Variations of Gibbs sampler

Recall the basic algorithm of Gibbs sampler:

1. Initialize $x = x^1$ and set $k = 1$.
2. Update $x^k \to x^{k+1}$:
   - Draw x_1^{k+1} from $t \mapsto \pi(t, x_2^k, x_3^k, \cdots, x_n^k)$,
   - Draw x_2^{k+1} from $t \mapsto \pi(x_1^{k+1}, t, x_3^k, \cdots, x_n^k)$,
   - $\vdots$
   - Draw x_n^{k+1} from $t \mapsto \pi(x_1^{k+1} x_2^{k+1}, \cdots, x_{n-1}^{k+1}, t)$.
3. Increase $k \to k + 1$ and repeat from 2. until a desired sample size is reached.

Some observations:

- The *full scan* Gibbs sampler that updates all components may be slow.
- Sometimes, the Gibbs sampler moves with reasonable steps only because *some* of the components move, but not all of them.
- A partially updated point,

$$x_1^{k+1}, x_2^{k+1}, \ldots, x_j^{k+1}, x_j^k, \ldots, x_n^k$$

  may be informative from the point of view of exploring the density.

Random scan Gibbs sampler

The simplest for updates only one, randomly chosen component, at a time.

1. Initialize $x = x^1 \in \mathbb{R}^n$ and set $k = 1$.
2. Update $x^k \to x^{k+1}$:
   - Draw j, $1 \le j \le n$, from uniform distribution over $\{1, 2, \ldots, n\}$.
   - Draw t from $t \mapsto \pi(x_1^k, x_2^k, \ldots, x_{j-1}^k, t, x_{j+1}^k, \cdots, x_n^k)$,
   - Update
   $$x^{k+1} = [x_1^k, x_2^k, \ldots, x_{j-1}^k, t, x_{j+1}^k, \cdots, x_n^k]^{\mathrm{T}}.$$
3. Increase $k \to k + 1$ and repeat from 2. until a desired sample size is reached.

Comments:

- If n is large, the sampler may be *much* faster than the full scan Gibbs.
- Looking at single component sample histories, the "fuzzy worm" may look like one obtained with Metropolis-Hastings having a low acceptance rate: long waiting periods before the component is updated.
- Gives relatively quickly an idea of the density.
- Convergence (variance reduction) may be slow.

Variation

Instead of updating only one component, we may update few components:

- Draw $j_1, j_2, \ldots, j_k$, $1 \le j_1 < j_2 < \cdots j_k \le n$, from uniform distribution over $\{1, 2, \ldots, n\}$.
- With $x = x^k$, draw t from $t \mapsto \pi(x_1, x_2, \ldots, x_{j_1-1}, t_1, x_{j_1+1}, \cdots, x_n)$, and set $x_{j_1} = t$,
- $\vdots$
- Draw t from $t \mapsto \pi(x_1, x_2, \ldots, x_{j_k-1}, t_1, x_{j_k+1}, \cdots, x_n)$, and set $x_{j_k} = t$, and set $x^{k+1} = x$.

With $k = n$, this algorithm coincides with full scan Gibbs sampler.

Single component Metropolis-Hastings

The one-dimensional updatings in Gibbs sampling are done with the Golden Rule that requires a large number of evaluations of

$$t \mapsto \pi(x + te_k).$$

The one-dimensional updating in Gibbs sampler may be slow if the forward map $x \mapsto \pi(x)$ is computationally demanding.

A way to avoid multiple evaluations: use a simple proposal distribution.

Random scan single component Metropolis-Hastings:

1. Initialize $x = x^1 \in \mathbb{R}^n$ and set $k = 1$.
2. Update $x^k \to x^{k+1}$:
   - Draw j, $1 \leq j \leq n$, from uniform distribution over $\{1, 2, \ldots, n\}$.
   - Draw t from a proposal distribution $t \mapsto q_j(t, x_j^k)$, e.g., one-dimensional Gaussian. Set
   $$y = [x_1^k, x_2^k, \ldots, x_{j-1}^k, t, x_{j+1}^k, \cdots, x_n^k]^{\mathrm{T}}.$$
   - Check acceptance: if
   $$\alpha = \frac{\pi(y) q_j(x_j^k, t)}{\pi(x) q_j(t, x_j^k)} > \xi \sim \text{Uniform}([0, 1]),$$
   set $x^{k+1} = y$, else $x^{k+1} = x^k$.
3. Increase $k \to k + 1$ and repeat from 2. until a desired sample size is reached.

Comments:

- The algorithm is fast
- The one-dimensional proposals are relatively simple to tune so that the acceptance rate is decent. Readjustments may be needed while the sample moves on.
- Easy to equip with adaptive step size algorithm.
- This algorithm (or variants of it) has found to be very useful in high-dimensional problems: dimensionality up from hundreds.
- Most of the problems of random scan Gibbs sampler present.

More variations:

Gibbs sampler depends on the chosen coordinate system. The sampler is inefficient at least in the following cases:

- The density is thin and long in a direction not parallel to the coordinate axes,
- The density is multimodal, and Gibbs sampler may be unable to jump from one mode to another, if there is no corridor in the support of the density that can be traversed with moves parallel to the coordinate directions.

We have seen that SVD and adaptation may help. But could one play with the proposal direction?

Hit and run – algorithm

Design an algorithm along the following lines:

- Hit: Choose a random direction from the present sample point,
- Run: Update the sample point with a random move to the chosen direction.

In principle, we have two different choices for the updating:

1. Gibbs type update: use Golden Rule and accept automatically, or
2. Metropolis-Hastings type update: draw from proposal distribution and check for acceptance.

Hit and Run Gibbs form

1. Initialize $x = x^1 \in \mathbb{R}^n$ and set $k = 1$.
2. Update $x^k \to x^{k+1}$:
   - Draw a random direction $e \in \mathbb{R}^n$, $\|e\| = 1$ from uniform distribution over all directions.
   - Draw t from $t \mapsto \pi(x^k + te)$,
   - Update
   $$x^{k+1} = x^k + te.$$
3. Increase $k \to k + 1$ and repeat from 2. until a desired sample size is reached.

Hit and Run MH form

1. Initialize $x = x^1 \in \mathbb{R}^n$ and set $k = 1$.
2. Update $x^k \to x^{k+1}$:
   - Draw a random direction $e \in \mathbb{R}^n$, $\|e\| = 1$ from uniform distribution over all directions.
   - Draw t from a proposal distribution $t \mapsto q(t)$.
   - Check for acceptance: if

$$\alpha = \frac{\pi(x^k + te)}{\pi(x^k)} > \xi \sim \text{Uniform}([0,1]),$$

   set $x^{k+1} = x^k + te$, else $x^{k+1} = x^k$.
3. Increase $k \to k+1$ and repeat from 2. until a desired sample size is reached.

Note: the Metropolis-Hastings algorithm described above is a random walk model:

Proposal

$$x_{\text{prop}} = x_{\text{curr}} + te,$$

so

$$t = e^{\mathrm{T}}(x_{\text{prop}} - x_{\text{curr}}),$$

so the transition kernel depends only on the difference.

More complicated proposals are possible, i.e., proposal kernel may be of the form

$$t \mapsto q(t, x_{\text{curr}}), \quad t = e^{\mathrm{T}} x_{\text{prop}}$$

with the update

$$x_{\text{curr}} \to x_{\text{curr}}^{\perp} + \big(e^{\mathrm{T}} x_{\text{prop}} - e^{\mathrm{T}} x_{\text{curr}}\big)e,$$

where $x_{\text{curr}}^{\perp}$ is the component of x_{curr} orthogonal to e.

Drawing the direction

In practice, the random direction $e \in \mathbb{R}^n$ is drawn as follows:

$$\text{Draw } w \sim \mathcal{N}(0, I), \text{ and set } e = \frac{w}{\|w\|}.$$

Intuitively, this is clear: white noise has no preferred directions, so e has uniform distribution.

More precisely:

$$\begin{aligned} \mathrm{P}\{e \in B \subset \mathbb{S}^{n-1}\} &= \mathrm{P}\{w \in \mathbb{R} \times B\} \\ &= \left(\frac{1}{2\pi}\right)^{n/2} \int_0^\infty r^{n-1} e^{-r^2/2} \int_B de \\ &= \frac{1}{|\mathbb{S}^{n-1}|} \int_B de. \end{aligned}$$

Example: Gaussian with bounds

Implementations of Hit and Run (HR), Gibbs form, for the standard form problem,

$$\pi(\mathbf{x}) \propto \exp\left(-\frac{1}{2}\|A\mathbf{x}-\mathbf{b}\|^2\right), \quad C\mathbf{x} \geq \mathbf{r}.$$

(For clarity, vectors denoted by **boldface**).

The algorithm consists of two steps:

1. Draw a random direction vector $\mathbf{e} \in \mathbb{R}^n$, $\|\mathbf{e}\| = 1$,
2. Sample a scalar t from the density $\pi(\mathbf{x}+t\mathbf{e})$ and update $\mathbf{x} \longrightarrow \mathbf{x}+t\mathbf{e}$.

Algorithm for sampling the scalar t when the direction $\mathbf{e}$ is given:

We have

$$A(\mathbf{x}+t\mathbf{e})-\mathbf{b}=(A\mathbf{e})t-\widetilde{\mathbf{b}},$$

where

$$\widetilde{\mathbf{b}}=\mathbf{b}-A\mathbf{x}.$$

Now,

$$\begin{aligned}
\|A(\mathbf{x}+t\mathbf{e})-\mathbf{b}\|^2 &= \|(A\mathbf{e})t-\widetilde{\mathbf{b}}\|^2 \\
&= \|A\mathbf{e}\|^2t^2-2t(A\mathbf{e})^{\mathrm{T}}\widetilde{\mathbf{b}}+\|\widetilde{\mathbf{b}}\|^2 \\
&= \left(\underbrace{\|A\mathbf{e}\|t}_{=s}-\underbrace{\frac{(A\mathbf{e})^{\mathrm{T}}\widetilde{\mathbf{b}}}{\|A\mathbf{e}\|}}_{=\overline{s}}\right)^2+\underbrace{\|\widetilde{\mathbf{b}}\|^2-\frac{(A\mathbf{e})^{\mathrm{T}}\widetilde{\mathbf{b}})^2}{\|A\mathbf{e}\|^2}}_{=\text{constant}}.
\end{aligned}$$

Denote

$$s = \|A\mathbf{e}\|t, \quad \overline{s} = \frac{(A\mathbf{e})^{\mathrm{T}}\widetilde{\mathbf{b}}}{\|A\mathbf{e}\|}.$$

Density to be used for sampling s is

$$\pi(s) \propto \exp\left(-\frac{1}{2}(s-\overline{s})^2\right) \quad + \text{ bounds.}$$

The bounds for s, assuming that $\mathbf{x}$ is given:

$$C(\mathbf{x} + t\mathbf{e}) \geq \mathbf{r}$$

implies

$$t(C\mathbf{e}) \geq \mathbf{r} - C\mathbf{x}.$$

Denoting

$$\mathbf{v} = C\mathbf{e},$$

and by scaling with $\|A\mathbf{e}\|$, we have

$$s\mathbf{v} \geq \mathbf{q}, \quad \mathbf{q} = \|A\mathbf{e}\|\big(\mathbf{r} - C\mathbf{x}\big). \tag{1}$$

Make a permutation of the components v_i of $\mathbf{v}$ and $\mathbf{q}$ so that the v_is are in decreasing order. Assume that the ℓ first elements are positive,

$$v_1 \geq \cdots \geq v_\ell > 0,$$

while the entries starting from the $k+1$, $k \geq \ell$ are negative,

$$0 > v_{k+1} \geq \cdots \geq v_n.$$

Writing the inequality (1) component by component and taking the signs into account, we obtain

$$v_i s \geq q_i, \text{ or } s \geq \frac{q_i}{v_i}, \quad 1 \leq i \leq \ell,$$

and

$$v_i s \geq q_i, \text{ or } s \leq \frac{q_i}{v_i}, \quad k+1 \leq i \leq n.$$

In addition, one should check that the inequalities corresponding to zero entries are valid, that is,

$$0 \geq q_i, \quad \ell+1 \leq i \leq k.$$

Lower and upper bounds for s,

$$s_{\min} = \max_{1 \le i \le \ell} \left(\frac{q_i}{v_i} \right), \quad s_{\max} = \min_{k+1 \le i \le n} \left(\frac{q_i}{v_i} \right).$$

The conditional probability density of s is

$$\pi(s) \propto \exp\left(-\frac{1}{2}(s - \overline{s})^2 \right), \quad s_{\min} \le s \le s_{\max},$$

and the random draw has to be done from this density.

To do the draws properly, we have to consider three possibilities, each one treated below separately.

1. $\overline{s} > s_{\max}$. This means that we have to draw from the left tail of the Gaussian distribution.

2. $\overline{s} < s_{\min}$. This time, we need to draw from the right tail of the distribution.

3. $s_{\min} \leq \overline{s} \leq s_{\max}$, the maximum thus being within the interval.

1. $\overline{s} > s_{\max}$. The maximum value of this tail is achieved at $s = s_{\max}$. We scale the density so that this maximum value equals one:

$$\tilde{\pi}(s) = \exp\left(-\frac{1}{2}(s-\overline{s})^2 + p\right), \quad p = \frac{1}{2}(s_{\max}-\overline{s})^2.$$

Find interval where this density is bigger than a threshold value $\delta > 0$. Write

$$\tilde{\pi}(s) = \delta,$$

take logarithm of both sides to obtain

$$\frac{1}{2}(s-\overline{s})^2 - p = \log\frac{1}{\delta},$$

and solve for s, bearing in mind that $s < \overline{s}$,

$$s = s_* = \overline{s} - \left(2p + 2\log\frac{1}{\delta}\right)^{1/2}.$$

Hence, the effective draw interval is

$$\max(s_{\min}, s_*) \le s \le s_{\max}.$$

2. $\overline{s} < s_{\min}$. This time, we need to draw from the right tail of the distribution. The maximum is attained at $s = s_{\min}$, and the scaled density now is

$$\tilde{\pi}(s) = \exp\left(-\frac{1}{2}(s-\overline{s})^2 + p\right), \quad p = \frac{1}{2}(s_{\min} - \overline{s})^2.$$

Again, we seek the effective interval, that in this time is

$$s_{\min} \le s \le \min(s_*, s_{\max}),$$

where

$$s_* = \overline{s} + \left(2p + 2\log\frac{1}{\delta}\right)^{1/2}.$$

3. $s_{\min} \leq \overline{s} \leq s_{\max}$, the maximum thus being within the interval. In this case, the scaled density is directly

$$\tilde{\pi}(s) = \exp\left(-\frac{1}{2}(s-\overline{s})^2\right),$$

and solving the equation

$$\tilde{\pi}(s) = \delta$$

leads to the solutions

$$s = s_{*\pm} = \overline{s} \pm \left(2\log\frac{1}{\delta}\right)^{1/2},$$

so the active interval for s in this case is

$$\max(s_{\min}, s_{*,-}) \leq s \leq \min(s_{\max}, s_{*,+}).$$

Assume now that we have updated the interval to be the effective interval. The random draws according to Golden Rule algorithm:

Divide the interval in N subintervals, and denote

$$s_{\min} = \sigma_0 < \sigma_1 < \cdots < \sigma_N = s_{\max}.$$

Evaluate the scaled probability density at the division points,

$$p_j = \tilde{\pi}(\sigma_j), \quad 0 \le j \le N.$$

Then, define the non-scaled discretized approximation of the integral

$$\int_{-\infty}^{\sigma_k} \widetilde{\pi}(s)ds \approx h \sum_{j=1}^{k} p_j = \Phi_k, \quad 1 \le j \le N,\ h = \frac{s_{\max} - s_{\min}}{N}.$$

The sampling is then done by the following steps:

1. Draw $t \sim \text{Uniform}([0, \Phi_N])$,
2. Find
$$i = \min_{1 \le k \le N} \{k \mid \Phi_k > t\}.$$
3. set $s = \sigma_{i-1}$.

Example

Comparing Full Scan Gibbs and Hit and Run.

Gaussian density

$$\pi(x) \propto \exp\left(-\frac{1}{2\sigma^2}\|Ax-b\|^2\right),$$

where

$$A = DU,$$

$$D = \begin{bmatrix} 1 & \\ & 0.01 \end{bmatrix}, \quad U = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

and $\sigma = 0.01$.

Test for convergence:

Plot the relative number of sample points within the ellipse of $p\%$ probability.

$$W = \frac{1}{\sigma}(AX - b) \sim \mathcal{N}(0, 1),$$

so x_j within an ellipse of probability $p\%$ if

$$\|w_j\| = \frac{1}{\sigma}\|(Ax_j - b)\| < \alpha(p) = \sqrt{2 \log\left(\frac{100}{100 - p}\right)}.$$

Sample size 150 000.

Algorithm

This version is with bound constraints

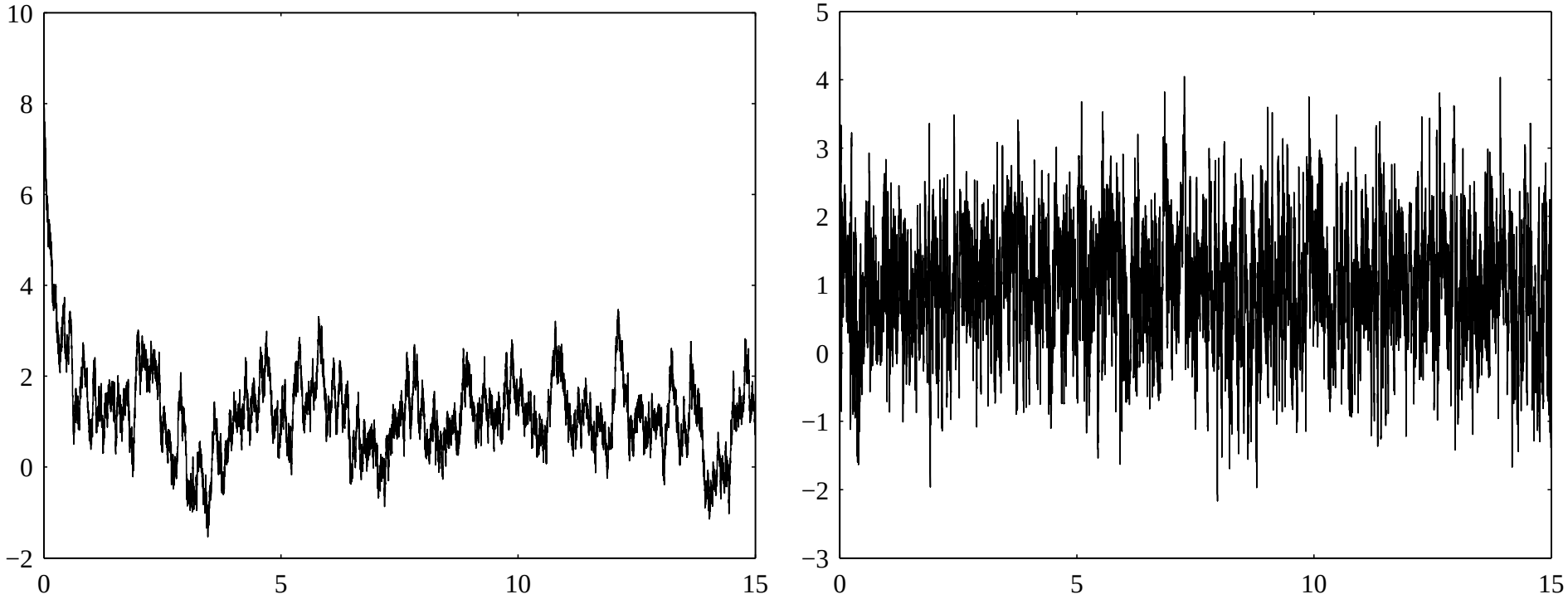
```
 large = 1000;
 X = zeros(ny,nsample);
 x = x_init;
 X(:,1) = x;


for n = 2:nsample
    % Hit: draw the direction
    e = randn(ny,1);
    e = (1/norm(e))*e;
```

```
% Run: draw the step length
bb = b - A*x;
Ae = A*e;
normAe = norm(Ae);
sbar = Ae'*bb/normAe;
v = C*e;
q = normAe*(r - C*x);
Iplus = find(v>tiny);
Iminus = find(v<-tiny);
smin = max([-large;q(Iplus)./v(Iplus)]);
smax = min([large;q(Iminus)./v(Iminus)]);
if smax - smin < tiny
   % the effective interval is reduced
   t = smin/normAe;
else
   % The interval is reasonable; draw from the density
   if sbar > smax
      % draw from the left tail
```

```
    p = 0.5*(smax - sbar)^2;
    sstar = sbar - sqrt(2*(p+log(1/cutoff)));
    smin = max(smin,sstar);
elseif sbar < smin
    % draw from the right tail
    p = 0.5*(smin - sbar)^2;
    sstar = sbar + sqrt(2*(p+log(1/cutoff)));
    smax = min(smax,sstar);
else
    % maximum is between the bounds
    p = 0;
    sstarp = sbar + sqrt(2*log(1/cutoff));
    sstarm = sbar - sqrt(2*log(1/cutoff));
    smax = min(smax,sstarp);
    smin = max(smin,sstarm);
end
s = linspace(smin,smax,bin);
pdf = exp(-0.5*(s-sbar).^2 + p);
```

```
        phi = cumsum(pdf);
        xi = phi(bin)*rand;
        jj = min(find(phi>xi));
        t = s(jj)/normAe;
    end
    x = x + t*e;
    X(:,n) = x;
end
```

Sample histories of the first component



Full scan Gibbs (left) and Hit and Run (right). Notice the difference with the burn-in.

Convergence to probability ellipses