Markov Chain Monte Carlo

Timothy Hanson

Department of Statistics, University of South Carolina

Stat 740: Statistical Computing

R packages

- There are *many* R packages to fit Bayesian models using MCMC with “built in” algorithms! Check to see if the model you want to fit is in a package. e.g. `DPpackage`, `spBayesSurv`, `ICBayes`, `MCMCglmm`, etc.
- `https://cran.r-project.org/web/views/Bayesian.html`

Independence sampling

- It's hard to find good independence M-H proposals.
- Lots of recent research on building independence proposals, e.g. `PTsampler` in `DPpackage`.
- For simple models we can use $\boldsymbol{\theta}^* \sim N_k(\hat{\boldsymbol{\theta}}, c[-\nabla^2 \log \pi(\hat{\boldsymbol{\theta}}|\mathbf{x})]^{-1})$ ($c \geq 1$) as an independence proposal; accept with probability

$$1 \wedge \frac{\pi(\boldsymbol{\theta}^*|\mathbf{x})\phi_k(\boldsymbol{\theta}^t|\hat{\boldsymbol{\theta}}, [-\nabla^2 \log \pi(\hat{\boldsymbol{\theta}}|\mathbf{x})]^{-1})}{\pi(\boldsymbol{\theta}^t|\mathbf{x})\phi_k(\boldsymbol{\theta}^*|\hat{\boldsymbol{\theta}}, [-\nabla^2 \log \pi(\hat{\boldsymbol{\theta}}|\mathbf{x})]^{-1})},$$

  where $a \wedge b$ is the smaller of a and b.
- Why $c \geq 1$?
- The "acceptance rate" is the proportion of accepted proposals.
- For independence proposals we want an acceptance rate as close to 1 as possible.

Example: Logistic regression on Challenger data.

Random-walk sampling

- Similarly, for random walk M-H we can use $\boldsymbol{\theta}^* \sim N_k(\boldsymbol{\theta}^t, c[-\nabla^2 \log \pi(\hat{\boldsymbol{\theta}}|\mathbf{x})]^{-1})$ for some c and try to get an acceptance rate between 20% and 50%. Accept with probability
$$1 \wedge \frac{\pi(\boldsymbol{\theta}^*|\mathbf{x})}{\pi(\boldsymbol{\theta}^t|\mathbf{x})},$$
- Choosing c smaller leads to larger acceptance rates, c bigger gives smaller acceptance rates. Need to "tune" proposal to get good acceptance rate & make algorithm efficient.
- Acceptance rate too high $\Rightarrow$ taking "small steps" in a local area of the posterior. Acceptance rate too low $\Rightarrow$ taking "giant leaps" away from area of high posterior mass into the tails each time.
- Adaptive random-walk M-H very useful; no need to "manually tune" proposal variance to get desired acceptance rate (later).

Example: Logistic regression on Challenger data.

General comments on MCMC

- For most models with many parameters, we write out the full joint distribution, simply “pull out” the pieces that involve each parameter to obtain that parameter’s unnormalized full conditional pdf/pmf, and perform componentwise updating.
- If all full conditionals are closed-form we have Gibbs sampling, otherwise it’s a hybrid sampler.
- JAGS, OpenBUGS are component-at-time samplers. Often use slice sampling, random-walk M-H with “pilot adaption,” and conditional conjugacy (when present).
- `MfUSampler` is an R package that does a component-at-a-time via ARMS, slice sampling and M-H w/ Gaussian proposal.
- Two examples where conditional conjugacy happens are the one-sample normal model and a finite mixture of normals.

Pros and cons of component-at-a-time

Pros:

- Often easy to program; "recognizable" full conditional distributions easy to sample.
- Automated in JAGS, OpenBUGS, etc.
- Always "works" but may be prohibitively slow.

Cons:

- Introduces potentially *a lot* of autocorrelation in the iterates (e.g. on board). A well thought-out M-H correlated proposal distribution can dramatically outperform component-at-a-time updates.

Normal data

$$\textbf{Model: } x_1, \ldots, x_n | \mu, \tau \overset{iid}{\sim} N(\mu, \tau^{-1}),$$

$$\textbf{Prior: } \mu \sim N(m, p^{-1}) \text{ ind. } \tau \sim \Gamma(a, b).$$

$$L(\mu, \tau | \mathbf{x}) \propto \prod_{i=1}^{n} \tau^{1/2} \exp\{-\tfrac{\tau}{2}(x_i - \mu)^2\} = \tau^{n/2} \exp\left\{-\frac{\tau}{2}\sum_{i=1}^{n}(x_i - \mu)^2\right\}.$$

Thus the posterior density is proportional to

$$\pi(\mu, \tau | \mathbf{x}) \propto \tau^{n/2} \exp\left\{-\frac{\tau}{2}\sum_{i=1}^{n}(x_i - \mu)^2\right\} \exp\{-\tfrac{p}{2}(\mu - m)^2\} \tau^{a-1} \exp\{-b\tau\}.$$

Recall:

$$\phi_d(\mathbf{x} | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)\phi_d(\mathbf{x} | \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \propto \phi_d(\mathbf{x} | \mathbf{V}[\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2], \mathbf{V}),$$

$$\text{where } \mathbf{V} = [\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}]^{-1}.$$

Normal data

Let $s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$ be the MLE for $\sigma^2 = \tau^{-1}$. Note that

$$\sum_{i=1}^{n}(x_i - \mu)^2 = \sum_{i=1}^{n}(x_i - \bar{x})^2 + n(\mu - \bar{x})^2 = n[s^2 + (\mu - \bar{x})^2].$$

Thus

$$\pi(\mu|\tau, \mathbf{x}) \propto \phi(\mu|\bar{x}, [n\tau]^{-1})\phi(\mu|m, p^{-1}),$$
$$\pi(\tau|\mu, \mathbf{x}) \propto \tau^{n/2+a-1}\exp\{-\tfrac{n}{2}[s^2 + (\mu - \bar{x})^2]\tau - b\tau\}.$$

Leading to

$$\mu|\tau, \mathbf{x} \sim N\left(\frac{n\tau\bar{x} + pm}{n\tau + p}, \frac{1}{n\tau + p}\right),$$
$$\tau|\mu, \mathbf{x} \sim \Gamma(a + \tfrac{n}{2}, b + \tfrac{n}{2}[s^2 + (\mu - \bar{x})^2]).$$

We are in good shape for Gibbs sampling!

Example: normal data hand-coded Gibbs sampler.

OpenBUGS & JAGS

`R2OpenBUGS` and `BRugs` interface R to OpenBUGS. `R2jags` interfaces R to JAGS. The syntax for OpenBUGS and JAGS are very similar, but but not exactly the same. BUGS is "Baysian inference Using Gibbs Sampling and JAGS is Just Another Gibbs Sampler.

Download JAGS from
`https://sourceforge.net/projects/mcmc-jags/files/latest/download`
and install it. Then start R and install the R package `R2jags`.

JAGS mainly uses slice sampling and the inverse cdf method for updating individual parameters.

Example: normal data via JAGS.

Finite mixture models: unsupervised learning

Recall model-based clustering:

$$X_1, \ldots, X_n | \boldsymbol{\theta} \overset{iid}{\sim} f(x) = \sum_{j=1}^{J} \pi_j \phi(x|\mu_j, \sigma_j^2).$$

Parameters are
$\boldsymbol{\theta} = (\pi_1, \ldots, \pi_{J-1}, \mu_1, \ldots, \mu_J, \sigma_1^2, \ldots, \sigma_J^2)' \in \mathbb{R}^{3J-1}$. Direct maximization of

$$L(\boldsymbol{\theta}|\mathbf{x}) = \prod_{j=1}^{n} \sum_{j=1}^{J} \pi_j \phi(x_i|\mu_j, \sigma_j^2)$$

is *very challenging*.

E-M offered a solution. How about Bayes?

Component membership

Recall method of composition $X_i|\boldsymbol{\theta}, z_i \sim N(\mu_{z_i}, \sigma^2_{z_i})$ conditionally, and $p(j|\boldsymbol{\theta}) = P(z_i = j|\boldsymbol{\theta}) = \pi_j$ marginally, gives same distribution $f(x)$ on previous slide. Add "missing" $\mathbf{z} = (z_1, \ldots, z_n)'$ to the model to get

$$\begin{aligned} L(\boldsymbol{\theta}|\mathbf{x}, \mathbf{z}) &= f(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) = f(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) f(\mathbf{z}|\boldsymbol{\theta}) \\ &= \left[\prod_{i=1}^n \phi(x_i|\mu_{z_i}, \sigma^2_{z_i})\right] \left[\prod_{i=1}^n p(z_i|\boldsymbol{\theta})\right] \\ &= \prod_{i=1}^n \phi(x_i|\mu_{z_i}, \sigma^2_{z_i}) \pi_{z_i} \end{aligned}$$

Bayesian analysis requires prior distributions, perhaps

$$(\pi_1, \ldots, \pi_J)' \sim \text{Dirichlet}(w, \ldots, w),$$

$$\mu_1, \ldots, \mu_J \overset{iid}{\sim} N(m, p^{-1}),$$

$$\tau_1, \ldots, \tau_J \overset{iid}{\sim} \Gamma(a, b).$$

Dirichlet distribution & posterior

Prior on π such that $\pi_1 + \cdots + \pi_J = 1$. Generalizes the beta. Using π for two different things, the density is $\pi(\boldsymbol{\pi}) = \frac{\Gamma(Jw)}{\Gamma(w)^J} \prod_{j=1}^J \pi_j^{w-1}$.

$$
\begin{aligned}
\pi(\boldsymbol{\mu}, \boldsymbol{\tau}, \boldsymbol{\pi}, \mathbf{z}|\mathbf{x}) &= \tfrac{\pi(\boldsymbol{\mu}, \boldsymbol{\tau}, \boldsymbol{\pi}, \mathbf{z}|\mathbf{x})}{f(\mathbf{x})} \\
&\propto f(\mathbf{x}|\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\tau}, \boldsymbol{\pi}) p(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\tau}, \boldsymbol{\pi}) \pi(\boldsymbol{\mu}, \boldsymbol{\tau}, \boldsymbol{\pi}) \\
&= L(\boldsymbol{\mu}, \boldsymbol{\tau}, \boldsymbol{\pi}|\mathbf{x}, \mathbf{z}) p(\mathbf{z}|\boldsymbol{\pi}) \pi(\boldsymbol{\mu}) \pi(\boldsymbol{\tau}) \pi(\boldsymbol{\pi}) \\
&= \prod_{i=1}^n \phi(x_i|\mu_{z_i}, \tau_{z_i}^{-1}) \prod_{i=1}^n \pi_{z_i} \\
&\quad \times \prod_{j=1}^J \phi(\mu_j|m, p^{-1}) \prod_{j=1}^J \frac{b^a}{\Gamma(a)} \tau_j^{a-1} e^{-b\tau_j} \\
&\quad \times \tfrac{\Gamma(Jw)}{\Gamma(w)^J} \prod_{j=1}^J \pi_j^{w-1}.
\end{aligned}
$$

Full conditional distributions

Just pick out parts of $\pi(\boldsymbol{\mu}, \boldsymbol{\tau}, \mathbf{w}, \mathbf{z}|\mathbf{x})$ on previous slide that involve the parameter of interest!

$$\pi(\mu_j|\text{else}) \propto \phi(\mu_j|m, p^{-1}) \prod_{i:z_i=j} \phi(x_i|\mu_j, \tau_j^{-1}),$$

$$\pi(\tau|\text{else}) \propto \tau_j^{a-1} e^{-b\tau_j} \prod_{i:z_i=j} \phi(x_i|\mu_j, \tau_j^{-1}),$$

$$\pi(\boldsymbol{\pi}|\text{else}) \propto \prod_{j=1}^{J} \pi_j^{\sum_{i=1}^{n} I\{z_i=j\}} \prod_{j=1}^{J} \pi_j^{w-1}.$$

$$P(z_i = j|\text{else}) \propto \phi(x_i|\mu_j, \tau_j^{-1})\pi_j,$$

All distributions have closed-form!

Let $n_j = \sum_{i=1}^{n} I\{z_i = j\}$ and $\bar{x}_j = \frac{1}{n_j} \sum_{i:z_i=j} x_i$.

$$\mu_j|\text{else} \sim N(\tfrac{n_j\tau\bar{x}_j + pm}{n_j\tau + p}, \tfrac{1}{n_j\tau + p}),$$

$$\tau_j|\text{else} \sim \Gamma\left(a + \tfrac{n_j}{2}, b + \tfrac{1}{2}\sum_{i:z_i=j}(x_i - \mu_j)^2\right),$$

$$\boldsymbol{\pi}|\text{else} \sim \text{Dirichlet}(w + n_1, \ldots, w + n_J).$$

$$P(z_i = j|\text{else}) = \frac{\phi(x_i|\mu_j, \tau_j^{-1})\pi_j}{\sum_{k=1}^{J} \phi(x_i|\mu_k, \tau_k^{-1})\pi_k},$$

Gibbs sampler straightforward to set up!

Example: Galaxy data hand-coded and in JAGS. Note: `BayesMix` automates this!

Tricking JAGS to fit general models

Say we have a model with independent likelihood contributions

$$L_i = f(y_i|\boldsymbol{\theta}),$$

where $\boldsymbol{\theta}$ are model parameters that can include random effects.

Define "data" $z_i = 0$ for $i = 1, \ldots, n$. Then

$$z_i \sim \text{Pois}\{-\log(L_i)\},$$

gives the correct likelihood! Simply have to define $-\log(L_i)$ for each datum; JAGS will (attempt) to do the rest.

What if we didn't augment likelihood w/ $\mathbf{z}$?

$$\pi(\boldsymbol{\theta}|\mathbf{x}) \propto \left[\prod_{j=1}^{n} \sum_{j=1}^{J} \pi_j \phi(x_i|\mu_j, \sigma_j^2)\right] \pi(\boldsymbol{\mu}, \boldsymbol{\tau}, \boldsymbol{\pi})$$

does not simplify! Can *attempt* a general M-H scheme (will utterly fail unless J is 2 or 3). JAGS performs componentwise updating via the "Poisson zeros trick."

Example: galaxy data using the zeros trick in JAGS.

Posterior inference for finite mixture

Inference centers on $\{(\boldsymbol{\mu}^m, \boldsymbol{\tau}^m, \boldsymbol{\pi}^m)\}_{m=1}^M$, we throw away the sampled $\mathbf{z}^m$ if we fit the model using latent data. We are using the method of composition to make updating easier!

An estimate of $f(x)$ is the posterior mean of the density $f(x|\boldsymbol{\theta})$

$$f(x|\mathbf{x}) = \int_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} f(x|\boldsymbol{\theta}) \underbrace{\pi(\boldsymbol{\theta}|\mathbf{x})}_{\text{MCMC}} \, d\boldsymbol{\theta}.$$

Using method of composition,

$$f(x|\mathbf{x}) \approx \tfrac{1}{M} \sum_{m=1}^{M} \underbrace{\sum_{j=1}^{J} \pi_j^m \phi(x|\boldsymbol{\mu}_j^m, \boldsymbol{\tau}_j^{-1m})}_{f(x|\boldsymbol{\theta}^m)}.$$

Ache hunting

Recall

$$y_i|\boldsymbol{\beta}, u_i \overset{ind.}{\sim} \text{Pois}\{t_i \exp(\beta_0 + \beta_1 a_i + u_i)\}, \quad u_1, \ldots, u_{47}|\tau \overset{iid}{\sim} N(0, \tau^{-1}).$$

Assume $\beta_0, \beta_1 \overset{iid}{\sim} N(0, 10^4)$ indep. of $\tau \sim \Gamma(10^{-4}, 10^{-4})$. These priors are proper but vague. Let $\boldsymbol{\theta} = (\beta_0, \beta_1, \tau)'$. The augmented likelihood is

$$L(\boldsymbol{\theta}|\mathbf{y}, \mathbf{u}) \propto \prod_{i=1}^{n} \exp\{-t_i e^{\beta_0 + \beta_1 a_i + u_i}\} e^{(\beta_0 + \beta_1 a_i + u_i) y_i}.$$

The augmented posterior is

$$\pi(\boldsymbol{\theta}, \mathbf{u}|\mathbf{y}) \propto L(\boldsymbol{\theta}|\mathbf{y}, \mathbf{u}) \left[\prod_{i=1}^{n} \tau^{1/2} e^{-\frac{\tau}{2} u_i^2}\right] \pi(\boldsymbol{\beta}) \pi(\tau).$$

MCMC component-at-a-time

The MCMC algorithm will cycle through sampling the full conditional distributions of $\beta_0, \beta_1, \tau, u_1, \ldots, u_{47}$ giving a final MCMC sample $\{(\beta_0^m, \beta_1^m, \tau^m, u_1^m, \ldots, u_{47}^m)_{m=1}^M$ from the posterior distribution $[\beta_0, \beta_1, \tau, u_1, \ldots, u_{47}|\mathbf{y}]$.

If we don't care about $u_1, \ldots, u_{47}$ we throw them away (method of composition) and are simply left with $[\beta_0, \beta_1, \tau|\mathbf{y}]$! Introducing $u_1, \ldots, u_{47}$ into the MCMC scheme simply makes it easier to update; no numerical integration required as when we used maximum likelihood.

Example: Ache hunting data.

Prediction

Prediction *is ridiculously easy* using MCMC. Say we want to predict how many kills y a new hunter aged a years will make on a t-day trek. This is given by

$$\begin{aligned}
p(y|\mathbf{y}) &= \int_{\mathbb{R}} \int_{\boldsymbol{\Theta}} p(y|u, \boldsymbol{\theta}, \mathbf{y}) \pi(u, \boldsymbol{\theta}|\mathbf{y}) du d\boldsymbol{\theta} \\
&= \int_{\mathbb{R}} \int_{\boldsymbol{\Theta}} p(y|u, \boldsymbol{\theta}) p(u|\boldsymbol{\theta}) \pi(\boldsymbol{\theta}|\mathbf{y}) du d\boldsymbol{\theta} \\
&= \int_{\mathbb{R}} \int_{\boldsymbol{\Theta}} \text{Pois}(y|te^{\beta_0+\beta_1 a+u}) \phi(u|0, \tau^{-1}) \underbrace{\pi(\boldsymbol{\theta}|\mathbf{y})}_{\text{MCMC}} du d\boldsymbol{\theta}
\end{aligned}$$

We can use the method of composition (again)!

$$u^m \sim N(0, \tau^{-1m}), \quad y^m \sim \text{Pois}(te^{\beta_0^m+\beta_1^m a+u^m}), \quad m = 1, \ldots, M.$$

Then $y^1, \ldots, y^M$ are a sample from the predictive number of kills over t days for a new age a hunter from the population age a years.

PROC MCMC and PROC NLMIXED in SAS

PROC NLMIXED fits very general models; like JAGS you can build a likelihood from “component parts” and fit general hierarchical nonlinear mixed effects models. PROC NLMIXED uses adaptive quadrature to compute the likelihood for models with random effects.

PROC MCMC works much like PROC NLMIXED but uses MCMC to fit the models instead. Can have much higher dimension for random effects vectors and fit more general models. Uses block update random-walk M-H. Like JAGS, adaption occurs during burn-in. However, the user gets to determine which “blocks” of parameters are updated together via random walk M-H.

PROC MCMC and PROC NLMIXED in SAS

Let's look at the Ache capuchin monkey hunting data again:

$$y_i \sim \text{Pois}\{t_i \exp(\beta_0 + \beta_1 a_i + \beta_2 a_i^2 + u_i)\}$$

where y_i is the number of monkeys killed on a trek that lasted t_i days for the ith hunter aged a_i years.

$$u_1, \ldots, u_{47} \overset{iid}{\sim} N(0, \sigma^2).$$

We will fit Bayesian and frequentist versions of this mixed model.

PROC NLMIXED computes the likelihood by integrating out the random effects $u_1, \ldots, u_{47}$ via adaptive Gaussian quadrature. *Inference is based on the large sample normal approximation* $\hat{\boldsymbol{\theta}} \sim N_4(\boldsymbol{\theta}, [-\nabla^2 \log L(\boldsymbol{\theta}|\mathbf{y})]^{-1})$.

PROC MCMC fits the model including sampling $u_1, \ldots, u_{47}$.

Adaptive M-H

Haario, Saksman, and Tamminen (2001) propose an ingenious solution to the random walk M-H "tuning dilemma."

They suggest using previous MCMC iterates to adjust a random-walk M-H proposal "on the fly" as the algorithm progresses. This works because the covariance matrix converges almost surely to the posterior covariance matrix for $\pi(\boldsymbol{\theta}|\mathbf{x})$.

That is, the chain "converges" to a random walk M-H scheme with a multiple of $cov(\boldsymbol{\theta}|\mathbf{x})$ as the proposal.

`adaptMCMC` performs adaptive M-H for arbitrary $\log \pi(\boldsymbol{\theta}|\mathbf{x})$. Need to supply an approximate starting covariance matrix for $\boldsymbol{\theta}|\mathbf{x}$, which can just be diagonal with estimated variances.

Adaptive M-H

The algorithm samples $\boldsymbol{\theta}^* \sim N_k(\boldsymbol{\theta}^j, c\mathbf{V}_j)$ and accepts $\boldsymbol{\theta}^{j+1} = \boldsymbol{\theta}^*$ with the usual probability

$$1 \wedge \frac{\pi(\boldsymbol{\theta}^*|\mathbf{x})}{\pi(\boldsymbol{\theta}^j|\mathbf{x})},$$

otherwise $\boldsymbol{\theta}^{j+1} = \boldsymbol{\theta}^j$. Haario et al. suggest $k = 2.4/k$. Here,

$$\mathbf{V}_j = \left\{ \begin{array}{cc} \mathbf{V}_0 & j \leq B \\ \frac{1}{j}\sum_{k=1}^{j}(\boldsymbol{\theta}^j - \bar{\boldsymbol{\theta}}_j)(\boldsymbol{\theta}^j - \bar{\boldsymbol{\theta}}_j)' & j > B \end{array} \right\}.$$

For a sequence $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \ldots$ a recursive formula for the mean is

$$\bar{\boldsymbol{\theta}}_j = \tfrac{1}{j}((j-1)\bar{\boldsymbol{\theta}}_{j-1} + \boldsymbol{\theta}_j).$$

A recursive formula for the variance is

$$\mathbf{V}_j = \tfrac{1}{j}[(j-1)(\mathbf{V}_{j-1} + \bar{\boldsymbol{\theta}}_{j-1}\bar{\boldsymbol{\theta}}'_{j-1}) + \boldsymbol{\theta}_j\boldsymbol{\theta}'_j] - \bar{\boldsymbol{\theta}}_j\bar{\boldsymbol{\theta}}'_j.$$

Example: Challenger data via adaptive M-H.

Componentwise adaption

Its is often better in terms of mixing to break up $\boldsymbol{\theta}$ into pieces; e.g. sometimes the full conditional of one (or more) piece is closed-form or a piece has elements that are related in some way (like regression coefficients).

One can do adaptive M-H on each piece separately. For example, say $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3)$ where $\boldsymbol{\theta}_j \in \mathbb{R}^{k_j}$ with $k_1 + k_2 + k_3 = k$. One can then sample

- $\boldsymbol{\theta}_1^* \sim N_{k_1}(\boldsymbol{\theta}_1^t, c_1 \mathbf{V}_{1t})$ and accept w/ probability ρ_{1t},
- $\boldsymbol{\theta}_2^* \sim N_{k_2}(\boldsymbol{\theta}_2^t, c_1 \mathbf{V}_{2t})$ and accept w/ probability ρ_{2t},
- $\boldsymbol{\theta}_3^* \sim N_{k_3}(\boldsymbol{\theta}_3^t, c_3 \mathbf{V}_{3t})$ and accept w/ probability ρ_{3t},

where the ρ_{tj} and V_{jt} are on the previous slide and $c_j = 2.4/k_j$.

See Haario, Saksman, and Tamminen (2005).

Adapting both c and $\mathbf{V}$

It is also possible to adapt the c parameter to achieve a desired acceptance rate, e.g. $c \to c_t$.

We will not delve into this further, but the `adaptMCMC` R package automates this for single block updates (all at once).

Example: Challenger data via `adaptMCMC`

Censored data

Censored data follow

$$t_1, \ldots, t_n \overset{iid}{\sim} f_{\boldsymbol{\theta}}(\cdot) \text{ indep. } c_1, \ldots, c_n \overset{iid}{\sim} h(\cdot).$$

We see $y_i = \min\{t_i, c_i\}$ and $\delta_i = I\{t_i < c_i\}$. The observed data is $\mathbf{x} = \{(y_i, \delta_i)\}_{i=1}^n$. Missing data are $\mathbf{z} = \{t_i : \delta_i = 0\}$.

One can try and sample the observed data posterior directly

$$\pi(\boldsymbol{\theta}|\mathbf{x}) \propto \pi(\boldsymbol{\theta}) \prod_{i=1}^n f_{\boldsymbol{\theta}}(y_i)^{\delta_i}[1 - F_{\boldsymbol{\theta}}(y_i)]^{1-\delta_i},$$

e.g. via M-H, adaptive M-H, or the zeros trick in JAGS.

Alternatively, as in E-M, one can introduce latent "missing data" to make updating simpler.

Censored data

Missing data are the true survival times t_i for $\delta_i = 0$. When $\delta_i = 0$ all we know is that $t_i \sim f_{\boldsymbol{\theta}}(\cdot)$ and $x_i > y_i$. For those i s.t. $\delta_i = 0$,

$$t_i | t_i > y_i, \boldsymbol{\theta}, \mathbf{x} \sim f_{\boldsymbol{\theta}}(\cdot) I\{t_i > y_i\}.$$

Simply sample t_i from $f_{\boldsymbol{\theta}}(\cdot)$ truncated to (y_i, ∞).

$$\pi(\boldsymbol{\theta}|\mathbf{x}, \mathbf{z}) \propto \pi(\boldsymbol{\theta}) \prod_{i=1}^{n} f_{\boldsymbol{\theta}}(t_i).$$

Simply alternate updating the censored t_i, then updating $\boldsymbol{\theta}$! Much, *much* easier than E-M!

JAGS can automate this, but it's a bit clumsy.

Right-censored normal data

$$\textbf{Model: } t_1, \ldots, t_n | \mu, \tau \overset{iid}{\sim} N(\mu, \tau^{-1}),$$

$$\textbf{Prior: } \mu \sim N(m, p^{-1}) \text{ ind. } \tau \sim \Gamma(a, b).$$

For all i s.t. $\delta_i = 0$,

$$t_i | t_i > y_i, \boldsymbol{\theta}, \mathbf{x} \sim N(\mu, \tau^{-1}) I\{t_i > y_i\},$$

is updated by taking $u_i \sim U\{\Phi(y_i|\mu, \tau^{-1}), 1\}$ and then $t_i = \mu + \tau^{-1/2}\Phi^{-1}(u_i)$. Then $\boldsymbol{\theta} = (\mu, \tau)$ is updated

$$\mu | \tau, \mathbf{t} \sim N\left(\frac{n\tau\bar{t} + pm}{n\tau + p}, \frac{1}{n\tau + p}\right),$$

$$\tau | \mu, \mathbf{t} \sim \Gamma(a + \tfrac{n}{2}, b + \tfrac{n}{2}[s^2 + (\mu - \bar{t})^2]).$$

where $s^2 = \frac{1}{n}\sum_{i=1}^{n}(t_i - \bar{t})^2$. Note that s^2 and $\bar{t}$ change every MCMC iteration along with the sampled $\{t_i : \delta_i = 0\}$.
Example: V.A. data by hand & JAGS.

Comments

- The slower the ACF dies down to zero, the more iterates you need!
- The effective sample size (ESS) is an estimate of how many *iid* samples you really have.
- Burn-in should range from 1000 to 10,000. However, if you know you are starting in a "reasonable" area (e.g. using the MLE) no burn-in is needed.
- I always take a mininum of 5,000 iterates after burnin, but much larger numbers cannot hurt, e.g. 10,000 or 50,000 or 100,000 or a million...
- I typically run one long chain and look at history (or trace) plots, cumulative percentiles, density estimates, and ACF plots.