STA2311: Advanced Computational Methods for Statistics I

Class 7: MCMC Basics

Radu Craiu Robert Zimmerman

University of Toronto

October 31, 2023

A Quick Review (?) of Markov Chains

Section 1

Introduction

The Need for MCMC

- Recall the setup from Class 6: we have some target density f that we wish to generate samples from, usually for the purpose of estimating some $\mathbb{E}_f[h]$
- We learned several methods of generating *exact* samples from f
  - e.g., rejection sampling, the inverse cdf method, distribution-specific techniques
- However, these techniques require relatively detailed knowledge about f
- For example, the rejection sampler requires knowing some easy-to-sample density g such that $f \leq c \cdot g$
  - The bigger the "gap" between f and g, the more we reject (and the less efficient the sampler is)
- Thus, these only work for relatively simple targets
- In particular, they fail miserably for high-dimensional targets!

When the Target Cannot be Sampled From Directly

- For a simple example, consider Bayesian logistic regression: we have independent observations $Y_1, \ldots, Y_n$ and covariates $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$ with $Y_i \mid \boldsymbol{x}_i \sim \text{Bernoulli}(\sigma(\beta^\top \boldsymbol{x}_i))$, where $\sigma(x) = (1 + e^{-x})^{-1}$
- We place a $\mathcal{N}_p(\boldsymbol{m}_0, \boldsymbol{S}_0)$ prior on β
- The posterior distribution satisfies

$$p(\beta \mid \boldsymbol{y}) \propto p(\beta) \cdot p(\boldsymbol{y} \mid \beta)$$
$$\propto \exp\left(-\frac{1}{2}(\beta - \boldsymbol{m}_0)^\top \boldsymbol{S}_0^{-1}(\beta - \boldsymbol{m}_0)\right) \cdot \prod_{i=1}^{n} \sigma(\beta^\top \boldsymbol{x}_i)^{y_i}(1 - \sigma(\beta^\top \boldsymbol{x}_i))^{1-y_i}$$

- This is completely impossible to sample from directly, even if we could calculate the normalizing constant (which we can't)
- Even if we could, for large p rejection sampling and importance sampling will fail miserably!

Two Basic Algorithms

- *Markov chain Monte Carlo (MCMC)* is, far and away, the most popular method of generating a sample $\boldsymbol{X} = (X_1, \ldots, X_d)$ from a complicated d-dimensional target distribution f
- For the two basic algorithms we will study today, very little is asked of us
- For the *Metropolis-Hastings* algorithm, we only need to know the functional form of f up to a constant
  - That is, we don't need the normalizing constant $\int f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$
- For the *Gibbs sampler*, we only need to know the conditional distribution of $X_h \mid X_1, \ldots, X_{h-1}, X_{h+1}, \ldots, X_d$ for each $h = 1, \ldots, d$
  - We will see that even this can be relaxed

The Cost

- Virtually all other MCMC algorithms in use are built upon either of these two techniques
- However, their relative ease of use comes at a cost
- The samples $\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \ldots$ they generate are only *approximately* distributed according to f
- And they aren't independent!
- Fortunately, there are well-studied methods for minimizing these issues, which we will see in Class 8
- Moreover, the *ergodic theorem* still guarantees that when the simulated chain is ergodic, the SLLN still holds:

$$\frac{1}{T}\sum_{t=0}^{T} h(\boldsymbol{X}^{(t)}) \xrightarrow{as} \mathbb{E}_f[h]$$

The Idea Behind MCMC

- The basic idea is straightforward
- We will construct an (irreducible, aperiodic) Markov chain $\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \ldots$ whose stationary distribution is f
- The samples we actually use are the observed values of $\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \ldots$
  - Or some subset thereof — to be discussed in Class 8
- Thus, the distribution of $\boldsymbol{X}^{(M)}$ only approaches f as $M \to \infty$
- But with a well-designed MCMC algorithm, we will get there tolerably quickly
  - More specifically, the chain will *mix* quickly — again, Class 8

Section 2

The Metropolis-Hastings Algorithm

Motivation

- As always, we want to sample from a target density f
- Assume we have at our disposal a conditional distribution $q(\cdot \mid \boldsymbol{x})$ which is easy to simulate from, such that the ratio

$$\frac{f(\boldsymbol{y})}{q(\boldsymbol{y} \mid \boldsymbol{x})}$$

  is known up to a constant *independent* of $\boldsymbol{x}$
- The distribution q is called the *proposal distribution*

The Algorithm Itself

- Then the *Metropolis-Hastings algorithm* proceeds as follows:

  1. Start with an initial $\boldsymbol{X}^{(0)} = \boldsymbol{x}^{(0)}$
  2. For each $t \geq 1$, given $\boldsymbol{X}^{(t)} = \boldsymbol{x}^{(t)}$,
     a. Generate $\boldsymbol{Y} \sim q(\cdot \mid \boldsymbol{x}^{(t)})$
     b. Accept $\boldsymbol{X}^{(t+1)} = \boldsymbol{Y}$ with probability

$$\min\left\{ \frac{f(\boldsymbol{Y})}{q(\boldsymbol{Y} \mid \boldsymbol{x}^{(t)})} \cdot \frac{q(\boldsymbol{x}^{(t)} \mid \boldsymbol{Y})}{f(\boldsymbol{x}^{(t)})}, 1 \right\};$$

  otherwise, take $\boldsymbol{X}^{(t+1)} = \boldsymbol{x}^{(t+1)}$

- The function

$$\rho(\boldsymbol{x}, \boldsymbol{y}) = \min\left\{ \frac{f(\boldsymbol{y})}{q(\boldsymbol{y} \mid \boldsymbol{x})} \cdot \frac{q(\boldsymbol{x} \mid \boldsymbol{y})}{f(\boldsymbol{x})}, 1 \right\}$$

  is called the *Metropolis-Hastings acceptance probability*

- Every variation/extension of the basic MH algorithm ultimately accepts new proposals with some variation of this acceptance probability

This Does What We Want

- To show that the Metropolis-Hastings algorithm has f as its stationary distribution, it's enough to show that its transition kernel satisfies the detailed balance condition with f
- Indeed, the transition kernel is

$$K(\boldsymbol{x}, \boldsymbol{y}) = \rho(\boldsymbol{x}, \boldsymbol{y}) \cdot q(\boldsymbol{y} \mid \boldsymbol{x}) + (1 - r(\boldsymbol{x})) \cdot \delta_{\boldsymbol{x}}(\boldsymbol{y})$$

  where $r(\boldsymbol{x}) = \int \rho(\boldsymbol{x}, \boldsymbol{y}) \cdot q(\boldsymbol{y} \mid \boldsymbol{x}) \, \mathrm{d}\boldsymbol{y}$

- It is not hard to check that

$$\rho(\boldsymbol{x}, \boldsymbol{y}) \cdot q(\boldsymbol{y} \mid \boldsymbol{x}) \cdot f(\boldsymbol{x}) = \rho(\boldsymbol{y}, \boldsymbol{x}) \cdot q(\boldsymbol{x} \mid \boldsymbol{y}) \cdot f(\boldsymbol{y})$$

  and

$$(1 - r(\boldsymbol{x})) \cdot \delta_{\boldsymbol{x}}(\boldsymbol{y}) \cdot f(\boldsymbol{x}) = (1 - r(\boldsymbol{y})) \cdot \delta_{\boldsymbol{y}}(\boldsymbol{x}) \cdot f(\boldsymbol{y})$$

  which together establish the claim

Choosing a Proposal Distribution

- A good proposal distribution will yield in a high acceptance rate, but *also* explore the state space reasonably quickly
- These two desiderata are fundamentally in opposition with each other
- Thus, we must strike a balance
  - We will see in Class 8 that certain proposals can be *adaptive* – i.e., they can change at each iteration
- For now, we will examine two special cases of Metropolis-Hastings which correspond to particular kinds of proposals

Independence Sampler

- One important special case of Metropolis-Hastings occurs when $q(\boldsymbol{y} \mid \boldsymbol{x}) = g(\boldsymbol{y})$ for some density g
- That is, the proposal is *independent* of the current value of the chain
- The resulting *independence sampler* proceeds as follows:
  1. Start with an initial $\boldsymbol{X}^{(0)} = \boldsymbol{x}^{(0)}$
  2. For each $t \geq 1$, given $\boldsymbol{X}^{(t)} = \boldsymbol{x}^{(t)}$,
     a. Generate $\boldsymbol{Y} \sim g(\cdot)$
     b. Accept $\boldsymbol{X}^{(t+1)} = \boldsymbol{Y}$ with probability

$$\left\{ \frac{f(\boldsymbol{Y})}{g(\boldsymbol{Y})} \cdot \frac{g(\boldsymbol{x}^{(t)})}{f(\boldsymbol{x}^{(t)})}, 1 \right\};$$

otherwise take $\boldsymbol{X}^{(t+1)} = \boldsymbol{x}^{(t)}$

Random-Walk Metropolis

- When $d = 1$, another important special case occurs when the proposal can be written as $q(y - x) = g(|y - x|)$ for some distribution g which is symmetric around zero
  - Classic examples include $\mathcal{N}(0, \sigma^2)$ and Unif$(-\delta, \delta)$ for some $\sigma^2, \delta > 0$
- As the name suggests, we simply take a random step away from $x^{(t)}$ in order to explore the state space
- The resulting *random-walk Metropolis algorithm* proceeds as follows:
  1. Start with an initial $X^{(0)} = x^{(0)}$ and a symmetric proposal g
  2. For each $t \geq 1$, given $X^{(t)} = x^{(t)}$,
     a. Generate $Y \sim g(|\cdot - x^{(t)}|)$
     b. Accept $X^{(t+1)} = Y_t$ with probability $\left\{ \frac{f(Y_t)}{f(x^{(t)})}, 1 \right\}$; otherwise take $X^{(t+1)} = x^{(t)}$
- While the idea extends naturally to $d > 1$, it suffers heavily from the curse of dimensionality

Example: Simulating from the Standard Normal Distribution

```
set.seed(2311)

delta <- 0.5

TT <- 15000
x <- 0*1:TT
x[1] <- runif(n=1, -1, 1)

for (t in 1:TT) {
  Yt <- runif(n=1, x[t]-delta, x[t]+delta)
  if (runif(n=1) < dnorm(Yt)/dnorm(x[t])) {
    x[t+1] <- Yt
  } else {
    x[t+1] <- x[t]
  }
}

hist(x)
```

Unconstrained Sampling

- What happens if we want to run random-walk Metropolis (or more general Metropolis-Hastings) on a target with bounded support?
  - Or a support with constraints, such as the standard simplex?
- We could simply reject proposals which fall outside the support, but this is incredibly wasteful
- Instead, it is usually a better idea to transform the target to an unconstrained one (as in Class 2)
- But this requires a bit of work...

Section 3

Metropolis-Hastings: Variations

Basic Issues

- In any MCMC algorithm, we are always faced with the basic task of reaching convergence in a reasonable amount of time
- In particular, the algorithm may be very slow to converge to the target f if the proposal q is chosen poorly
- For example, in random-walk Metropolis, choosing σ^2 too small will cause the algorithm to explore the state space very slowly; on the other hand, choosing σ^2 too large will result in a large number of rejections

Problems with High Dimensions

- High-dimensional posterior distributions typically have many modes
- Thus, it is very easy for the basic Metropolis-Hastings algorithm to get stuck in a local mode
- Since we don't know the locations of these modes, we can't choose the proposal q to account for them
- Moreover, in high dimensions most regions of the state-space have very low probability, which just makes things worse
- Some variations, such as *multiple-try Metropolis*, sample multiple proposals at each step in order to speed things up
  - Although (at most) one sampled proposal is actually accepted, so there is a heavy computational cost

Multiple-Try Metropolis

- Given some non-negative symmetric function $\lambda(\boldsymbol{x}, \boldsymbol{y})$, define the weight function

$$w(\boldsymbol{x}, \boldsymbol{y}) \propto f(\boldsymbol{x}) \cdot q(\boldsymbol{x}, \boldsymbol{y}) \cdot \lambda(\boldsymbol{x}, \boldsymbol{y})$$

- The basic *multiple-try Metropolis* algorithm proceeds as follows:
  1. Start with an initial $\boldsymbol{X}^{(0)} = \boldsymbol{x}^{(0)}$
  2. For each $t \geq 1$, given $\boldsymbol{X}^{(t)} = \boldsymbol{x}^{(t)}$,
     a. Draw $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_k \overset{iid}{\sim} q(\cdot \mid \boldsymbol{x}^{(t)})$
     b. Sample $\boldsymbol{Y}$ from $\{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_k\}$ according to the weights $w(\boldsymbol{Y}_1, \boldsymbol{x}^{(t)}), \ldots, w(\boldsymbol{Y}_k, \boldsymbol{x}^{(t)})$
     c. Sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{k-1} \sim q(\cdot \mid \boldsymbol{Y})$ and set $\boldsymbol{x}_k = \boldsymbol{x}^{(t)}$
     d. Accept $\boldsymbol{X}^{(t+1)} = \boldsymbol{Y}$ with probability

$$\min\left\{ \frac{w(\boldsymbol{y}_1, \boldsymbol{x}^{(t)}) + \cdots + w(\boldsymbol{y}_k, \boldsymbol{x}^{(t)})}{w(\boldsymbol{x}_1, \boldsymbol{y}) + \cdots + w(\boldsymbol{x}_k, \boldsymbol{y})}, 1 \right\};$$

     otherwise take $\boldsymbol{X}^{(t+1)} = \boldsymbol{x}^{(t)}$

- For details, see Liu et al. [2000]

Problems with Evaluating the Target

- So far, we have assumed that we can actually evaluate the target f up to a normalizing constant
- But sometimes this is not even the case!
- However, we might still be able to produce an *unbiased estimate* of $f(\boldsymbol{x})$ for each particular $\boldsymbol{x}$
- That is, for any $\boldsymbol{x}$, we can produce an estimator $\hat{f}(\boldsymbol{x})$ such that $\mathbb{E}\left[\hat{f}(\boldsymbol{x})\right] = f(\boldsymbol{x})$

Pseudo-Marginal Metropolis–Hastings

- It turns out that we can simply replace $f(\boldsymbol{x})$ in the original MH algorithm with an unbiased estimator
- The *pseudo-marginal Metropolis-Hastings algorithm* proceeds as follows:
  1. Start with an initial $\boldsymbol{X}^{(0)} = \boldsymbol{x}^{(0)}$
  2. For each $t \geq 1$, given $\boldsymbol{X}^{(t)} = \boldsymbol{x}^{(t)}$,
     a. Generate $\boldsymbol{Y} \sim q(\cdot \mid \boldsymbol{x}^{(t)})$
     b. Accept $\boldsymbol{X}^{(t+1)} = \boldsymbol{Y}$ with probability

$$\min\left\{ \frac{\hat{f}(\boldsymbol{Y})}{q(\boldsymbol{Y} \mid \boldsymbol{x}^{(t)})} \cdot \frac{q(\boldsymbol{x}^{(t)} \mid \boldsymbol{Y})}{\hat{f}(\boldsymbol{x}^{(t)})}, 1 \right\};$$

otherwise take $\boldsymbol{X}^{(t+1)} = \boldsymbol{x}^{(t)}$

Section 4

Gibbs Sampling

Multidimensional Targets

- Suppose that the target distribution f is multidimensional
- As we saw in Class 6, sampling from non-trivial d-dimensional distributions becomes more difficult as d becomes large
  - Mainly due to the complicated dependencies among the components of $\boldsymbol{X} = (X_1, \ldots, X_d) \sim f$
- This is especially true in Bayesian statistics, where we are interested in high-dimensional posteriors...
  - ...of large vectors of parameters in a model
  - ...of latent variables in hierarchical models
- However, there are sometimes cases where the *conditional distribution* of $X_h \mid \boldsymbol{X}_{-h}$ is available for each $h \in \{1, \ldots, d\}$
  - Here $\boldsymbol{X}_{-h} := (X_1, \ldots, X_{h-1}, X_{h+1}, \ldots, X_d)$

Conditional Distributions

- For example, if $\boldsymbol{X} = (X_1, X_2) \sim \mathcal{N}_2\left(\begin{pmatrix}\mu_1 \\ \mu_2\end{pmatrix}, \begin{pmatrix}\sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22}\end{pmatrix}\right)$, then

$$X_1 \mid X_2 = x_2 \sim \mathcal{N}\left(\mu_1 + \frac{\sigma_{12}}{\sigma_{22}}(x_2 - \mu_2), \sigma_{11} - \frac{\sigma_{12}^2}{\sigma_{22}}\right)$$

- Formulas exist for arbitrary subvectors of $\mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ conditional on other subvectors
  - Usually these involve block partitions of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$
- When the joint distribution of $\boldsymbol{X}$ is non-normal, tedious calculations are often needed to determine the relevant conditional distributions (if they even have known forms)
- But for now, assume we have the ability to determine all of the conditional distributions of $X_h \mid \boldsymbol{X}_{-h}$

The Algorithm Itself

- If $\boldsymbol{X} \sim f$, denote the conditional density of $X_h \mid \boldsymbol{X}_{-h} = \boldsymbol{x}_{-h}$ by

$$f_h(\cdot \mid x_1, \ldots, x_{h-1}, x_{h+1}, \ldots, x_d) := \frac{f(x_1, \ldots, \cdot, \ldots, x_d)}{\int f(x_1, \ldots, x, \ldots, x_d)\, \mathrm{d}x}$$

  - Which we assume we know how to sample from!

- The basic *Gibbs sampler* is as follows:

  1. Initialize the process at $\boldsymbol{X}^{(0)}$
  2. For $t \geq 1$, sample $\boldsymbol{X}^{(t)}$ as follows:

  - For $1 \leq h \leq d$, sample $X_h^{(t)} \sim f_h(\cdot \mid x_1^{(t)}, \ldots, x_{h-1}^{(t)}, x_{h+1}^{(t-1)}, \ldots, x_d^{(t-1)})$
  - Accept $\boldsymbol{X}^{(t)} = (X_1^{(t)}, \ldots, X_d^{(t)})$

This Does What We Want

Example: Univariate Normal Posterior

- Suppose $X_1, \ldots, X_n \overset{iid}{\sim} \mathcal{N}(\mu, \sigma^2)$ and we impose a non-informative prior $p(\mu, \sigma^2) \propto 1/\sigma^2$
- The posterior we want to sample from is

$$p(\mu, \sigma^2 \mid \boldsymbol{x}) \propto (\sigma^2)^{-(\frac{n}{2}+1)} \cdot \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2 \right)$$

- We need to determine $p(\mu \mid \tau^2, \boldsymbol{x})$ and $p(\tau^2 \mid \mu, \boldsymbol{x})$

Example: Univariate Normal Posterior (Continued)

- We have

$$p(\mu \mid \sigma^2, \boldsymbol{x}) \propto$$

so $\mu \mid \sigma^2, \boldsymbol{x} \sim \mathcal{N}\left(\bar{x}, \frac{\sigma^2}{n}\right)$

- And

$$p(\sigma^2 \mid \mu, \boldsymbol{x}) \propto$$

so $\sigma^2 \mid \mu, \boldsymbol{x} \sim \text{InvGamma}\left(\frac{n}{2}, \frac{1}{2}\sum_{i=1}^{n}(x_i - \mu)^2\right)$

- So the Gibbs sampler samples $(\mu^{(t)}, \sigma^{2(t)})$ via

$$\mu^{(t)} \sim \mathcal{N}\left(\bar{x}, \frac{\sigma^{2(t-1)}}{n}\right)$$

$$\sigma^{2(t)} \sim \text{InvGamma}\left(\frac{n}{2}, \frac{1}{2}\sum_{i=1}^{n}(x_i - \mu^{(t)})^2\right)$$

Example: Univariate Normal Posterior (Continued)

```r
set.seed(2311)

n <- 100
x <- rnorm(n=n, mean=5, sd=3)
xbar <- mean(x)

TT <- 1000
theta <- matrix(0L, nrow=2, ncol=TT)
rownames(theta) <- c("mu", "sigma2")
theta[,1] <- c(0, 1)

for (t in 2:TT) {
  theta[1,t] <- rnorm(n=1, mean=xbar, sd=sqrt(theta[2,t-1]/n))
  theta[2,t] <- 1/rgamma(n=1, shape=n/2, rate=sum((x-theta[1,t])^2)/2)
}

plot(theta[1,100:TT], type="l")
plot(theta[2,100:TT], type="l")
```

Systematic Scan and Random Scan

- In higher dimensional problems, we can sample the components of $\boldsymbol{X}$ in any order we please
- That is, the sampling step at time t can be replaced by the following, for any fixed permutation σ of $(1, \ldots, d)$:
- "For $1 \leq h \leq d$, sample

$$X_{\sigma(h)}^{(t)} \sim f_{\sigma(h)}(\cdot \mid x_{\sigma(1)}^{(t)}, \ldots, x_{\sigma(h-1)}^{(t)}, x_{\sigma(h+1)}^{(t-1)}, \ldots, x_{\sigma(d)}^{(t-1)})$$

  and then accept $\boldsymbol{X}^{(t)} = (X_1^{(t)}, \ldots, X_d^{(t)})$"
- This version of the Gibbs sampler is called *systematic scan* (or *deterministic scan*)

Systematic Scan and Random Scan (Continued)

- In contrast, *random scan* (or *sequential scan*) randomly chooses a single index at time t
- That is, the sampling step at time t is is replaced by the following:
- "Sample $H \sim \text{Unif}\{1, \ldots, d\}$ and then sample

$$X_H^{(t)} \sim f_H(\cdot \mid x_1^{(t-1)}, \ldots, x_{H-1}^{(t-1)}, x_{H+1}^{(t-1)}, \ldots, x_d^{(t-1)})$$

  and then accept $\boldsymbol{X}^{(t)} = (X_1^{(t-1)}, \ldots, X_H^{(t)}, \ldots, X_d^{(t-1)})$"
- Surprisingly, random scan is more theoretically supported [He et al., 2016]
  - But systematic scan is clearly computationally cheaper!

Section 5

Gibbs Sampling: Variations

Gibbs Sampling: Variations

- The basic Gibbs sampler only works if all of the conditional distributions were available to sample from
- In practice, this is rarely the case
- We will discuss three modifications of the Gibbs sampler, each of which helps to circumvent this issue in different ways

Blocked Gibbs

- Suppose that sampling from the conditional distribution of some subvector $\boldsymbol{X}_{1:h} = (X_1, X_2, \ldots, X_h)$ of $\boldsymbol{X}$ is easier than sampling from each of the conditional distributions of the marginals $X_1, \ldots, X_h$
- Of course this applies to any subcollection of $X_1, \ldots, X_d$; WLOG we can re-order the components into the "block" as above
- Let $f_{1:h}(\cdot, \ldots, \cdot \mid x_{h+1}, \ldots, x_d) = \frac{f(\cdot,\ldots,\cdot,x_{h+1},\ldots,x_d)}{\int\int\int f(x_1,\ldots,x_d)\,\mathrm{d}\boldsymbol{x}_{1:h}}$
- Then the *blocked Gibbs sampler* proceeds as follows:
  1. Initialize the process at $\boldsymbol{X}^{(0)}$
  2. For $t \geq 1$, sample $\boldsymbol{X}^{(t)}$ as follows:
     a. Sample $\boldsymbol{X}_{1:h}^{(t)} \sim f(\cdot, \ldots, \cdot \mid x_{h+1}^{(t-1)}, \ldots, x_d^{(t-1)})$
     b. For $h+1 \leq i \leq d$, sample $X_i^{(t)} \sim f_h(\cdot \mid x_1^{(t)}, \ldots, x_h^{(t)}, x_{h+1}^{(t-1)}, \ldots, x_d^{(t-1)})$

Collapsed Gibbs

- Suppose that sampling from all of the conditional distributions of $\boldsymbol{X}$ was difficult, but the following conditions hold:
  - There exists some component X_h such that $f_h(\cdot \mid x_1, \ldots, x_{h-1}, x_{h+1}, \ldots, x_d)$ *is* easy to sample from
  - Running a Gibbs a sampler to sample from the "marginal" distribution $f_{-h}(\boldsymbol{x}_{-h}) := \int f(\boldsymbol{x}) \, \mathrm{d}x_h$ is easy
- For $i \neq h$, let

$$f_{-h;i}(\cdot \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d) = \frac{f_{-h}(x_1,\ldots,x_{i-1},\cdot,x_{i+1},\ldots,x_d)}{\int f_{-h}(x_1,\ldots,x_{i-1},x,x_{i+1},\ldots,x_d) \, \mathrm{d}x}$$

- Then the *collapsed Gibbs sampler* proceeds as follows:
  1. Initialize the process at $\boldsymbol{X}^{(0)}$
  2. For $t \geq 1$, sample $\boldsymbol{X}^{(t)}$ as follows:
     a. For $i \neq h$, sample $X_i^{(t)} \sim f_{-h;i}(\cdot \mid x_1^{(t)}, \ldots, x_{i-1}^{(t)}, x_{i+1}^{(t-1)}, \ldots, x_d^{(t-1)})$
     b. Sample $X_h^{(t)} \sim f_h(\cdot \mid x_1^{(t)}, \ldots, x_{h-1}^{(t)}, x_{h+1}^{(t)}, \ldots, x_d^{(t)})$

Metropolis-within-Gibbs

- What happens if we replace a conditional sampling step within the Gibbs sampler with a Metropolis-Hastings update step?
- That is, instead of trying to sample from some $f_h(\cdot \mid x_1^{(t)}, \ldots, x_{h-1}^{(t)}, x_{h+1}^{(t-1)}, \ldots, x_d^{(t-1)})$, we sample $Y_h^{(t)}$ from some proposal $q_h(\cdot \mid x_1^{(t)}, \ldots, x_{h-1}^{(t)}, x_{h+1}^{(t-1)}, \ldots, x_d^{(t-1)})$, and accept $X_h^{(t)} = Y_h^{(t)}$ according to the Metropolis-Hastings acceptance probability?
- We can do this for some components (with different proposals q_h), and leave other components alone if they're easy to sample directly
- The resulting algorithm is called *Metropolis-within-Gibbs*
- Amazingly, all of the theory still caries through!

Metropolis-within-Gibbs (Continued)

- The *Metropolis-within-Gibbs* algorithm proceeds as follows:
  1. Start with an initial $\boldsymbol{X}^{(0)} = \boldsymbol{x}^{(0)}$
  2. For each $t \geq 1$, given $\boldsymbol{X}^{(t)} = \boldsymbol{x}^{(t)}$, sample $\boldsymbol{X}^{(t+1)}$ by doing the following for each $1 \leq h \leq d$:
     a. Draw $Y_h \sim q_h(\cdot \mid x_1^{(t+1)}, \ldots, x_{h-1}^{(t+1)}, x_h^{(t)}, x_{h+1}^{(t)} \ldots, x_d^{(t)})$
     b. Accept $X_h^{(t+1)} = Y_h$ with probability

$$\min\left\{ \frac{\left(\frac{f_h(Y_h \mid x_1^{(t+1)}, \ldots, x_{h-1}^{(t+1)}, x_{h+1}^{(t)}, \ldots, x_d^{(t)})}{q_h(Y_h \mid x_1^{(t+1)}, \ldots, x_{h-1}^{(t+1)}, x_h^{(t)}, x_{h+1}^{(t)} \ldots, x_d^{(t)})} \right)}{\left(\frac{f_h(x_h^{(t)} \mid x_1^{(t+1)}, \ldots, x_{h-1}^{(t+1)}, x_{h+1}^{(t)}, \ldots, x_d^{(t)})}{q_h(x_h^{(t)} \mid x_1^{(t+1)}, \ldots, x_{h-1}^{(t+1)}, Y_h, x_{h+1}^{(t)} \ldots, x_d^{(t)})} \right)}, 1 \right\};$$

     otherwise take $X_h^{(t+1)} = x_h^{(t)}$

- Note: the Metropolis-Hastings step is performed only *once* per iteration

Example: Metropolis-within-Gibbs (Continued)

- Suppose we want to sample from the target

$$f(x_1, x_2) \propto |\sin(\sqrt{x_1 x_2})|, \quad x_1 \in (0, 3), x_2 \in (0, 5)$$

- To implement Metropolis-within-Gibbs, we can use a random-walk Metropolis step to sample from each conditional distribution

Example: Metropolis-within-Gibbs (Continued)

```
set.seed(2311)

f <- function(x) {
  ifelse(((x[1] > 0) && (x[1] < 3) && (x[2] > 0) && (x[2] < 5)),
    abs(sin(sqrt(x[1]*x[2]))), 0)
}

TT <- 10000
X <- matrix(0L, nrow=2, ncol=TT)
X[,1] <- c(runif(n=1,0,3), runif(n=1,0,5))

for (t in 2:(TT-1)) {
  Y <- X[,t-1]
  for (h in 1:2) {
    Y[h] <- X[h,t-1] + rnorm(n=1)
    if (runif(n=1) < f(Y)/f(X[,t-1])) {
      X[h,t] <- Y[h]
    } else {
      X[h,t] <- X[h,t-1]
    }
  }
}
```

References I

Bryan D He, Christopher M De Sa, Ioannis Mitliagkas, and Christopher Ré. Scan order in gibbs sampling: Models in which it matters and bounds on how much. *Advances in neural information processing systems*, 29, 2016.

Jun S Liu, Faming Liang, and Wing Hung Wong. The multiple-try method and local optimization in metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134, 2000.