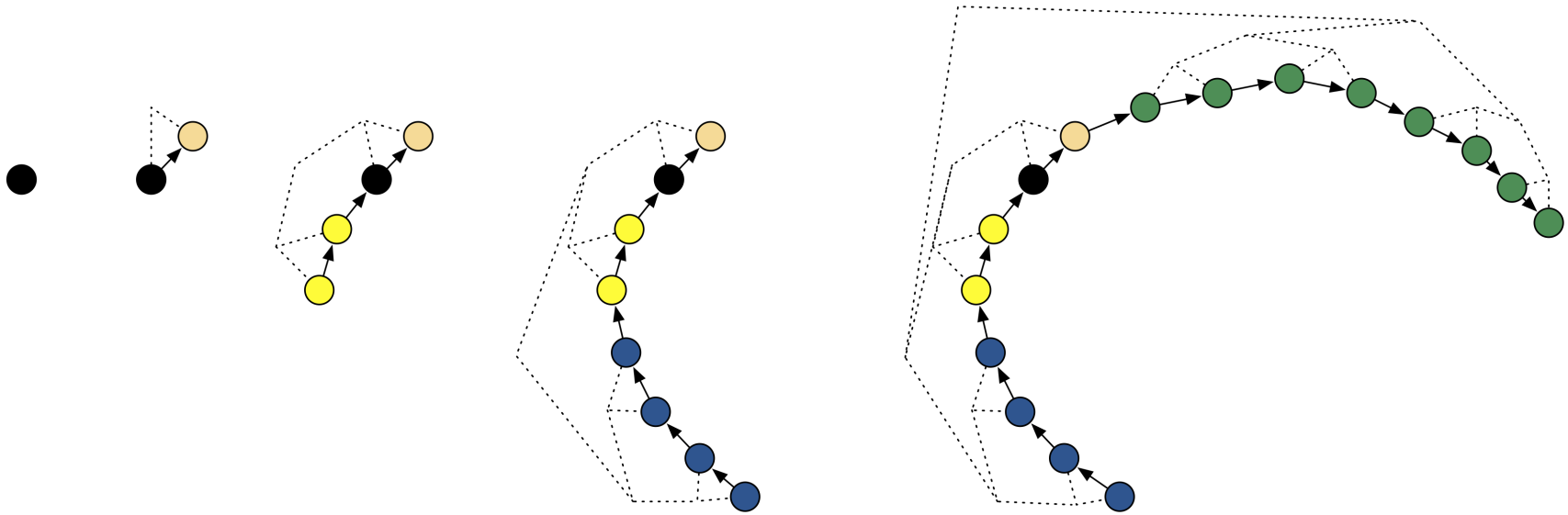
1. the distribution for the momentum (often independent normal),
2. the scaling factor ϵ,
3. the number of leapfrog steps L per iteration.

Some theory suggest that we can tune HMC such that the acceptance probability is around 65%.

No-U-Turn Sampler

The no-U-turn sampler (NUTS) allows us to automatically tune the number of steps L: we increases L until the simulated dynamics is long enough such that the proposed position θ^* starts to move back towards the initial position θ if we run more steps.

- This is measured by the angle between $\theta^* - \theta$ and current momentum ϕ^*.



Basic Algorithm of NUTS

A basic NUTS works as follows. Given the initial status,

1. Sample $u \mid \theta, \phi \sim \text{Uniform}\left[0, \exp\left\{-H\left(\theta, \phi\right)\right\}\right]$.
2. Apply the leapfrog method (with some modification) until a U-turn occurs.
3. Sample from the points in

$$\left\{(\theta, \phi):\ \exp\left\{-H\left(\theta, \phi\right)\right\} \geq u\right\}$$

that the leapfrog step has visited and the detailed balance condition is fulfilled.

Adaptively Tune ϵ

A too small ϵ will waste computation by taking needlessly tiny steps, and a too large will cause high rejection rates.

- In HMC, we tune ϵ in the warm-up stage of MCMC such that the average acceptance probability δ is the user specified value.
- In NUTS, there is no Metropolis accept/reject step. But we can still compute the ratio as if we were using the accept/reject step and set ϵ such that the pseudo acceptance probability is the user specified value.

In stan, the default is $\delta = 0.8$.