

Markov Chain Monte Carlo

David Madigan
Columbia University

Markov Chain Monte Carlo

- Generate a sequence of random variables $\{X_0, X_1, \dots\}$ where at each time $t \geq 0$ the next state X_{t+1} is sampled from $P(X_{t+1} | X_t)$

$$X_0 \rightarrow X_1 \rightarrow X_2 \dots$$

$P(\cdot | \cdot)$ is called the transition kernel of the chain. Subject to regularity condition, the chain “forgets” its starting position. That is, $P^{(t)}(\cdot | X_0)$ will converge to a stationary distribution that does not depend on t or X_0 . Denote the stationary distribution by $\phi(\cdot)$.

After some “burn-in”, points $\{X_t, t = m+1, m+2, \dots, n\}$ will be dependent samples approximately from $\phi(\cdot)$.

We can use output from the Markov Chain to estimate $E[h(x)]$ where $X \sim \phi(\cdot)$
:

$$\bar{h} = \frac{1}{n - m} \sum_{t=m+1}^n h(X_t)$$

(convergence is ensured by the “ergodic theorem”)

Metropolis-Hastings Algorithm

At each time t , X_{t+1} is chosen as follows:

- (1) Sample a candidate point Y from a proposal distribution $q(\cdot | X_t)$
- (2) Accept Y with probability:

$$\alpha(X_t, Y) = \min \left(1, \frac{\phi(Y)q(X_t | Y)}{\phi(X_t)q(Y | X_t)} \right)$$

- (3) If accepted, $X_{t+1} = Y$, else $X_{t+1} = X_t$.

Why does this work?

The transition kernel for the M-H sampler is:

$$P(X_{t+1} | X_t) = q(X_{t+1} | X_t) \alpha(X_t, X_{t+1}) + I(X_{t+1} = X_t) \left[1 - \int q(Y | X_t) \alpha(X_t, Y) dY \right]$$

Note:

$$\frac{\alpha(X_t, X_{t+1})}{\alpha(X_{t+1}, X_t)} = \frac{\min\left(1, \frac{\phi(X_{t+1})q(X_t | X_{t+1})}{\phi(X_t)q(X_{t+1} | X_t)}\right)}{\min\left(1, \frac{\phi(X_t)q(X_{t+1} | X_t)}{\phi(X_{t+1})q(X_t | X_{t+1})}\right)} = \frac{\phi(X_{t+1})q(X_t | X_{t+1})}{\phi(X_t)q(X_{t+1} | X_t)}$$
$$\Rightarrow \alpha(X_t, X_{t+1})\phi(X_t)q(X_{t+1} | X_t) = \alpha(X_{t+1}, X_t)\phi(X_{t+1})q(X_t | X_{t+1})$$
$$\Rightarrow \phi(X_t)p(X_{t+1} | X_t) = \phi(X_{t+1})p(X_t | X_{t+1}) \quad \text{"detailed balance"}$$
$$\Rightarrow \int \phi(X_t)p(X_{t+1} | X_t)dX_t = \phi(X_{t+1})$$

So, if $X_t \sim \phi(\cdot)$, then $X_{t+1} \sim \phi(\cdot)$.

Note: q must result in an irreducible and aperiodic process

Canonical forms for $q(\cdot|\cdot)$

Metropolis Algorithm

If $q(Y|X) = q(X|Y) \quad \forall X, Y$ then M-H becomes:

$$\alpha(X|Y) = \min\left(1, \frac{\phi(Y)}{\phi(X)}\right)$$

Example:

$q(\cdot|X) = \text{mvn}(X, \Sigma)$ where Σ is constant. Note that Σ needs careful choice...

Independence Sampler

If $q(Y | X) = q(Y) \dots$

$$\alpha(X | Y) = \min \left(1, \frac{\phi(Y) / q(Y)}{\phi(X) / q(X)} \right)$$

(works best if $q(\cdot)$ is a good approximation to $\phi(\cdot)$)

Single Component M-H

We can divide X into components $\{X_{.1}, X_{.2}, \dots, X_{.k}\}$ and update these components one by one.

Gibbs Sampling (special case of single component M-H)

$q_i(Y_i | X_{-i}) = \phi(Y_i | X_{-i})$ – conditional distribution of the i -th component given all the others.

```

numlter <- 10
e <- 10
x <- rep(0,numlter)
candidate <- rep(0,numlter)
mu <- 3
sigma <- 5
accept <- 0

for (i in 2:numlter) {
  candidate[i] <- runif(1,x[i-1]-e,x[i-1]+e);
  if (runif(1) < (dnorm(candidate[i],mu,sigma)/dnorm(x[i-1],mu,sigma)) ) {
    x[i] <- candidate[i]
    accept <- accept+1
  }
  else {
    x[i] <- x[i-1]
  }
}

par(mfrow=c(2,1))
plot(candidate,col=3,type="b",ylim=c(min(x)-1,max(x)+1))
par(new=TRUE)
plot(x,type="b",ylim=c(min(x)-1,max(x)+1))
hist(x,nclass=20,freq=FALSE,xlim=c(min(x)-1,max(x)+1),ylim=c(0,1.1*dnorm(mu,mu,sigma)))
par(new=TRUE)
temp<- seq(min(x)-1,max(x)+1,length=1000)
plot(temp,dnorm(temp,mu,sigma),xlim=c(min(x)-1,max(x)+1),ylim=c(0,1.1*dnorm(mu,mu,sigma)),type="l")

cat("Acceptance Rate: ",accept/numlter,"\n");

```

```
bioassay <- data.frame(doses=c(-.863,-.296,-.053,.727),  
                      rats=c(5,5,5,5),deaths=c(0.5,1,3,4.5),freq=c(0.5,1,3,4.5)/5)
```

```
log.post<-function(alpha,beta,data=bioassay){  
  ldens <- 0  
  for (i in 1:length(data$doses)){  
    theta <- 1/(1+exp(-alpha-beta*data$doses[i]))  
    ldens <- ldens + data$deaths[i]*log(theta)+(data$rats[i]-data$deaths[i])*log(1-theta)  
  }  
  ldens  
}
```

```
numlter <- 1000  
alpha <- rep(0,numlter);  
beta <- rep(0,numlter);
```

```
alpha[1] <- -5  
beta[1] <- -5  
e <- 1
```

```
for (i in 2:numlter) {  
  alpha[i] <- runif(1,alpha[i-1]-e,alpha[i-1]+e);  
  beta[i] <- runif(1,beta[i-1]-e,beta[i-1]+e);  
  if (runif(1) > exp(log.post(alpha[i],beta[i])-log.post(alpha[i-1],beta[i-1]))) {  
    alpha[i] <- alpha[i-1];  
    beta[i] <- beta[i-1];  
  }  
}
```

```
plot(alpha,beta,type="l")
```


