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 $p(\theta \mid y) = \frac{f(y|\theta)p(\theta)}{\int_\Theta f(y|\theta^*)p(\theta^*)\mathrm{d}\theta^*}$

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

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 $p(\theta|y)$: $\left\{\theta^{(m)}\right\}_{m=1}^{M}$

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

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

By a LLN,

$$\widehat{h}_M \xrightarrow{p} \mathbb{E}[h(\theta) \mid y]$$

Possible Solution: Simulation (II)

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

The estimator $\widehat{h}_M$ is still valid.

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

$$\widehat{h}_M \xrightarrow{p} \mathbb{E}[h(\theta) \mid y]$$

Markov Chains

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

$$\Pr\left(\theta^{(j+1)} \mid \theta^{(j)}, ..., \theta^{(1)}\right) = \Pr\left(\theta^{(j+1)} \mid \theta^{(j)}\right)$$

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

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

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

$$p_S(\theta') = \int_\Theta \kappa(\theta'|\theta) p_S(\theta) \, \mathrm{d}\theta$$

Markov Chain Monte Carlo (MCMC)

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

$$\widehat{h}_M = \frac{1}{M} \sum_{m=1}^{M} h\left(\theta^{(m)}\right) \xrightarrow{p} \mathbb{E}[h(\theta) \mid y]$$

- Two popular methods:
  (1) Metropolis-Hastings Algorithm
  (2) Gibbs Sampler

Outline

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 $m = 1, ..., M$

1. Draw θ^* from $q(\theta \mid \theta^{(m-1)})$ and u from $\mathcal{U}(0, 1)$ independently
2. Compute acceptance probability

$$\rho(\theta^* | \theta^{(m-1)}) = \min \left\{ 1, \frac{f(y \mid \theta^*) p(\theta^*) q(\theta^{(m-1)} \mid \theta^*)}{f(y \mid \theta^{(m-1)}) p(\theta^{(m-1)}) q(\theta^* \mid \theta^{(m-1)})} \right\}$$

3. New draw

$$\theta^{(m)} = \begin{cases} \theta^* & \text{if } u \leq \rho(\theta^* | \theta^{(m-1)}) \\ \theta^{(m-1)} & \text{otherwise} \end{cases}$$

Why does it work? (Intuition)

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

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

$$p(\theta' \mid y) \geq p(\theta \mid y) \quad \text{or} \quad p(\theta' \mid y) < p(\theta \mid y)$$

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

$$\min\left\{1, \frac{p(\theta' \mid y)}{p(\theta \mid y)}\right\}$$

Proposal Density

What makes a good proposal density?

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

Main classes for proposal densities:

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

Other Implementation Details

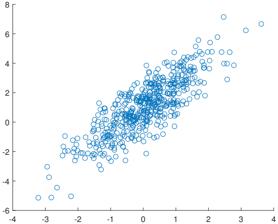
Burn-in

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

Thinning

- Only retain every dth iteration of the chain
- Reduces dependence between draws
  $\rightarrow$ BUT! Average on thinned sequence has greater variance than average over entire sequence
- Possibly useful when computationally-constrained
  $\rightarrow$ 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:

$$\begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 10 & 0 \\ 0 & 5 \end{pmatrix} \right)$$

- Proposal:

$$\begin{pmatrix} \beta_0^* \\ \beta_1^* \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} + \varepsilon, \ \varepsilon \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0.01 & 0 \\ 0 & 0.01 \end{pmatrix} \right)$$

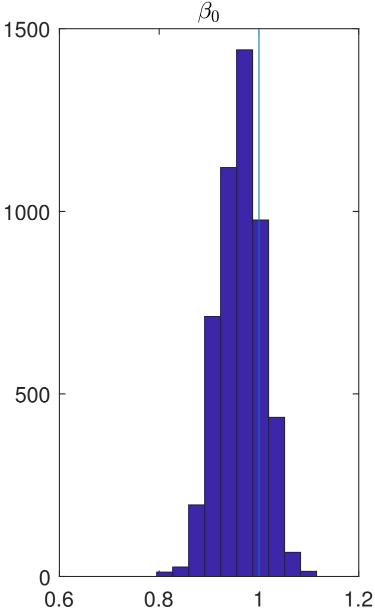
Example: Code (I)

```
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
```

Example: Code (II)

```
% 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



Outline

Gibbs Sampler

Inputs:

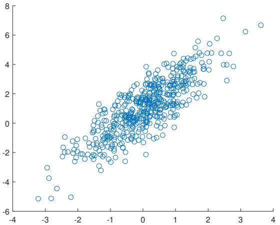
- Partition of the parameter vector $\theta = (\theta_1, \theta_2)$
- Way to draw from the conditional posteriors $p(\theta_1 \mid \theta_2, y)$ and $p(\theta_2 \mid \theta_1, y)$

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

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

Generalizable to a partition $\theta = (\theta_1, ..., \theta_d)$

Example: Normal Regression with Independent N-IG Priors



- Model:

$$y = X\beta + \varepsilon,\ \varepsilon \sim \mathcal{N}(0, \sigma^2 I_n)$$

- Priors:

$$\begin{aligned} \beta &\sim \mathcal{N}(\beta_0, \Sigma_0) \\ \sigma^2 &\sim \text{Inv-Gamma}(a_0, b_0) \end{aligned}$$

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)

$$
\begin{aligned}
\beta \mid \sigma^2, y \sim & \mathcal{N}(\beta_1, \Sigma_1) \\
\sigma^2 \mid \beta, y \sim & \text{Inv-Gamma}(a_1, b_1)
\end{aligned}
$$

with

$$
\begin{aligned}
\Sigma_1 = & \left(\Sigma_0^{-1} + \frac{1}{\sigma^2} X'X \right)^{-1} \\
\beta_1 = & \Sigma_1 \left(\Sigma_0^{-1} \beta_0 + \left(\frac{1}{\sigma^2} X'X \right) \widehat{\beta} \right) \\
\widehat{\beta} = & (X'X)^{-1} X'y \\
a_1 = & \frac{N}{2} + a_0 \\
b_1 = & \left(\frac{1}{b_0} + \frac{1}{2} (y - X\beta)'(y - X\beta) \right)^{-1}
\end{aligned}
$$

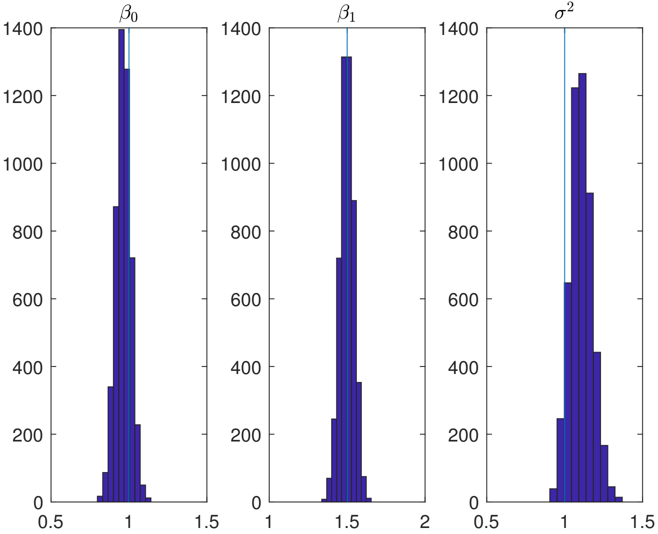
Example: Code (I)

```
% 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)

```
% 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



Gibbs Sampler as a Special Case of Metropolis-Hastings

Define a MH algorithm where for each iteration $m = 1, ..., M$, there are d sub-steps. Entire algorithm has Md steps. The jth sub-step corresponds to an update of the jth partition of the parameter vector.

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

$$q_s^{\text{Gibbs}}(\theta^*|\theta^{(s-1)}) = \begin{cases} p(\theta_j^*|\theta_{-j}^{(s-1)}, y) & \text{if } \theta_{-j}^* = \theta_{-j}^{(s-1)} \\ 0 & \text{otherwise} \end{cases}$$

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

$$\frac{p(\theta^* \mid y)/q^{\text{Gibbs}}_{j,m}(\theta^* \mid \theta^{(s-1)})}{p(\theta^{(s-1)} \mid y)/q^{\text{Gibbs}}_{j,m}(\theta^{(s-1)} \mid \theta^*)} = \frac{p(\theta^* \mid y)/p(\theta^*_j|\theta^{(s-1)}_{-j}, y)}{p(\theta^{(s-1)} \mid y)/p(\theta^{(s-1)}_j \mid \theta^*_{-j}, y)}$$

Note that

$$p(\theta^*|y) = p(\theta^*_j|\theta^*_{-j}, y)p(\theta^*_{-j}|y) = p(\theta^*_j|\theta^{(s-1)}_{-j}, y)p(\theta^{(s-1)}_{-j}|y)$$

$$p(\theta^{(s-1)}|y) = p(\theta^{(s-1)}_j|\theta^{(s-1)}_{-j}, y)p(\theta^{(s-1)}_{-j}|y) = p(\theta^{(s-1)}_j|\theta^*_{-j}, y)p(\theta^{(s-1)}_{-j}|y)$$

Then,

$$\frac{p(\theta^* \mid y)/p(\theta^*_j|\theta^{(s-1)}_{-j}, y)}{p(\theta^{(s-1)} \mid y)/p(\theta^{(s-1)}_j \mid \theta^*_{-j}, y)} = \frac{p(\theta^{(s-1)}_{-j} \mid y)}{p(\theta^{(s-1)}_{-j} \mid y)} = 1$$

Thus, all proposals are accepted.

Combining Gibbs Sampler and Metropolis-Hastings

- When the dimension of θ 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 $p(\theta_j \mid \theta_{-j}, y)$ may be impossible
- We can construct a specific MH algorithm where we instead draw from a proposal $g(\theta_j \mid \theta_{-j}, y)$ in cases where we cannot draw from the conditional distribution. Then, the proposal density at the mth MH iteration and jth sub-step is

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