Chapter 9.3b The Metropolis Algorithm

Jim Albert and Monika Hu

Chapter 9 Simulation by Markov Chain Monte Carlo

The general algorithm

- Generalize the random walk sampler in the previous section.
- The Markov chain Monte Carlo sampling strategy sets up an irreducible, aperiodic Markov chain for which the stationary distribution equals the posterior distribution of interest.
- This Metropolis algorithm, is applicable to a wide range of Bayesian inference problems.
- This algorithm is a special case of the Metropolis-Hastings algorithm, where the proposal distribution is symmetric.

Setup

- Suppose the posterior density is written as

$$\pi_n(\theta) \propto \pi(\theta)L(\theta),$$

  where $\pi(\theta)$ is the prior and $L(\theta)$ is the likelihood function.
- it is not necessary to compute the normalizing constant – only the product of likelihood and prior is needed.

Basic Steps of Metropolis Algorithm

Start at any θ value where the posterior density is positive.

1. (PROPOSE) Given the current value $\theta^{(j)}$ propose a new value θ^{P} selected at random in the interval $(\theta^{(j)} - C, \theta^{(j)} + C)$ where C is a preselected constant.

2. (ACCEPTANCE PROBABILITY) Compute the ratio R of the posterior density at the proposed value and the current value:

$$R = \frac{\pi_n(\theta^P)}{\pi_n(\theta^{(j)})}. \tag{1}$$

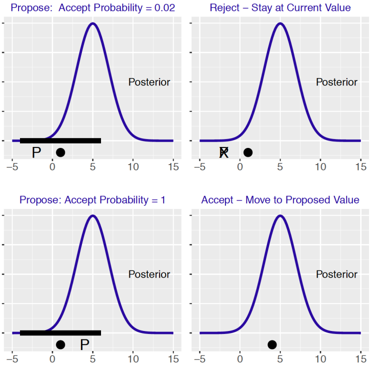
   The acceptance probability is the minimum of R and 1:

$$PROB = \min\{R, 1\}. \tag{2}$$

3. (MOVE OR STAY?) With probability $PROB$, move to the proposed value θ^{P}; otherwise stay at the current value $\theta^{(j)}$.

Illustration - bell-shaped curve is the posterior density

In top panel, the acceptance probability is 0.02 and one decides not to accept this proposal. In bottom panel, one proposes a value corresponding to a higher posterior density value and it is accepted.



A general function for the Metropolis algorithm

- One writes a short function in R to implement this sampling for an arbitrary probability distribution.
- The function `metropolis()` has five inputs:
- `logpost` is a function defining the logarithm of the density
- `current` is the starting value
- `C` defines the neighborhood where one looks for a proposal value
- `iter` is the number of iterations of the algorithm
- `...` denotes any data or parameters needed in the function `logpost()`.

The metropolis function

```
metropolis <- function(logpost, current, C, iter, ...){
  S <- rep(0, iter)
  n_accept <- 0
  for(j in 1:iter){
  candidate <- runif(1, min=current - C,
                     max=current + C)
  prob <- exp(logpost(candidate, ...) -
             logpost(current, ...))
  accept <- ifelse(runif(1) < prob, "yes", "no")
  current <- ifelse(accept == "yes",
                    candidate, current)
  S[j] <- current
  n_accept <- n_accept + (accept == "yes")
  }
  list(S=S, accept_rate=n_accept / iter)
}
```