Intro to Markov Chain Monte Carlo

Rebecca C. Steorts
Bayesian Methods and Modern Statistics: STA 360/601

Module 6

Metropolis Algorithm

Setup: Assume pmf π on $\mathcal{X}$ (countable).

Have $f : \mathcal{X} \to \mathbb{R}$.

Goal:

a) sample/approximate from π
b) approximate $E_\pi[f(x)], X \sim \pi$.

The assumption is that π and or $f(X)$ are complicated!

Why things work!

Big idea and why it works: we apply the ergodic theorem.

"If we take samples $X = (X_0, X_1, \ldots,)$ then by the ergodic theorem, they will eventually reach π, which is known as the stationary distribution (the true pmf)."

Metropolis Algorithm

The approach is to apply the ergodic theorem.

1. If we run the Markov chain long enough, then the last state is approximately from π.
2. Under some regularly conditions,

$$\frac{1}{n}\sum_{i=1}^{n} f(X_i) \xrightarrow{a.s} E_{\pi}[f(x)].$$

Terminology

1. Proposal matrix = stochastic matrix.

   Let
   $$Q = (Q_{ab} : a, b \in \mathcal{X}).$$

   Note: I will use $Q_{ab} = Q(a, b)$ at times.
2. Note:
   $$\pi(x) = \tilde{\pi}(x)/z, z > 0.$$

   What is known and unknown above?

Metropolis Algorithm

- Choose a symmetric proposal matrix Q. So, $Q_{ab} = Q_{ba}$.
- Initialize $x_o \in X$.
- for $i \in 1, 2, \ldots, n-1$:
  - Sample proposal x from $Q(x_i, x) = p(x \mid x_i)$.
  - Sample u from Uniform$(0, 1)$.
  - If
    $$r < \frac{\tilde{\pi}(x)}{\tilde{\pi}(x_i)},$$
    accept and $x_{i+1} = x$.
  - Otherwise, reject and $x_{i+1} = x_i$.

You do not need to know the general proof of this.

Metropolis within a Bayesian setting

Goal: We want to sample from

$$p(\theta \mid y) = \frac{f(y \mid \theta)\pi(\theta)}{m(y)}.$$

Typically, we don't know $m(y)$.

The notation is a bit more complicated, but the set up is the same.

We'll approach it a bit differently, but the idea is exactly the same.

Building a Metropolis sampler

We know $\pi(\theta)$ and $f(y \mid \theta)$, so we can can draw samples from these.

Our notation here will be that we assume parameter values $\theta_1, \theta_2, \ldots, \theta_s$ which are drawn from $\pi(\theta)$.

We assume a new parameter value comes in that is θ^*.

Similar to before we assume a symmetric proposal distribution, which we call $J(\theta^* \mid \theta^{(s)})$.

- What does symmetry mean here? $J(\theta_a \mid \theta_b) = J(\theta_b \mid \theta_a)$.
- That is, the probability of proposing $\theta^* = \theta_a$ given that $\theta^{(s)} = \theta_b$ is equal to the probability of proposing $\theta^* = \theta_b$ given that $\theta^{(s)} = \theta_a$.
- Symmetric proposals include:

$$J(\theta^* \mid \theta^{(s)}) = \text{Uniform}(\theta^{(s)} - \delta, \theta^{(s)} + \delta)$$

  and

$$J(\theta^* \mid \theta^{(s)}) = \text{Normal}(\theta^{(s)}, \delta^2).$$

The Metropolis algorithm proceeds as follows:

1. Sample $\theta^* \sim J(\theta \mid \theta^{(s)})$.
2. Compute the acceptance ratio (r):

$$r = \frac{p(\theta^*|y)}{p(\theta^{(s)}|y)} = \frac{p(y \mid \theta^*)p(\theta^*)}{p(y \mid \theta^{(s)})p(\theta^{(s)})}.$$

3. Let

$$\theta^{(s+1)} = \begin{cases} \theta^* & \text{with prob min(r,1)} \\ \theta^{(s)} & \text{otherwise.} \end{cases}$$

Remark: Step 3 can be accomplished by sampling $u \sim \text{Uniform}(0,1)$ and setting $\theta^{(s+1)} = \theta^*$ if $u < r$ and setting $\theta^{(s+1)} = \theta^{(s)}$ otherwise.

Exercise: Convince yourselves that step 3 is the same as the remark!

A Toy Example of Metropolis

Let's test out the Metropolis algorithm for the conjugate Normal-Normal model with a known variance situation.

$$\begin{aligned} X_1, \ldots, X_n \mid \theta &\overset{iid}{\sim} \text{Normal}(\theta, \sigma^2) \\ \theta &\sim \text{Normal}(\mu, \tau^2). \end{aligned}$$

Recall that the posterior of θ is $\text{Normal}(\mu_n, \tau_n^2)$, where

$$\mu_n = \bar{x}\frac{n/\sigma^2}{n/\sigma^2 + 1/\tau^2} + \mu\frac{1/\tau^2}{n/\sigma^2 + 1/\tau^2}$$

and

$$\tau_n^2 = \frac{1}{n/\sigma^2 + 1/\tau^2}.$$

A Toy Example of Metropolis

In this example: $\sigma^2 = 1$, $\tau^2 = 10$, $\mu = 5$, $n = 5$, and

$$y = (9.37, 10.18, 9.16, 11.60, 10.33).$$

For these data, $\mu_n = 10.03$ and $\tau_n^2 = 0.20$.

Note: this is a toy example for illustration.

We need to compute the acceptance ratio r. Then

$$\begin{align}
r &= \frac{p(\theta^*|x)}{p(\theta^{(s)}|x)} \tag{1} \\
&= \frac{p(x|\theta^*)p(\theta^*)}{p(x|\theta^{(s)})p(\theta^{(s)})} \tag{2} \\
&= \left(\frac{\prod_i \mathsf{dnorm}(x_i, \theta^*, \sigma)}{\prod_i \mathsf{dnorm}(x_i, \theta^{(s)}, \sigma)}\right)\left(\frac{\prod_i \mathsf{dnorm}(\theta^*, \mu, \sigma)}{\prod_i \mathsf{dnorm}(\theta^{(s)}, \mu, \sigma)}\right) \tag{3}
\end{align}$$

In many cases, computing the ratio r directly can be numerically unstable, however, this can be modified by taking $\log r$.
This results in

$$\begin{aligned}\log r = &\sum_i \left[\log \mathsf{dnorm}(x_i, \theta^*, \sigma) - \log \mathsf{dnorm}(x_i, \theta^{(s)}, \sigma)\right] \\ &+ \sum_i \left[\log \mathsf{dnorm}(\theta^*, \mu, \sigma) - \log \mathsf{dnorm}(\theta^{(s)}, \mu, \sigma)\right].\end{aligned}$$

Then a proposal is accepted if $\log u < \log r$, where u is sample from the Uniform(0,1).

We generate 10,000 iterations of the Metropolis algorithm starting at $\theta^{(0)} = 0$ and using a normal proposal distribution, where

$$\theta^{(s+1)} \sim \text{Normal}(\theta^{(s)}, 2).$$

Figure 1 shows a trace plot for this run as well as a histogram for the Metropolis algorithm compared with a draw from the true normal density.

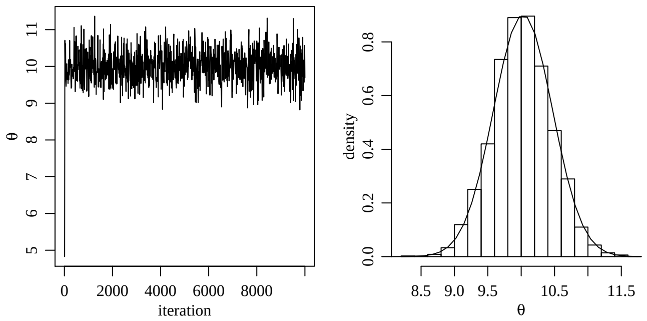

Figure 1: Left: trace plot of the Metropolis sampler. Right: Histogram versus true normal density for 10,000 iterations.

```
# setting values
set.seed(1)
s2<-1
t2<-10
mu<-5;
n<-5

# rounding the rnorm to 2 decimal places
y<-round(rnorm(n,10,1),2)
# mean of the normal posterior
mu.n<-( mean(y)*n/s2 + mu/t2 )/( n/s2+1/t2)
# variance of the normal posterior
t2.n<-1/(n/s2+1/t2)
# defining the data
y<-c(9.37, 10.18, 9.16, 11.60, 10.33)
```

```
####metropolis part####
##S = total num of simulations
theta<-0 ; delta<-2 ; S<-10000 ; THETA<-NULL ; set.seed(1)

for(s in 1:S){
## simulating our proposal
  #the new value of theta
  theta.star<-rnorm(1,theta,sqrt(delta))

##taking the log of the ratio r
  log.r<-( sum(dnorm(y,theta.star,sqrt(s2),log=TRUE))
              + dnorm(theta.star,mu,sqrt(t2),log=TRUE) )
        - ( sum(dnorm(y,theta,sqrt(s2),log=TRUE))
             +  dnorm(theta,mu,sqrt(t2),log=TRUE) )

  if(log(runif(1))<log.r) { theta<-theta.star }
##updating THETA
  THETA<-c(THETA,theta)
}
```

```
##two plots: trace of theta and
comparing the empirical distribution
##of simulated values to the true posterior

par(mar=c(3,3,1,1),mgp=c(1.75,.75,0))
par(mfrow=c(1,2))

# creating a sequence
skeep<-seq(10,S,by=10)
# making a trace place
plot(skeep,THETA[skeep],type="l",
xlab="iteration",ylab=expression(theta))

# making a histogram
hist(THETA[-(1:50)],prob=TRUE,main="",
xlab=expression(theta),ylab="density")
th<-seq(min(THETA),max(THETA),length=100)
lines(th,dnorm(th,mu.n,sqrt(t2.n)) )
```

Intro to Markov chain Monte Carlo (MCMC)

Goal: sample from $f(x)$, or approximate $E_f[h(X)]$.

Recall that $f(x)$ is very complicated and hard to sample from.

How to deal with this?

1. What's a simple way?
2. What are two other ways?
3. What happens in high dimensions?

High dimensional spaces

- In low dimensions, IS and RS works pretty well.
- But in high dimensions, a proposal $g(x)$ that worked in 2-D, often doesn't mean that it will work in any dimension.
- Why? It's hard to capture high dimensional spaces!

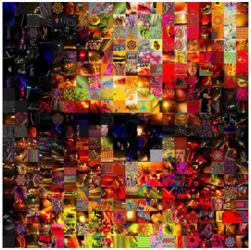

Figure 2: A high dimensional space (many images).

We turn to Markov chain Monte Carlo (MCMC).

Intution

Imagine that we have a complicated function f below and it's high probability regions are represented in green.

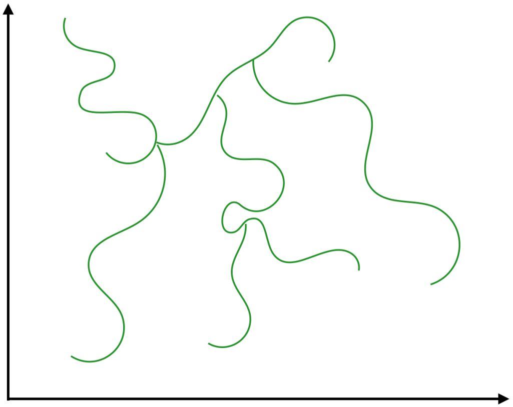

Figure 3: Example of a Markov chain

Intution

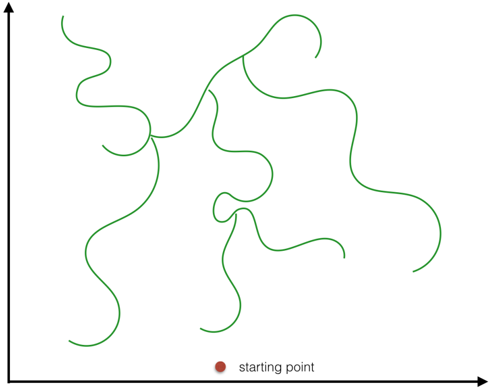

Figure 4: Example of a Markov chain and red starting point

Intution

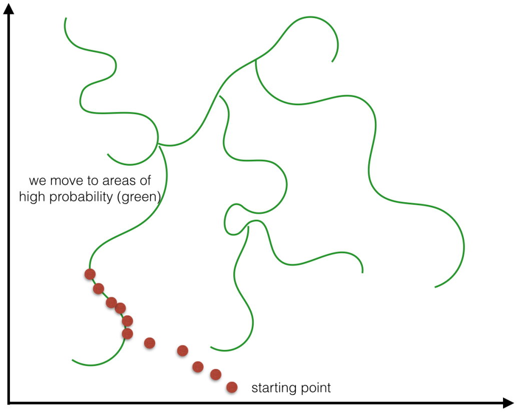

Figure 5: Example of a Markov chain and moving from the starting point to a high probability region.

What is Markov Chain Monte Carlo

- Markov Chain – where we go next only depends on our last state (the Markov property).
- Monte Carlo – just simulating data.

Why MCMC?

(a) the region of high probability tends to be "connected"

- That is, we can get from one point to another without going through a low-probability region, and

(b) we tend to be interested in the expectations of functions that are relatively smooth and have lots of "symmetries"

- That is, one only needs to evaluate them at a small number of representative points in order to get the general picture.

Advantages/Disadvantages of MCMC:

Advantages:

- applicable even when we can't directly draw samples
- works for complicated distributions in high-dimensional spaces, even when we don't know where the regions of high probability are
- relatively easy to implement
- fairly reliable

Disadvantages:

- slower than simple Monte Carlo or importance sampling (i.e., requires more samples for the same level of accuracy)
- can be very difficult to assess accuracy and evaluate convergence, even empirically

Hard Discs in a Box Example

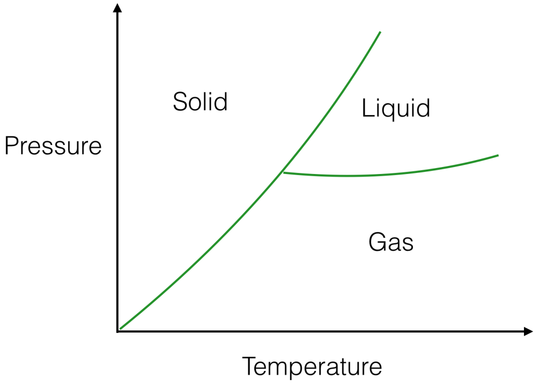

Figure 6: Example of a phase diagram in chemistry.

Many materials have phase diagrams that look like the picture above.

Hard Discs in a Box Example

To understand this phenoma, a theoretical model was proposed: Metropolis, Rosenbluth, Rosenbluth, and Teller, 1953

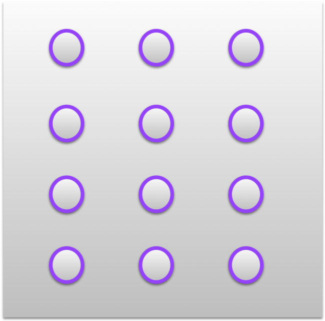

Figure 7: Example of N molecules (hard discs) bouncing around in a box.

Called hard discs because the molecules cannot overlap.

Hard Discs in a Box Example

Have an (x, y) coordinate for each molecule.

The total dimension of the space is $\mathbb{R}^{2N}$.

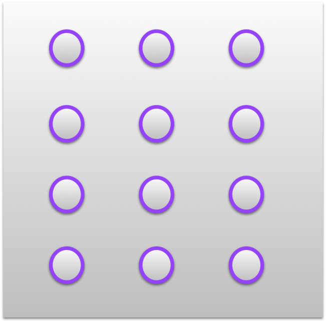

Figure 8: Example of N molecules (hard discs) bouncing around in a box.

Hard Discs in a Box Example

$X \sim f(x)$ (Boltzman distribution).

Goal: compute $E_f[h(x)]$.

Since X is high dimensional, they proposed “clever moves” of the molecules.

High Level Overview of Metropolis Algorithm

Metropolis algorithm: For iterations $i = 1, \ldots, n$, do:

1. Consider a molecule and a box around the molecule.
2. Uniformly draw a point in the box.
3. According to a "rule", you accept or reject the point.
4. If it's accepted, you move the molecule.

[For clarification, you could use this as pseudocode on the exam instead of writing R code.]

Example of one iteration of algorithm

Consider a molecule and a box around the molecule.

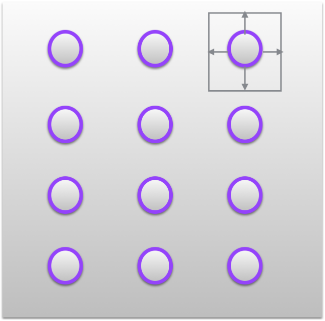

Figure 9: This illustrates step 1 of the algorithm.

Example of one iteration of algorithm

Uniformly draw a point in the box.

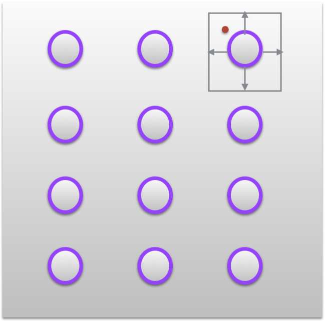

Figure 10: This illustrates step 2 of the algorithm.

Example of one iteration of algorithm

According to a "rule", you accept or reject the point.

Here, it was accepted, so we move the point.

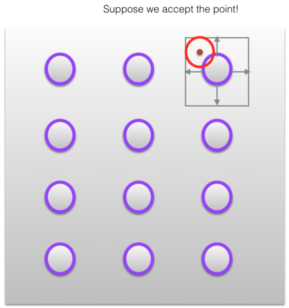

Figure 11: This illustrates step 3 and 4 of the algorithm.

Example of one iteration of algorithm

Here, we show one entire iteration of the algorithm.

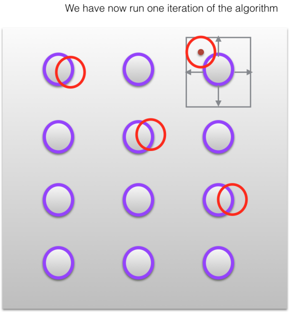

Figure 12: This illustrates one iteration of the algorithm.

After running many iterations n (not N), we have an approximation for $E_f(h(X))$, which is $\frac{1}{n}\sum_i h(X_i)$.

We will talk about the details later of why this is a "good approximation."

Some food for thought

We just covered the Metropolis algorithm (1953 paper).

- We did not cover the exact procedure for accepting or rejecting (to come).
- Are the X_i's independent?
- Our approximation holds by The Ergodic Theorem for those that want to learn more about it.
- The ergodic theorem says: "if we start at a point x_o and we keeping moving around in our high dimensional space, then we are guaranteed to eventually reach all points in that space with probability 1."