Examples of Adaptive MCMC

Gareth O. Roberts and Jeffrey S. Rosenthal

Chelsea Lofland

AMS 280D

January 31, 2014

Introduction

- Metropolis algorithms need tuning of proposal covariance for efficient mixing
- This can be problematic in high dimensions
- Adaptive MCMC used to "learn" good parameter values automatically
- Results in better mixing properties and correct convergence

Conditions

Adaptive MCMC do not always preserve stationarity of the target distribution $\pi(\cdot)$

Assuming:

- **Diminishing Adaption condition** - two successive transition kernels are similar
- **Boundary Convergence condition** - ergodicity of transition kernels

these are satisfied:

- Asymptotic convergence
- Weak law of large numbers (WLLN)

Adaptive Metropolis (AM)

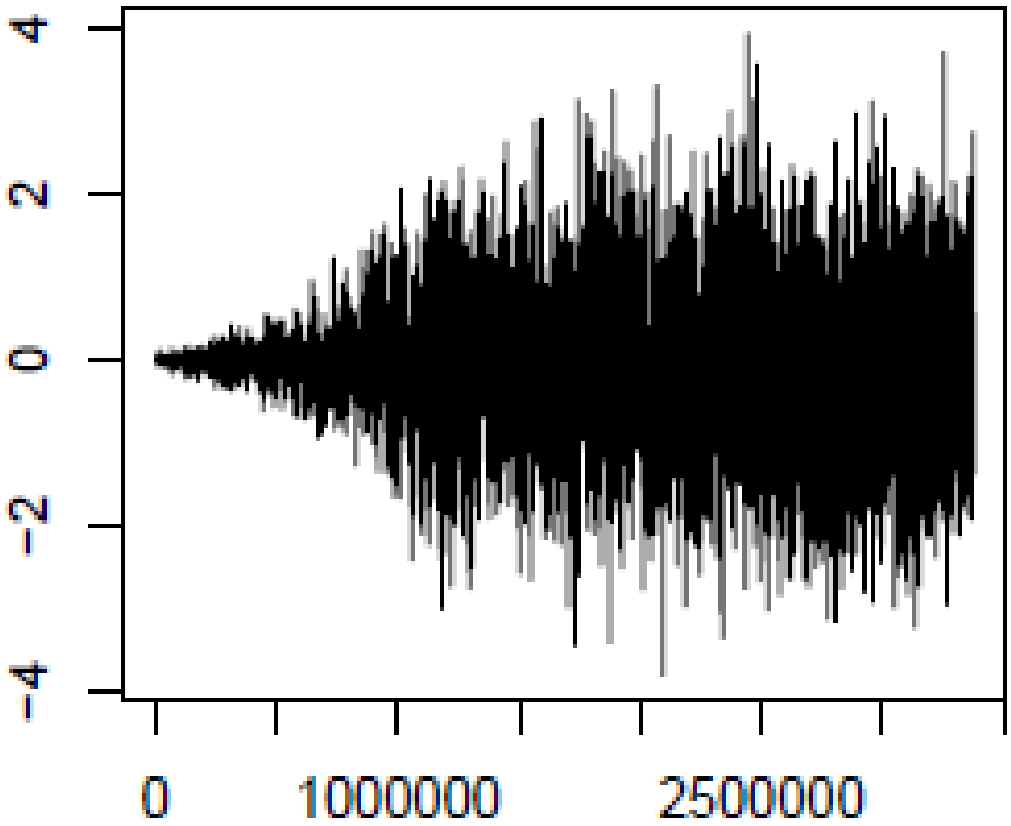
Target distribution $\pi(\cdot)$ d-dimensional.
At iteration n,

$$Q_n(x,\cdot) = \begin{cases} N(x, 0.1^2 I_d/d) & \text{if } n \leq 2d \\ (1-\beta)N(x, 2.38^2 \Sigma_n/d) + \beta N(x, 0.1^2 I_d/d) & \text{if } n > 2d \end{cases}$$

where
Σ_n is the current covariance matrix, used to estimate the optimal Σ
β is a small constant, > 0 to ensure Boundary Convergence

Adaptive Metropolis (AM)

Test with target $N(0, MM')$ where M is dxd and $M_{ij} \overset{iid}{\sim} N(0,1)$
The target covariance matrix is unpredictable and difficult for high dimension.
The trace (d=200) demonstrates the "learning" abilities of the algorithm.



Adaptive Metropolis (AM)

Can also measure how well the algorithm is performing with suboptimality factor

$$b = d \frac{\sum_{i=1}^{d} \lambda_i^{-2}}{(\sum_{i=1}^{d} \lambda_i^{-1})^2}$$

where λ_i are the eigenvalues of $\Sigma_p^{1/2} \Sigma^{-1/2}$

- This shows what factor the mixing rate will be slower by
- b is desired to be close to 1
- This is acheived after many iterations in the algorithm despite b starting at huge values:
  - d=100: 193.53 to 1.086
  - d=200: 183,000 to 1.04

Adaptive Metropolis-Within-Gibbs

Consider

$$\theta_i \sim Cauchy(\mu, A) \text{ for } 1 \leq i \leq K$$
$$Y_{ij} \sim N(\theta_i, V) \text{ for } 1 \leq j \leq r_i$$
$$p(\mu) = N(0, 1)$$
$$p(A) = p(V) = IG(1, 1)$$

This gives a target distribution on (K+3)-dimensional vector $(A, V, \mu, \theta_1, ..., \theta_K)$

- Let $K = 500$ and $5 \leq r_i \leq 500$

Adaptive Metropolis-Within-Gibbs

If the usual Metropolis-Within-Gibbs approach is taken, how should the proposal variance be chosen? Should it be different for each variable? This can be addressed with an adaptive algorithm!

Adaptive Metropolis-Within-Gibbs

- For each variable, create ls_i to represent the log of the SD
- Begin with unit variance ($ls_i = 0$)
- After n^{th} batch of 50 iterations, update ls_i by
  - adding $\delta(n)$ if acceptance rate above 0.44
  - subtracting $\delta(n)$ if acceptance rate below 0.44
- Choose $\delta(n) \to 0$ to satisfy Diminishing Adaption condition
- Let $\delta(n) = \min(0.01, n^{-1/2})$
- Restrict each ls_i to $[-M, M]$ for some $M < \infty$ to satisfy the Boundary Convergence condition, though this isn't needed in practice

Adaptive Metropolis-Within-Gibbs

Variable	r_i	Algorithm	ACT	Avg Sq Dist
θ_1	5	Adaptive	2.59	14.932
θ_1	5	Fixed	31.69	0.863
θ_2	50	Adaptive	2.72	1.508
θ_2	50	Fixed	7.33	0.581
θ_3	500	Adaptive	2.72	0.150
θ_3	500	Fixed	2.67	0.147

- Adaptive algorithm has much smaller autocorrelation times and larger average squared jumping distances than a fixed algorithm.
- θ_3 shows minimal comparison since it is already close to optimal

State-Dependent Scalings

Consider the case when the proposal variance depends on the current state, ex. $Q(x, \cdot) = N(x, \sigma_x^2)$

- Usual Metropolis-Hastings: a proposal from x to y is accepted with probability

$$\min\left(1, \frac{\pi(y)}{\pi(x)}(\sigma_x/\sigma_y)^d exp\{-\frac{1}{2}(x-y)^2(\sigma_y^{-2} - \sigma_x^{-2})\}\right)$$

State-Dependent Scalings

- Proposed algorithm:

$$Q_{a,b}(x,\cdot) = N\left(x, e^a\left(\frac{1+|x|}{exp(\hat{\pi})}\right)^b\right)$$

  where $\hat{\pi}$ is the current estimate of $\pi(log(1+|x|))$
- Update a and b by adding or subracting $\delta(n)$ to
  - again make acceptance rate as close to 0.44 as possible
  - make the acceptance rate also in the regions $\{x \in X : log(1+|x|) > \hat{\pi}\}$ and $\{x \in X : log(1+|x|) \leq \hat{\pi}\}$
- Again restrict a, b to $[-M, M]$

State-Dependent Scalings

- Algorithm shows convergence and good mixing
- Since $\delta(n) \to 0$ slowly, a and b oscillate
- Can again assess how well the algorithm is performing by comparing the autocorrelation time and average squared jumping distance to the non-adaptive algorithm
- Unclear of how to generalize to higher dimensions

State-Dependent Scalings

Algorithm	Acceptance Rate	ACT	Avg Sq Dist
Adaptive	0.456	2.63	0.769
$\sigma^2 = exp(-5)$	0.973	49.92	0.006
$\sigma^2 = exp(-1)$	0.813	8.95	0.234
$\sigma^2 = 1$	0.704	4.67	0.450
$\sigma^2 = 2.38^2$	0.445	2.68	0.748
$\sigma^2 = exp(5)$	0.237	7.22	0.305
$\sigma_x^2 = e^{1.5}(\frac{1+\lvert x\rvert}{0.534822})^{1.6}$	0.456	2.58	0.778

Regional Adaptive Metropolis Algorithm (RAMA)

- σ_x^2 are piecewise constant over various regions of the state space
- Partition state space X into finite number of disjoint regions.
- Proposal $Q(x, \cdot) = N(x, exp(2a_i))$ from x to y is accepted with probability

$$\min\left[1, \frac{\pi(y)}{\pi(x)} exp\Big\{ d(a_i - a_j) - \frac{1}{2}(x-y)^2[exp(-2a_j) - exp(-2a_i)]\Big\}\right]$$

Regional Adaptive Metropolis Algorithm (RAMA)

- To make the acceptance rate as close to 0.234 as possible in each region, after n^{th} batch of 100 iterations and for $1 \leq i \leq d$
  - Add $\delta(n)$ to a_i if acceptance rate above 0.234
  - Subtract $\delta(n)$ to a_i if acceptance rate below 0.234
- Then,
  - if $a_i > M$ then set $a_i = M$
  - if $a_i < -M$ then set $a_i = -M$
- $\delta(n) \to 0$ and $M < \infty$ again ensure the conditions are satisfied.

Regional Adaptive Metropolis Algorithm (RAMA)

Example: Let $X = R^d$ and $\pi(\cdot) = N(0, I_d)$

- $Q_{a,b}(x, \cdot) = N\left(x, e^{2a}\mathbf{1}_{\|x\|^2 \leq d} + e^{2b}\mathbf{1}_{\|x\|^2 > d}\right)$
- Restrict a, b to $[-M, M]$, and update every 100 iterations by
  - adding $\delta(n)$ to a to have the acceptance rate in $\{||x||^2 \leq d\}$ as close to 0.234 as possible
  - adding $\delta(n)$ to b to have the acceptance rate in $\{||x||^2 > d\}$ as close to 0.234 as possible
- Let $\delta(n) = \min(0.01, n^{-1/2}) = 0.01$, $M = 100$, $d = 10$, initial $a = b = 0$

Regional Adaptive Metropolis Algorithm (RAMA)

a,b	ACT	Avg Sq Dist
Adaptive	15.54	0.1246
-0.3, -0.13	15.07	0.1258
-0.3, 0.0	15.44	0.1213
0.0, -0.13	17.04	0.1118
0.0, 0.0	17.037	0.1100
-0.3, -0.3	16.01	0.1215

Remarks on RAMA

- $M < \infty$ condition was not found to be necessary - conjecture that Boundary Convergence will be satisfied under appropriate regularity conditions (ex. the densities are jointly continuous)
- Important to start with small values of a_i to avoid a small acceptance probability and therefore never exploring the region
- Regions currently user-defined but could be extended to be automatically selected by the computer
- Optimality of equal acceptance rate across regions is of concern at region boundaries.
- If $\delta(n)$ is constant then Diminishing Adaption might not hold - interest in the severity of the error.

To Log or Not To Log?

Taking the log of a random variable is known to be useful for heavy tailed distributions. Can an adaptive algorithm know when it is advantageous to look at the density for the log of the random variable instead?

- To avoid negative or near-negative values, modify log to

$$l(w) = sgn(w)log(1 + |w|)$$

where

$$sgn(w) = \begin{cases} 1 & \text{if } w > 0 \\ -1 & \text{if } w < 0 \end{cases}$$

To Log or Not To Log?

- The target for $log(W)$ is $\tilde{\pi}(w) = e^{|w|}\pi(e^{|w|} - 1)$
- A random-walk Metropolis algorithm will be ergodic iff π satisfies

$$log\pi(x) - log\pi(y) \geq \alpha(y - x), y > x \geq x_1$$

  for $x_1 > 0, \alpha > 0$
- Note that if π satisfies this condition then so does $\tilde{\pi}$

To Log or Not To Log?

Consider Random-Walk algorithms on both π and $\tilde{\pi}$

- $Q(x, \cdot) = N(x, \sigma^2)$
- After n^{th} batch of 100 iterations, allow each to adapt scaling parameter σ by adding or subtracting $\delta(n)$ to $log(\sigma)$ to acheive an acceptance rate as close to 0.44 as possible.
- Once every 100 iterations, decide whether to switch versions based on whether the average squared jumping distance is smaller than from the last time of the other version.

To Log or Not To Log?

Target	Log percent	ls_{reg}	ls_{log}
Normal	3.62	2.52	2.08
Cauchy	99.0	3.49	2.66
Uniform	4.95	6.66	2.65

- Log percent is percentage of the time that the adaptive algorithm spent on the logged density $\tilde{\pi}$
- *ls* is mean of the log proposal standard deviation
- Advantage in taking the log comes, as expected, from distributions with heavy tails.
- Possible to extend to higher dimensions, which is especially useful to avoid taking the log by hand for each coordinate separately.

Conclusion

- Adaptive MCMC provides promising results for finding good values for proposal variance, especially in cases of high dimension when it is unreasonable to do by hand.
- See `probability.ca/adapt` for the c code! (Some take forever..)