Metropolis Hastings Algorithm

Michael Rose

November 3, 2018

Question

Suppose we have data $X_1, ..., X_n$ which we believe come from a normal distribution with mean θ and variance 1. Suppose we are uncertain about θ, and for us θ has the following Cauchy Distribution:

$$f(\theta) = \frac{1}{\pi(1+\theta^2)}, -\infty < \theta < \infty$$

This is a special form of the Cauchy called the standard Cauchy Distribution with parameters $x_0 = 0, \gamma = 1$.

We will write a Metropolis Hastings Algorithm whose limiting distribution is our posterior distribution:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\theta)^2}{2}} \frac{1}{\pi(1+\theta^2)}$$
$$X_i \sim \text{Normal}(\theta, 1)$$
$$\theta \sim \text{Cauchy}(0, 1)$$

First lets simulate some data

```
set.seed(100)
# choose a cauchy sample for mean
theta_x <- rcauchy(1,location = 0, scale = 1)
theta_x
```

```
## [1] 1.449469
```

```
# generate samples
data_y <- rnorm(n = 1000, mean = theta_x, sd = 1)
```

Now we can create our metropolis algorithm:

The Metropolis Algorithm calls for θ_{prop} to be sampled from a symmetric proposal distribution centered at the current parameter value, θ_{curr}. For this task we will use $\theta_{prop} \sim \text{Normal}(\theta_{curr}, \sigma^2)$.

The proposal distribution is seperate and distinct from either the prior or posterior distribution for the parameter. The proposal distribution's sole purpose is to give candidate parameter values to *try* and potentially accept as a valid sample from the posterior distribution of θ.

- samples is the number of samples we want to draw from the posterior distribution and determines the length of the resulting MCMC chain
- theta_start gives us a θ to start the algorithm
- sd is the standard deviation of the proposal distribution

Within the function, we construct a for loop that repeatedly draws θ_{prop} from a standard normal proposal distribution (using rnorm). It then computes the ratio of Bayes' numerators and carries out the accept / reject logic. We store the results in a vector called posterior_thetas which are initialized to NA.

```
metropolis_algo <- function(samples, theta_start, sd){
  # declarations
  theta_curr <- theta_start
  # vector of NAs to store sampled parameters
  posterior_thetas <- rep(NA, times = samples)

  for (i in 1:samples){
```

```r
    # proposal distribution
    theta_prop <- rnorm(n = 1, mean = theta_curr, sd = sd)

    # if proposed parameter is outside range, set to current value. Else keep proposed value
    theta_prop <- ifelse((theta_prop < 0 | theta_prop > 1), theta_curr, theta_prop)

    # bayes numerators
    posterior_prop <- dcauchy(theta_prop, location = 0, scale = 1) *
      dnorm(data_y, mean = theta_prop, sd = 1)
    posterior_curr <- dcauchy(theta_curr, location = 0, scale = 1) *
      dnorm(data_y, mean = theta_curr, sd = 1)

    # calculate probability of accepting
    p_accept_theta_prop <- min(posterior_prop / posterior_curr, 1.0)

    rand_unif <- runif(n=1)

    # probabilistically accept proposed theta
    theta_select <- ifelse(p_accept_theta_prop > rand_unif, theta_prop, theta_curr)

    posterior_thetas[i] <- theta_select

    # reset theta_curr for the next iteration of the loop
    theta_curr <- theta_select
  }
  return(posterior_thetas)
}
```

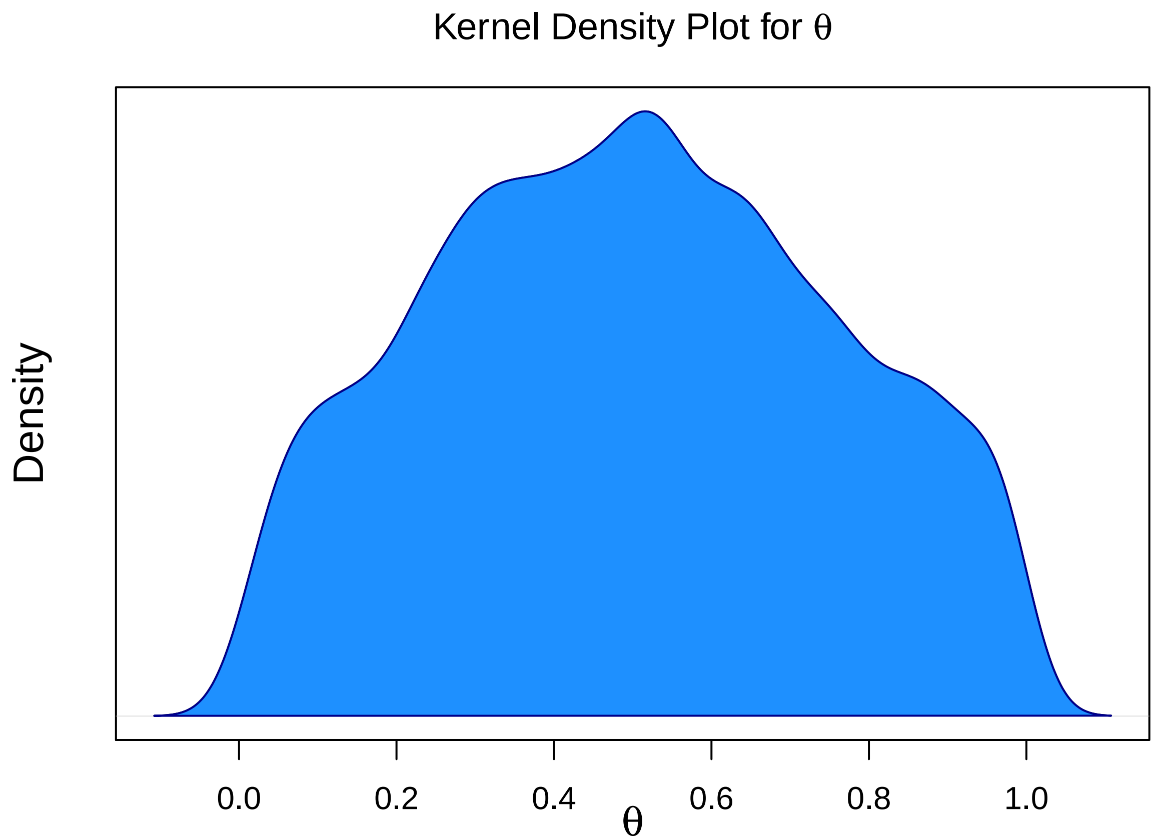
Now we can try 10,000 samples with a starting value of 0.9 and a sd for our normal proposal distribution of 0.05

```r
set.seed(999)
posterior_thetas <- metropolis_algo(samples = 10000, theta_start = 0.9, sd = 0.05)
```

Lets take a look at the kernel density estimate of the posterior.

```r
opar <- par()
par(mar = c(2.5, 3.5, 3, 2.1), mgp = c(1.7, 1, 0))

d <- density(posterior_thetas)
plot(d, main = expression(paste("Kernel Density Plot for ", theta)),
     xlab = expression(theta), ylab = "Density", yaxt = "n", cex.lab = 1.3)
polygon(d, col = 'dodgerblue1', border = 'dark blue')
```


Kernel Density Plot for θ
Density
0.0
0.2
0.4
0.6
0.8
1.0
θ