Lecture 9: Brief reminder

- Problem: Sample from $\pi(x)$, $x \in S$.
- MCMC idea:
  - Construct Markov chain with $\pi(x)$ as limiting distribution.
  - Simulate the Markov chain for a long time so that it has time to converge.
  - Most MCMC samplers are based on reversible Markov chains $\Rightarrow$ Their convergence is proved by checking the detailed balance equation.

Review: Metropolis-Hastings algorithm

Init $x_0 \sim g(x_0)$
for $i = 1, 2, \ldots$ **do**
    Generate a proposal $x^\star \sim Q(x^\star | x_{i-1})$
    $u \sim \mathrm{U}(0, 1)$
    if $u < \underbrace{\min\left(1, \frac{\pi(x^\star)}{\pi(x_{i-1})} \times \underbrace{\frac{Q(x_{i-1}|x^\star)}{Q(x^\star|x_{i-1})}}_{\text{Proposal ratio}}\right)}_{\text{Acceptance probability } \alpha}$ **then**
        $x_i \leftarrow x^\star$
    else
        $x_i \leftarrow x_{i-1}$
    end if
end for

Special cases of the Metropolis-Hastings algorithm

Depending on the choice of $Q(x^\star | x_{i-1})$ different special cases result. In particular, two classes are important

- The independence proposal
- The Metropolis algorithm

Independence proposal

- The proposal distribution does not depend on the current value x_{i-1}

$$Q(x|x_{i-1}) = Q(x).$$

- $Q(x)$ is an approximation to $\pi(x)$
  $\Rightarrow$ Acceptance rate should be close to 1.
- The sampler is closer to rejection sampler. However, here if we reject, then we retain the sample.

Experience:

- Performance is either very good or very bad, usually very bad.
- The tails of the proposal distribution should be at least as heavy as the tails of the target distribution.

The Metropolis algorithm

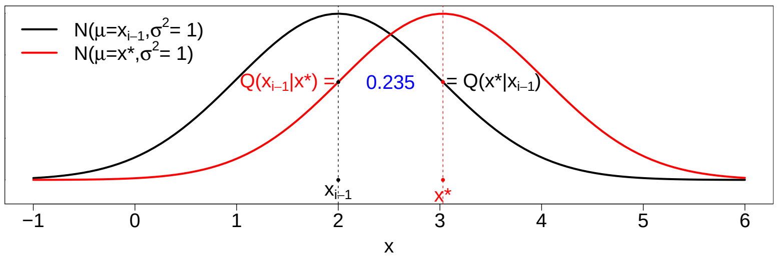
The proposal density is symmetric around the current value, that means

$$Q(x_{i-1}|x^\star) = Q(x^\star|x_{i-1}).$$

Hence,

$$\alpha = \min\left(1, \frac{\pi(x^\star)}{\pi(x_{i-1})} \times \frac{Q(x_{i-1}|x^\star)}{Q(x^\star|x_{i-1})}\right) = \min\left(1, \frac{\pi(x^\star)}{\pi(x_{i-1})}\right)$$

A particular case is the random walk proposal, defined as the current value x_{i-1} plus a random variate of a 0-centred symmetric distribution.



Examples for random walks proposal

Assume x is scalar.

Then all proposal kernels, which add a random variable generated from a zero-symmetrical distribution to the current value x_{i-1}, are random walk proposals. For example:

$$x^\star \sim \mathcal{N}(x_{i-1}, \sigma^2)$$

$$x^\star \sim t_\nu(x_{i-1}, \sigma^2)$$

$$x^\star \sim \mathcal{U}(x_{i-1} - d, x_{i-1} + d)$$

Efficiency of the Metropolis-Hastings algorithm

The efficiency and performance of the Metropolis-Hastings algorithm depends crucially on the relative frequency of acceptance.

An acceptance rate of one is not always good. Consider the random walk proposal:

- Too large acceptance rate $\Rightarrow$ Slow exploration of the target density.
- Too small acceptance rate $\Rightarrow$ Large moves are proposed, but rarely accepted.

Tuning the acceptance rate:

- For random walk proposals, acceptance rates between 20% and 50% are typically recommended. They can be achieved by changing the variance of the proposal distribution.
- For independence proposals a high acceptance rate is desired, which means that the proposal density is close to the target density.

Example: Random walk proposal

Exploration of a standard Gaussian distribution ($\mathcal{N}(0, 1)$) using a random walk Metropolis algorithm. As proposal assume a Gaussian distribution with variance σ^2, where.

- $\sigma = 0.24$
- $\sigma = 2.4$
- $\sigma = 24$

See R-code `demo_mcmcRW.R`.

Example of Rao (1973)

The vector $\boldsymbol{y} = (y_1, y_2, y_3, y_4) = (125, 18, 20, 34)$ is multinomial distributed with probabilities

$$\left\{\frac{1}{2} + \frac{\theta}{4}, \frac{1-\theta}{4}, \frac{1-\theta}{4}, \frac{\theta}{4}\right\}$$

We would like to simulate from the posterior distribution (assuming a uniform prior)

$$f(\theta|\boldsymbol{y}) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}.$$

using MCMC and compare two proposal kernels:

1. independence proposal
2. random walk proposal

See R-code `demo_mcmcRao.R`.

Rao: Independence proposal

$$\theta^\star \sim \mathcal{N}(\mathrm{Mod}(\theta|\boldsymbol{y}), F^2 \times I_p^{-1}), \tag{5}$$

where $\mathrm{Mod}(\theta|\text{data})$ denotes the posterior mode, I_p the negative curvature of the log posterior at the mode, and F a factor to blow up the standard deviation.
Of note, asymptotically the posterior distribution follows (5) for $F = 1$.

Rao: Random walk proposal

$$\theta^\star \sim \mathrm{U}(\theta^{(k)} - d, \theta^{(k)} + d),$$

where $\theta^{(k)}$ denotes the current state of the Markov chain and $d = \sqrt{12}/2 \cdot 0.1$.

Comments on the Metropolis-Hasting algorithm

- A trivial special case results when

  $$Q(x^\star|x_{i-1}) = \pi(x^\star),$$

  That means, we propose realisations from the target distribution. Then $\alpha = 1$ and all proposals are accepted.
- The advantage of the MH-algorithm is that arbitrary proposal kernels can be used. The algorithm will always converge to the target distribution.
- However, the speed of convergence and the dependence between the successive samples depends strongly on the proposal distribution.

Example: Ising/Potts model

Model developed in statistical mechanics (analysis of magnetic material) and used also in image restauration for example.

Let $x = (x^1, \ldots, x^n)$ represent the colors (black/white) in the pixels of a given image, with $x^i \in \{0, 1\}$, where the distribution function is given by

$$\pi(x) = c \cdot \exp\left(-\beta \sum_{i \sim j} I(x^i \neq x^j)\right)$$

where β denotes the interaction parameter, $I(.)$ the indicator function and

$$c = \frac{1}{\sum_x \exp(-\beta \sum_{i\sim j} I(x^i \neq x^j))}.$$

Note: The state space size and hence the number of terms in c is $2^n = 2^{40\,000} \approx 10^{12\,041}$ for a 200×200 grid. Thus, we cannot compute c.

Simualtion using Metropolis-Hastings algorithm

Current state $x = (x^1, \ldots, x^n)$. Propose a new state $y = (y^1, \ldots, y^n)$ as follows:

- draw a node $k \in \{1, 2, \ldots, n\}$ at random
- propose to reverse the value of node k, i.e.

$$y = (x^1, \ldots, x^{k-1}, 1 - x^k, x^{k+1}, \ldots, x^n).$$

Thus

$$Q(y \mid x) = \begin{cases} \frac{1}{n} & \text{if } x \text{ and } y \text{ differ in exactly one node} \\ 0 & \text{else.} \end{cases}$$

Acceptance probability

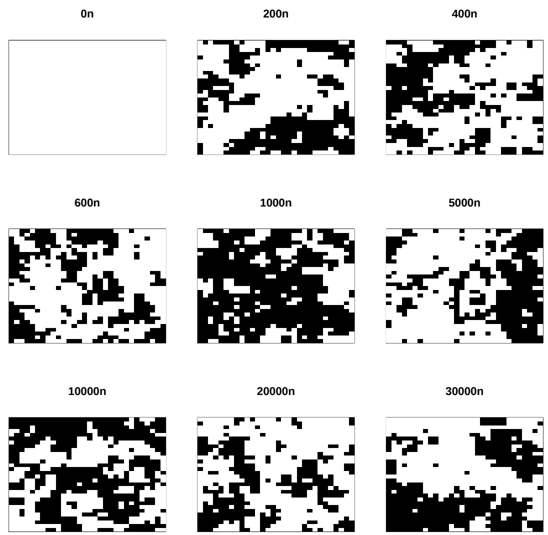
$$\alpha(y \mid x) = \min\left\{1, \frac{\pi(y)}{\pi(x)} \cdot \frac{Q(x \mid y)}{Q(y \mid x)}\right\}$$

$$= \min\left\{1, \frac{\exp\left(-\beta \sum_{i\sim j} I(y^i \neq y^j)\right)}{\exp\left(-\beta \sum_{i\sim j} I(x^i \neq x^j)\right)} \cdot \frac{\frac{1}{n}}{\frac{1}{n}}\right\}$$

$$= \min\left\{1, \frac{\exp\left(-\beta \sum_{i\sim k} I(x^i \neq 1 - x^k)\right)}{\exp\left(-\beta \sum_{i\sim k} I(x^i \neq x^k)\right)}\right\}$$

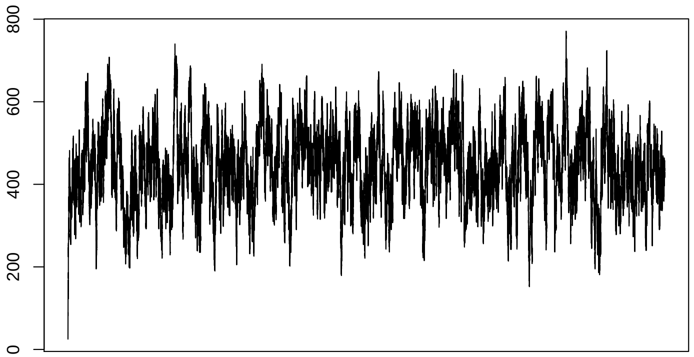
Ising example

$\beta = 0.8$:



Ising example: Traceplot

Traceplot showing the number of 1s.



MCMC and iterative conditioning

The use of the MH-algorithms gains on importance when it is applied iteratively on components of x.

Let $\boldsymbol{x}$ be decomposed by several (for simplicity scalar) components.

$$\boldsymbol{x} = (x^1, \ldots, x^p)$$

Now the MH-algorithm is applied iteratively on the components x^j, conditioning on the current values of $\boldsymbol{x}^{-j}$ with

$$\boldsymbol{x}^{-j} = (x^1, \ldots, x^{j-1}, x^{j+1}, \ldots, x^p)$$

MCMC and iterative conditioning

To be concrete, one uses

- a proposal kernel $Q(x^{j,\star}|x^j_{i-1}, \boldsymbol{x}^{-j}_{i-1})$, $j = 1, \ldots, p$.
- with acceptance probability

$$\alpha = \min\left(1, \frac{\pi(x^{j,\star}|\boldsymbol{x}^{-j}_{i-1})}{\pi(x^j_{i-1}|\boldsymbol{x}^{-j}_{i-1})} \times \frac{Q(x^j_{i-1}|x^{j,\star}, \boldsymbol{x}^{-j}_{i-1})}{Q(x^{j,\star}|x^j_{i-1}, \boldsymbol{x}^{-j}_{i-1})}\right)$$

This algorithm converges to the stationary distribution with density $\pi(\boldsymbol{x})$, as long as all components are arbitrary often updated.

Conditional densities

Of note, the acceptance probability α only uses the full conditional densities $\pi(x^j|\boldsymbol{x}^{-j})$, $j = 1, \ldots, p$, and not the joint density $\pi(\boldsymbol{x})$.
Both are related as follows

$$\pi(x^j|\boldsymbol{x}^{-j}) = \frac{\pi(\boldsymbol{x})}{\pi(\boldsymbol{x}^{-j})} \propto \pi(\boldsymbol{x})$$

Thus, the (non-normalised) conditional densities of $x^j|\boldsymbol{x}^{-j}$ can be directly derived from $\pi(\boldsymbol{x})$ by omitting all multiplicative factors, that do not depend on x^j.

Gibbs sampling

Are all conditional densities $\pi(x^j|\boldsymbol{x}^{-j})$, $j = 1, \ldots, p$ *standard* it seems natural to use those as proposal kernel, i.e.

$$Q(x^{j,\star}|x^j_{i-1}, \boldsymbol{x}^{-j}_{i-1}) = \pi(x^{j,\star}|\boldsymbol{x}^{-j}_{i-1})$$

In this case, we get $\alpha = 1$ which leads to the well known Gibbs sampler, which updates parameters iteratively by sampling from the corresponding full conditional distributions.

Why is the acceptance rate 1?

For ease of notation let x denote the current state and $x^\star$ the proposed new state where we update the $j-$th component of x, so that:

$$x = (x^1, \ldots, x^{j-1}, x^j, x^{j+1}, \ldots, x^p)^\top$$
$$x^\star = (x^1, \ldots, x^{j-1}, x^{\star,j}, x^{j+1}, \ldots, x^p)^\top$$

where $x^{\star,j}$ denotes the propsed value for the $j-$th component. Then

$$\begin{aligned}\frac{\pi(x^\star)}{\pi(x)} \cdot \frac{Q(x \mid x^\star)}{Q(x^\star \mid x)} &= \frac{\pi(x^{\star,j} \mid x^{\star,-j})\pi(x^{\star,-j})}{\pi(x^j \mid x^{-j})\pi(x^{-j})} \cdot \frac{Q(x \mid x^\star)}{Q(x^\star \mid x)} \\ &= \frac{\pi(x^{\star,j} \mid x^{-j})\pi(x^{-j})}{\pi(x^j \mid x^{-j})\pi(x^{-j})} \cdot \frac{Q(x \mid x^\star)}{Q(x^\star \mid x)} \\ &= \frac{\pi(x^{\star,j} \mid x^{-j})\pi(x^{-j})}{\pi(x^j \mid x^{-j})\pi(x^{-j})} \cdot \frac{\pi(x^j \mid x^{\star,-j})}{\pi(x^{\star,j} \mid x^{-j})} \\ &= 1\end{aligned}$$

Gibbs-Sampling algorithm

Idea: Sequentially sampling from univariate conditional distributions (which are often available in closed form).

1. Select starting values $\boldsymbol{x}_0$ and set $i = 0$.
2. Repeatedly:

$$\begin{aligned}&\text{Sample} \quad x^1_{i+1}|\cdot \sim \pi(x^1|x^2_i, \ldots, x^p_i) \\ &\text{Sample} \quad x^2_{i+1}|\cdot \sim \pi(x^2|x^1_{i+1}, x^3_i, \ldots, x^p_i) \\ &\vdots \\ &\text{Sample} \quad x^{p-1}_{i+1}|\cdot \sim \pi(x^{p-1}|x^1_{i+1}, x^2_{i+1}, \ldots, x^{p-2}_{i+1}, x^p_i) \\ &\text{Sample} \quad x^p_{i+1}|\cdot \sim \pi(x^p|x^1_{i+1}, \ldots, x^{p-1}_{i+1})\end{aligned}$$

   where $|\cdot$ denotes conditioning on the most recent updates of all other elements of $\boldsymbol{x}$.
3. Increment i and go to step 2.

Remarks on Gibbs sampling

- High dimensional updates of $\boldsymbol{x}$ can be boiled down to scalar updates.
- Visiting schedule: Various approaches exist (and can be justified) to ordering the variables in the sampling loop. One approach is random sweeps: variables are chosen at random to resample.
- Gibbs sampling assumes that it is easy to sample from the full-conditional distribution. This is sometimes not so easy. Alternatively, a Metropolis-Hastings proposal can be used for the j-th component, i.e. Metropolis-within-Gibbs $\Rightarrow$ Hybrid Gibbs sampler.

Remarks on Gibbs sampling

- Blocking or grouping is possible, that means not all elements of $\boldsymbol{x}$ are treated individually. Might be useful when elements of $\boldsymbol{x}$ are correlated.
- Care must be taken when improper prior are used, which may lead to an improper posterior distribution. Impropriety implies that there does not exist a joint density to which the full-conditional distributions correspond.

Example: Deriving full-conditionals

Assume $y_i|\mu,\kappa \sim \mathcal{N}(\mu,\kappa^{-1})$, $i = 1,\dots,n$. As prior for μ and κ we choose a normal and gamma distribution, respectively, where:

$$\mu \sim \mathcal{N}(\mu_0, \kappa_0^{-1})$$
$$\kappa \sim \mathcal{G}(a, b)$$

The full-conditionals are

$$\mu|\kappa,\boldsymbol{y} \sim \mathcal{N}\left(\frac{\mu_0\kappa_0 + \bar{y}n\kappa}{\kappa_0 + n\kappa}, (\kappa_0, n\kappa)^{-1}\right)$$
$$\kappa|\mu,\boldsymbol{y} \sim \mathcal{G}\left(a + \frac{n}{2}, b + \frac{1}{2}\sum_{i=1}^{n}(y_i - \mu)^2\right)$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ denotes the mean over all y. (see lecture 7 for details).