Markov Chain Monte Carlo Methods

Micole De Vera

CEMFI

1 March 2021
Bayesian Inference

Elements:

- Data: \( \{ y_i \}_{i=1}^{n} \)
- Model/likelihood: \( f(y \mid \theta) \)
- Prior on parameters: \( p(\theta), \theta \in \Theta \)

Goal: Integrals involving the posterior \( p(\theta \mid y) = \frac{f(y \mid \theta)p(\theta)}{\int_{\Theta} f(y \mid \theta^*)p(\theta^*)d\theta^*} \)

\[ \mathbb{E}[h(\theta) \mid y] = \int_{\Theta} h(\theta)p(\theta \mid y) \, d\theta \]

This includes posterior means, posterior variances, credible intervals, and the posterior cdf.

Problems:

- Obtaining the posterior density is difficult/impossible
- Integrals are too complicated (intractable)
Possible Solution: Simulation (I)

Suppose we can produce iid draws from $p(\theta|y)$: \( \{\theta^{(m)}\}_{m=1}^{M} \)

An estimator of $\mathbb{E}[h(\theta) \mid y]$ could be

$$\hat{h}_M = \frac{1}{M} \sum_{m=1}^{M} h\left(\theta^{(m)}\right)$$

By a LLN,

$$\hat{h}_M \xrightarrow{p} \mathbb{E}[h(\theta) \mid y]$$
Possible Solution: Simulation (II)

Maybe we cannot sample \textit{iid} from the posterior but we can obtain a stationary, ergodic sequence \( \{ \theta^{(m)} \}_{m=1}^{M} \) with marginal density \( p(\theta \mid y) \).

The estimator \( \hat{h}_{M} \) is still valid.

Under stationarity and ergodicity, we have a LLN that tells us

\[
\hat{h}_{M} \xrightarrow{p} \mathbb{E}[h(\theta) \mid y]
\]
Markov Chains

**Definition** *(Markov Chain)* A continuous-state Markov Chain is a sequence \(\theta^{(1)}, \theta^{(2)}, \ldots\) that satisfies the Markov property:

\[
Pr \left( \theta^{(j+1)} | \theta^{(j)}, \ldots, \theta^{(1)} \right) = Pr \left( \theta^{(j+1)} | \theta^{(j)} \right)
\]

where \(Pr(\theta' | \theta)\) is called the transition kernel and is denoted by \(\kappa(\theta' | \theta)\). It gives us the marginal density of the next-period draws:

\[
p_m(\theta') = \int_{\Theta} \kappa(\theta' | \theta)p_{m-1}(\theta) \, d\theta
\]

The stationary distribution of the given transition kernel (if it exists), is such that

\[
p_s(\theta') = \int_{\Theta} \kappa(\theta' | \theta)p_s(\theta) \, d\theta
\]
Markov Chain Monte Carlo (MCMC)

- MCMC is a collection of methods to construct transition kernels \( \kappa(\theta' | \theta) \) with stationary distribution \( p(\theta | y) \).
- Given an initial value \( \theta^{(0)} \) we can generate a sequence \( \theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(M)} \) using the transition kernel \( \kappa(\theta' | \theta) \).
  With \( M \to \infty \),
  - Marginal distribution of \( \theta^{(M)} \) converges to \( p(\theta | y) \).
  - The dependent sample \( \{ \theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(M)} \} \) will have an empirical distribution that approaches \( p(\theta | y) \).
  - Usually, the way we will construct the sequence is such that we can use a LLN

\[
\hat{h}_M = \frac{1}{M} \sum_{m=1}^{M} h(\theta^{(m)}) \xrightarrow{p} \mathbb{E}[h(\theta) | y]
\]

- Two popular methods:
  1. Metropolis-Hastings Algorithm
  2. Gibbs Sampler
Outline

Introduction

Metropolis-Hastings Algorithm
- Presentation of Algorithm
- Some Details on Implementation
- Example

Gibbs Sampler
- Presentation of Algorithm
- Example
- Gibbs Sampler as a Special Case of MH
- Combining Ideas
Metropolis-Hastings (MH) Algorithm

Inputs:
- Way to compute the un-normalized posterior
  \[ p(\theta \mid y) \propto f(y \mid \theta)p(\theta) \]
- Proposal density we know how to draw from: \( q(\theta' \mid \theta) \)

Algorithm: Start with initial draw \( \theta^{(0)} \). For \( m = 1, \ldots, M \)
1. Draw \( \theta^* \) from \( q(\theta \mid \theta^{(m-1)}) \) and \( u \) from \( \mathcal{U}(0, 1) \) independently
2. Compute acceptance probability
   \[ \rho(\theta^* \mid \theta^{(m-1)}) = \min \left\{ 1, \frac{f(y \mid \theta^*)p(\theta^*)q(\theta^{(m-1)} \mid \theta^*)}{f(y \mid \theta^{(m-1)})p(\theta^{(m-1)})q(\theta^* \mid \theta^{(m-1)})} \right\} \]
3. New draw
   \[ \theta^{(m)} = \begin{cases} 
   \theta^* & \text{if } u \leq \rho(\theta^* \mid \theta^{(m-1)}) \\
   \theta^{(m-1)} & \text{otherwise} 
   \end{cases} \]
Why does it work? (Intuition)

Suppose that we are using a symmetric proposal distribution; that is, \( q(\theta^* | \theta) = q(\theta | \theta^*) \). The sequence \( \theta^{(1)}, ..., \theta^{(M)} \) generated by \( \kappa(\theta' | \theta) \) should have empirical distribution close to \( p(\theta | y) \).

- Given \((\theta', \theta)\), one of the following is true:

\[
p(\theta' | y) \geq p(\theta | y) \text{ or } p(\theta' | y) < p(\theta | y)
\]

- If \( p(\theta' | y) \geq p(\theta | y) \)
  - For every \( \theta \) in the sequence, we should have at least as many \( \theta' \)
  - Accept all \( \theta \rightarrow \theta' \)
- If \( p(\theta' | y) < p(\theta | y) \)
  - For every \( \theta \), we should have on average \( \frac{p(\theta' | y)}{p(\theta | y)} \) draws of \( \theta' \)
  - Accept \( \theta \rightarrow \theta' \) with probability \( \frac{p(\theta' | y)}{p(\theta | y)} \)
- Given \( \theta \), accept proposal \( \theta' \) with probability

\[
\min \left\{ 1, \frac{p(\theta' | y)}{p(\theta | y)} \right\}
\]
Proposal Density

What makes a good proposal density?

- It is easy to sample from $q(\theta^*|\theta)$ for any $\theta$
- Easy to compute the acceptance ratio $\rho$
- Proposals are reasonable distances apart in $\Theta$
- Proposals are not rejected too frequently

Main classes for proposal densities:

- **Random Walk:** $\theta^* = \theta^{(m)} + \varepsilon$
  - If the distribution of $\varepsilon$ is symmetric about 0, then $q(\theta^* | \theta) = q(\theta | \theta^*)$
  - Typical choices: $\varepsilon \sim \mathcal{N}(0, \Omega)$ or $\varepsilon \sim \mathcal{U}(-\delta, \delta)$
- **Independent:** $q(\theta^* | \theta) = q(\theta^*)$
  - $\{\theta^{(m)}\}$ may display less serial dependence
  - Candidate: “easy-to-draw-from” approximation of the posterior
Other Implementation Details

**Burn-in**
- Discard first $n$ draws
- Reduces dependence on the (possibly “bad”) initial draw
- Idea: Your initial draws might be in a low probability region
  ⇒ oversampling of low probability region
  ⇒ allow time for algorithm to “get to” high probability region

**Thinning**
- Only retain every $d$th iteration of the chain
- Reduces dependence between draws
  → BUT! Average on thinned sequence has greater variance than average over entire sequence
- Possibly useful when computationally-constrained
  → If the chain has very high autocorrelations, you would want to run the chain for a long time but you might not be able to store the entire chain (or operations on long chains are costly)
Example: Normal Linear Regression with Known Variance

- Model:

\[ y_i | \beta, x_i \sim \mathcal{N}(\beta_0 + \beta_1 x_i, 1) \]

- Prior:

\[
\begin{pmatrix}
\beta_0 \\
\beta_1
\end{pmatrix}
\sim \mathcal{N}
\begin{pmatrix}
1 \\
1
\end{pmatrix}
, \begin{pmatrix}
10 & 0 \\
0 & 5
\end{pmatrix}
\]

- Proposal:

\[
\begin{pmatrix}
\beta_{0}^* \\
\beta_{1}^*
\end{pmatrix}
= \begin{pmatrix}
\beta_0 \\
\beta_1
\end{pmatrix}
+ \varepsilon, \ \varepsilon \sim \mathcal{N}
\begin{pmatrix}
0 \\
0
\end{pmatrix}
, \begin{pmatrix}
0.01 & 0 \\
0 & 0.01
\end{pmatrix}
\]

Example: Code (I)

```matlab
function val = llikelihood(y, x, params)
    b0 = params(1);
    b1 = params(2);

    % Get predictions
    pred = b0 + b1 * x;
    indiv_like = normpdf(y, pred, 1);
    indiv_ll = log(indiv_like);
    val = sum(indiv_ll);
end

function val = lprior(params)
    b0 = params(1);
    b1 = params(2);

    % Prior on b0;
    b0_prior = normpdf(b0, 1, 10);
    b1_prior = normpdf(b1, 1, 5);

    % Prior
    val = log(b0_prior) + log(b1_prior);
end

function val = unnorm_lpost(y, x, params)
    val = llikelihood(y, x, params) + lprior(params);
end
```
% MH Parameters
burn = 5000;
M = 5000;

chain = NaN(2, burn + M);
chain(:, 1) = [1; 1];
accept = NaN(1, burn + M);

for m = 2:(burn + M)
    % Proposal
    proposal = chain(:, m - 1) + mvnrnd([0; 0], [0.01, 0;
                                           0, 0.01]);

    % Acceptance probability
    rho = exp(unnorm_lpost(y, x, proposal) - unnorm_lpost(y, x, chain(:, m - 1)));
    rho = min(1, rho);

    % Update
    if rand(1) <= rho
        chain(:, m) = proposal;
        accept(m) = 1;
    else
        chain(:, m) = chain(:, m - 1);
        accept(m) = 0;
    end
end

% Acceptance ratio
mean(accept(:, burn+1:end))
Example: Posterior Draws

\[ \beta_0 \]

\[ \beta_1 \]
Outline

Introduction

Metropolis-Hastings Algorithm
  Presentation of Algorithm
  Some Details on Implementation
  Example

Gibbs Sampler
  Presentation of Algorithm
  Example
  Gibbs Sampler as a Special Case of MH
  Combining Ideas
Gibbs Sampler

Inputs:

- Partition of the parameter vector $\theta = (\theta_1, \theta_2)$
- Way to draw from the conditional posteriors $p(\theta_1 | \theta_2, y)$ and $p(\theta_2 | \theta_1, y)$

Algorithm: Start with initial draw $\theta_1^{(0)}$. For $m = 1, \ldots, M$

1. Draw $\theta_2^{(m)}$ from $p(\theta_2 | \theta_1^{(m-1)}, y)$
2. Draw $\theta_1^{(m)}$ from $p(\theta_1 | \theta_2^{(m)}, y)$

Generalizable to a partition $\theta = (\theta_1, \ldots, \theta_d)$
Example: Normal Regression with Independent N-IG Priors

▶ Model:

\[ y = X\beta + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2 I_n) \]

▶ Priors:

\[ \beta \sim N(\beta_0, \Sigma_0) \]
\[ \sigma^2 \sim \text{Inv-Gamma}(a_0, b_0) \]

We will use the parameterization consistent with Matlab: \( a_0 \) is the shape parameter while \( b_0 \) is the scale parameter.
Example: Conditional Posteriors

The conditional posteriors are (verifying this is good exercise)

\[ \beta \mid \sigma^2, y \sim \mathcal{N}(\beta_1, \Sigma_1) \]
\[ \sigma^2 \mid \beta, y \sim \text{Inv-Gamma}(a_1, b_1) \]

with

\[ \Sigma_1 = \left( \Sigma_0^{-1} + \frac{1}{\sigma^2} X'X \right)^{-1} \]
\[ \beta_1 = \Sigma_1 \left( \Sigma_0^{-1} \beta_0 + \frac{1}{\sigma^2} X'X \hat{\beta} \right) \]
\[ \hat{\beta} = (X'X)^{-1} X' y \]
\[ a_1 = \frac{N}{2} + a_0 \]
\[ b_1 = \left( \frac{1}{b_0} + \frac{1}{2} (y - X\beta)'(y - X\beta) \right)^{-1} \]
Example: Code (I)

```matlab
% Prior hyperparameters
beta0 = [1; 1];
Sigma0 = [2, 0; 0, 2];
a0 = 1;
b0 = 1;

% OLS coefficient
beta_ols = (X' * X) \ X' * y;
```
Example: Code (II)

```matlab
% Gibbs Sampler
burn = 5000;
M = 5000;
chain = NaN(3, burn + M);

chain(1:2, 1) = beta_ols;

for m = 2:(burn + M)
    % Draw sigma_sq conditional on beta
    a1 = (N / 2) + a0;
    b1 = (1 / b0) + 0.5 * (y - X * chain(1:2, m - 1))' * (y - X * chain(1:2, m - 1));
    b1 = 1 / b1;
    chain(3, m) = 1 / gamrnd(a1, b1);

    % Draw beta conditional on sigma_sq
    Sigma1 = pinv(pinv(Sigma0) + X' * X / chain(3, m));
    beta1 = Sigma1 * (pinv(Sigma0) * beta0 + X' * X * beta_ols / chain(3, m));
    chain(1:2, m) = mvnrnd(beta1, Sigma1);
end
```
Posterior Draws
Define a MH algorithm where for each iteration $m = 1, ..., M$, there are $d$ sub-steps. Entire algorithm has $Md$ steps. The $j$th sub-step corresponds to an update of the $j$th partition of the parameter vector.

The proposal density implied by the Gibbs sampler for the $j$th sub-step of the $m$th iteration (also the $(md + j)$th step) is

$$q^\text{Gibbs}_s(\theta^*|\theta^{(s-1)}) = \begin{cases} p(\theta^*_j|\theta^{(s-1)}_{-j}, y) & \text{if } \theta^*_j = \theta^{(s-1)}_{-j} \\ 0 & \text{otherwise} \end{cases}$$

where $s = md + j$. 
Gibbs Sampler as a Special Case of Metropolis-Hastings

Consider a valid proposal – that is, \( \theta^*_j = \theta_{(s-1)}^j \). Then, the MH acceptance ratio is

\[
\frac{p(\theta^* | y) / q_{j,m}(\theta^* | \theta_{(s-1)})}{p(\theta_{(s-1)} | y) / q_{j,m}(\theta_{(s-1)} | \theta^*)} = \frac{p(\theta^* | y) / p(\theta^*_j | \theta_{(s-1)}^j, y)}{p(\theta_{(s-1)} | y) / p(\theta_{(s-1)}^j | \theta^*_j, y)}
\]

Note that

\[
p(\theta^* | y) = p(\theta^*_j | \theta^*_j, y) p(\theta^*_{(s-1)} | \theta_{(s-1)}, y) = p(\theta^*_j | \theta^*_j, y) p(\theta^*_{(s-1)} | \theta_{(s-1)}, y)
\]

\[
p(\theta_{(s-1)} | y) = p(\theta_{(s-1)}^j | \theta_{(s-1)}^j, y) p(\theta_{(s-1)} | \theta_{(s-1)}, y) = p(\theta_{(s-1)}^j | \theta^*_j, y) p(\theta_{(s-1)} | \theta^*_j, y)
\]

Then,

\[
\frac{p(\theta^* | y) / p(\theta^*_j | \theta_{(s-1)}^j, y)}{p(\theta_{(s-1)} | y) / p(\theta_{(s-1)}^j | \theta^*_j, y)} = \frac{p(\theta_{(s-1)} | y)}{p(\theta_{(s-1)} | y)} = 1
\]

Thus, all proposals are accepted.
Combining Gibbs Sampler and Metropolis-Hastings

- When the dimension of $\theta$ is large, it is often beneficial to work with a partition of the vector $\theta = (\theta_1, ..., \theta_d)$, as in the Gibbs sampler.
- In some cases, however, sampling from some (or all) of the conditional distributions $p(\theta_j | \theta_{-j}, y)$ may be impossible.
- We can construct a specific MH algorithm where we instead draw from a proposal $g(\theta_j | \theta_{-j}, y)$ in cases where we cannot draw from the conditional distribution. Then, the proposal density at the $m$th MH iteration and $j$th sub-step is

$$q(\theta^* | \theta^{(md+j-1)}) = \begin{cases} g(\theta^* | \theta^{(md+j-1)}_{-j}) & \text{if } \theta^*_{-j} = \theta^{(md+j-1)}_{-j} \\ 0 & \text{otherwise} \end{cases}$$