Computational Statistics: Markov Chain Monte Carlo

Dylan Bahia, Harry Newton, Joe Rutherford, Rui Zhang

STOR-i | Lancaster University

MCMC algorithms

- Random Walk Metropolis (RWM)
- Metropolis-adjusted Langevin algorithm (MALA)
- Hamiltonian Monte Carlo

RWM

- Given a target density $\pi(\boldsymbol{x})$, the RWM uses the following proposal density:

$$q(\boldsymbol{x}' \mid \boldsymbol{x}) = \boldsymbol{x} + \mathrm{N}(0, \sigma).$$

RWM

- Given a target density $\pi(\boldsymbol{x})$, the RWM uses the following proposal density:

$$q(\boldsymbol{x'} \mid \boldsymbol{x}) = \boldsymbol{x} + \mathrm{N}(0, \sigma).$$

- MALA uses the following proposal density:

$$q(\boldsymbol{x'} \mid \boldsymbol{x}) = \boldsymbol{x} + \frac{\sigma}{2}\nabla \log \pi(\boldsymbol{x}) + \mathrm{N}(0, \sigma).$$

Illustration

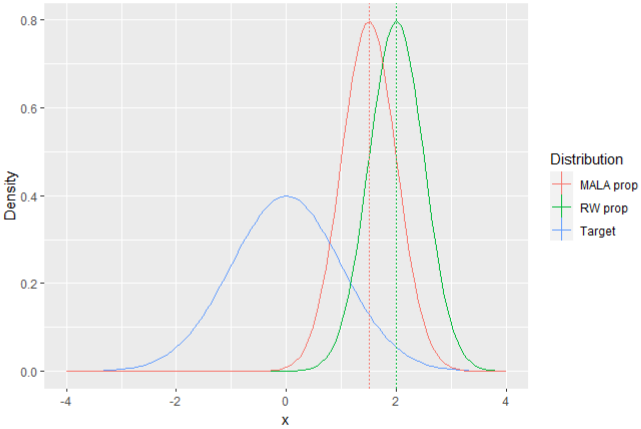


Figure 1: The densities of the two proposal distributions given the current value of the chain is 2. Target distribution is a standard Normal and $\sigma = 0.5$ for both proposals.

HMC

- Uses Hamiltonian dynamics.

HMC

- Uses Hamiltonian dynamics.
- $(\boldsymbol{x}, \boldsymbol{\rho})$:
  - $\boldsymbol{x}$ are the parameters of interest.
  - $\boldsymbol{\rho}$ are momentum variables.

Illustration

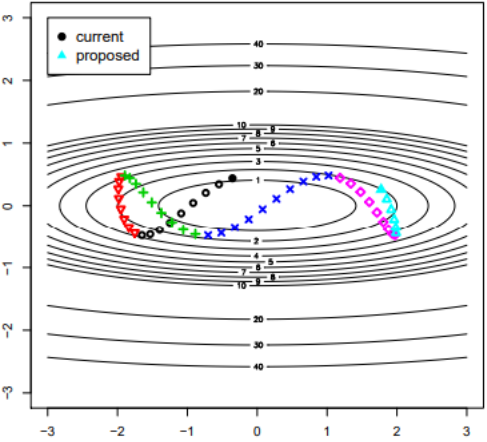


Figure 2: A simulation of the movement particle along the posterior surface [3].

Algorithm

Algorithm

- Joint density

$$\pi(\boldsymbol{x}, \boldsymbol{\rho}) = \pi(\boldsymbol{\rho} \mid \boldsymbol{x})\pi(\boldsymbol{x}).$$

Algorithm

- Joint density

$$\pi(\boldsymbol{x}, \boldsymbol{\rho}) = \pi(\boldsymbol{\rho} \mid \boldsymbol{x})\pi(\boldsymbol{x}).$$

- Define

$$\begin{aligned} H(\boldsymbol{x}, \boldsymbol{\rho}) &= -\log \pi(\boldsymbol{x}, \boldsymbol{\rho}) \\ &= -\log \pi(\boldsymbol{\rho} \mid \boldsymbol{x}) - \log \pi(\boldsymbol{x}) \\ &= \text{Kinetic energy} + \text{potential energy}. \end{aligned}$$

Algorithm

- Need to solve the following differential equations:

$$\frac{d\boldsymbol{x}}{dt} = -\nabla \log \pi(\boldsymbol{\rho})$$
$$\frac{d\boldsymbol{\rho}}{dt} = \nabla \log \pi(\boldsymbol{x})$$

Algorithm

- Need to solve the following differential equations:

$$\frac{d\boldsymbol{x}}{dt} = -\nabla \log \pi(\boldsymbol{\rho})$$
$$\frac{d\boldsymbol{\rho}}{dt} = \nabla \log \pi(\boldsymbol{x})$$

- Leapfrog integrator: need a step size ϵ and number of steps L.

MH step

- Given a current position $\boldsymbol{x}_n$ and momentum $\boldsymbol{\rho}_n$;

MH step

- Given a current position $\boldsymbol{x}_n$ and momentum $\boldsymbol{\rho}_n$;
- Simulate $\boldsymbol{\rho}$ from $N(0, \mathbb{I})$.

MH step

- Given a current position $\boldsymbol{x}_n$ and momentum $\boldsymbol{\rho}_n$;
- Simulate $\boldsymbol{\rho}$ from $N(0, \mathbb{I})$.
- Repeat the following L times (Leapfrog integrator):
  - $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \frac{\epsilon}{2} \nabla \log(\pi(\boldsymbol{x}))$
  - $\boldsymbol{x} \leftarrow \boldsymbol{x} + \epsilon \boldsymbol{\rho}$
  - $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \frac{\epsilon}{2} \nabla \log(\pi(\boldsymbol{x}))$

MH step

- Given a current position $\boldsymbol{x}_n$ and momentum $\boldsymbol{\rho}_n$;
- Simulate $\boldsymbol{\rho}$ from $N(0, \mathbb{I})$.
- Repeat the following L times (Leapfrog integrator):
  - $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \frac{\epsilon}{2} \nabla \log(\pi(\boldsymbol{x}))$
  - $\boldsymbol{x} \leftarrow \boldsymbol{x} + \epsilon \boldsymbol{\rho}$
  - $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \frac{\epsilon}{2} \nabla \log(\pi(\boldsymbol{x}))$
- Accept $(\boldsymbol{x}', \boldsymbol{\rho}')$ with probability

$$\min(1, \exp(H(\boldsymbol{x}_n, \boldsymbol{\rho}_n) - H(\boldsymbol{x}', \boldsymbol{\rho}')))$$

MH step

- Given a current position $\boldsymbol{x}_n$ and momentum $\boldsymbol{\rho}_n$;
- Simulate $\boldsymbol{\rho}$ from $N(0, \mathbb{I})$.
- Repeat the following L times (Leapfrog integrator):
  - $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \frac{\epsilon}{2} \nabla \log(\pi(\boldsymbol{x}))$
  - $\boldsymbol{x} \leftarrow \boldsymbol{x} + \epsilon \boldsymbol{\rho}$
  - $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \frac{\epsilon}{2} \nabla \log(\pi(\boldsymbol{x}))$
- Accept $(\boldsymbol{x}', \boldsymbol{\rho}')$ with probability

$$\min(1, \exp(H(\boldsymbol{x}_n, \boldsymbol{\rho}_n) - H(\boldsymbol{x}', \boldsymbol{\rho}')))$$

- $L = 1$ gives MALA algorithm

Choosing L and ϵ

Choosing L and ϵ

- Can be difficult to choose L and ϵ which facilitate efficiency.

Choosing L and ϵ

- Can be difficult to choose L and ϵ which facilitate efficiency.
- Poor choices can compromise ergodicity.

Choosing L and ϵ

- Can be difficult to choose L and ϵ which facilitate efficiency.
- Poor choices can compromise ergodicity.
- One solution: use NUTS algorithm.

Choosing L and ϵ

- Can be difficult to choose L and ϵ which facilitate efficiency.
- Poor choices can compromise ergodicity.
- One solution: use NUTS algorithm.
- Another solution: Use AAPS algorithm.

Tuning

From earlier:

- RWM uses the following proposal density:

$$q(\boldsymbol{x}' \mid \boldsymbol{x}) = \boldsymbol{x} + \mathrm{N}(0, \sigma).$$

- MALA uses the following proposal density:

$$q(\boldsymbol{x}' \mid \boldsymbol{x}) = \boldsymbol{x} + \frac{\sigma}{2}\nabla \log \pi(\boldsymbol{x}) + \mathrm{N}(0, \sigma).$$

We must find the value of σ that provides each target distribution with the optimal acceptance rate (α).

$$\alpha = \frac{\text{number of times we update the state}}{\text{total number of iterations}}$$

Optimal Scaling

	Parameter Scaling	Optimal Accept. Rate
RWM	$\propto d^{-1}$	0.234
MALA	$\propto d^{-1/3}$	0.574
HMC	$\propto d^{-1/4}$	0.651

Table 1: Optimal Scaling Results

Target Distributions

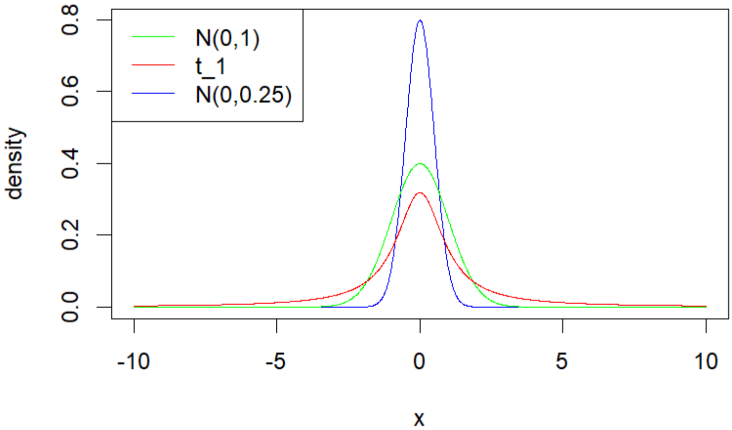


Figure 3: Graph of the densities of all chosen target distributions

Target Distribution 1

- Normal(0,1) distribution
- $f(x, y) = -\frac{1}{2}(x^2 + y^2)$
- Number of dimensions: $d = 2$
- Optimal acceptance rates:
  - RWM: $\alpha = 0.35$
  - MALA: $\alpha = 0.574$

Target Distribution 1 Graphs

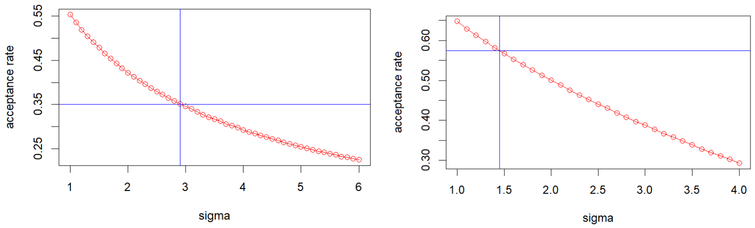


Figure 4: Graphs showing the effect sigma has on the acceptance rate of the Normal(0,1) target distribution

Target Distribution 2

- Product of t-distributions with 1 degree of freedom ($\nu = 1$)
- $f(x) = \prod_{i=1}^{d} \left(\frac{1}{\pi(1+x_i^2)} \right)$
- Number of dimensions: d
- Optimal acceptance rates ($d = 1$):
  - RWM: $\alpha = 0.44$
  - MALA: $\alpha = 0.574$

Target Distribution 2 Graphs

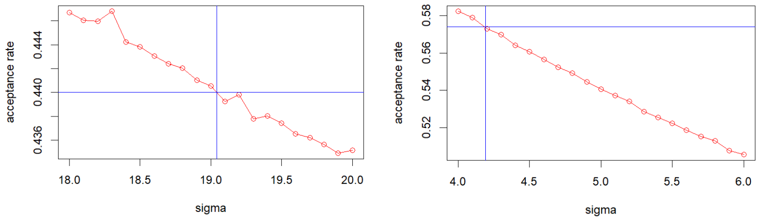
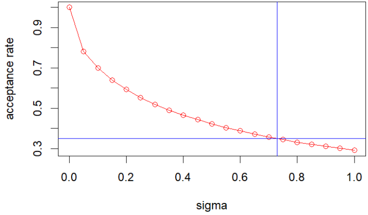


Figure 5: Graphs showing the effect sigma has on the acceptance rate of the product of t_1 target distribution

Target Distribution 3

- Normal(0,0.25) distribution
- $f(x, y) = -\frac{1}{2}(4x^2 + 4y^2)$
- Number of dimensions: $d = 2$
- Optimal acceptance rates:
  - RWM: $\alpha = 0.35$
  - MALA: $\alpha = 0.574$

Target Distribution 3 Graphs



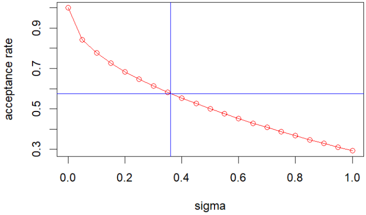


Figure 6: Graphs showing the effect sigma has on the acceptance rate of the Normal$(0,0.5^2)$ target distribution

Summary

	RWM	MALA
N(0,1)	2.9	1.45
t_1	19.04	4.19
N(0,0.5^2)	0.73	0.36

Figure 7: Table showing the optimal σ values for each distribution for each method

Is our MCMC Efficient?

Most MCMC algorithms eventually converge on their target distribution.

An efficient MCMC algorithm will converge if it can explore the parameter space effectively. Algorithms which do this have good **mixing**.

Convergence: Gelman-Rubin Diagnostic [2] (2)

Diagnositc Procedure:

1. Run m parallel MCMC chains with varied initial conditions. These ICs should be more dispersed than $\pi(x)$.
2. Calculate $\hat{V}$ and W

W is the within chain variance
$\hat{V}$ is the pooled variance estimate

Convergence: Gelman-Rubin Diagnostic [2]

This diagnostic defines a quantity $\hat{R}$; known as the **potential scale reduction factor**. MCMC converges with target as $\hat{R} \to 1$.

$$\hat{R} = \frac{\hat{V}}{W}$$

$\hat{R} \leq 1.1$ is seen as a good indicator of convergence, but not always...

Gelman-Rubin Diagnostic

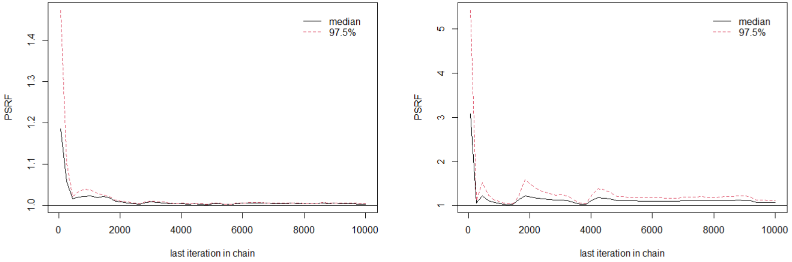


Figure 8: Variation in GR for $m = 10$ for MALA (left) and RW (right). Data was obtained using Gelman.plot in CODA.

Efficiency: ESS

An efficient MCMC is one where a large proportion of the samples provide quality information on the target distribution. Quantified by **Effective Sample Size**:

$$\text{ESS} = \frac{N}{1 + 2\sum_i Corr_\pi(g(X_0), g(X_i))}$$

Efficiency: ESS

When dealing with multi-variate distributions it is more appropriate to use the **Multi-Variate ESS** (MESS) [4], based upon

$ESS = n\left(\frac{\lambda_g^2}{\sigma_g^2}\right)$ [1]

$$\text{MESS} = N\left(\frac{|\Lambda_g|}{|\Sigma_g|}\right)^{\frac{1}{p}}$$

Λ_g is the population covariance matrix. Σ is an estimate of Monte Carlo standard error.

ESS Comparison

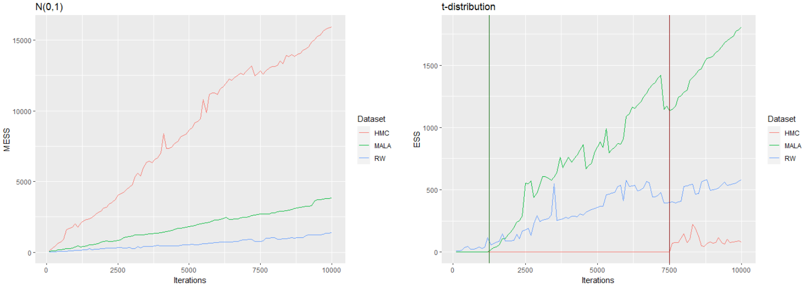


Figure 9: Uni-variate (t-distb) $IC = 8.733$, 2D (Gaussian) $IC = [8.733, -0.407]$. MALA and RW used tuned h values, and HMC was tuned at $L = 200$, $\epsilon = 0.01$. ESS was calculated using coda and MESS was calculated with mcmcse package

Adaptive MCMC

	Parameter Scaling	Optimal Accept. Rate
RWM	$\propto d^{-1}$	0.234
MALA	$\propto d^{-1/3}$	0.574
HMC	$\propto d^{-1/4}$	0.651

Table 2: Optimal Scaling Results

Adaptive MCMC

	Parameter Scaling	Optimal Accept. Rate
RWM	$\propto d^{-1}$	0.234
MALA	$\propto d^{-1/3}$	0.574
HMC	$\propto d^{-1/4}$	0.651

Table 2: Optimal Scaling Results

Can we use them on the fly?

Adaptive MCMC

	Parameter Scaling	Optimal Accept. Rate
RWM	$\propto d^{-1}$	0.234
MALA	$\propto d^{-1/3}$	0.574
HMC	$\propto d^{-1/4}$	0.651

Table 2: Optimal Scaling Results

Can we use them on the fly? Adaptive MCMC!

Adaptive MCMC

Goal: Tune the parameter(s) adaptively to match the "ideal" parameterisation.

Adaptive MCMC

Goal: Tune the parameter(s) adaptively to match the "ideal" parameterisation.

One way to do so ... Solve this:

$$\mathbb{E}\left[\alpha(x, y)\right] - \alpha^* = 0$$
$$\mathbb{E}\left[(X, XX^T)\right] - (\mu, \Sigma) = 0$$

Adaptive MCMC

Goal: Tune the parameter(s) adaptively to match the "ideal" parameterisation.

One way to do so ... Solve this:

$$\mathbb{E}\left[\alpha(x,y)\right] - \alpha^* = 0$$
$$\mathbb{E}\left[(X, XX^T)\right] - (\mu, \Sigma) = 0$$

Current state x, proposed state y, current step acceptance rate $\alpha(x,y)$, α^* optimal acceptance rate, states so far X, target distribution mean μ covariance Σ.

Adaptive MCMC

Solve it via Robbins-Monro iterations.

Adaptive MCMC

Solve it via Robbins-Monro iterations.

Given current state X_i, proposed step Y_{i+1}, next step X_{i+1}. Target acceptance rate α^*. Current stepsize λ_i, mean μ_i, covariance matrix Σ_i. Current parameter of algorithm $\sigma_i = \lambda_i \Sigma_i$.

Adaptive MCMC

Solve it via Robbins-Monro iterations.

Given current state X_i, proposed step Y_{i+1}, next step X_{i+1}. Target acceptance rate α^*. Current stepsize λ_i, mean μ_i, covariance matrix Σ_i. Current parameter of algorithm $\sigma_i = \lambda_i \Sigma_i$.

$$\log(\lambda_{i+1}) = \log(\lambda_i) + \gamma_{i+1}[\alpha(X_i, Y_{i+1}) - \alpha^*]$$

$$\mu_{i+1} = \mu_i + \gamma_{i+1}(X_{i+1} - \mu_i)$$

$$\Sigma_{i+1} = \Sigma_i + \gamma_{i+1}[(X_{i+1} - \mu_i)(X_{i+1} - \mu_i)^T - \Sigma_i]$$

Adaptive MCMC

Solve it via Robbins-Monro iterations.

Given current state X_i, proposed step Y_{i+1}, next step X_{i+1}. Target acceptance rate α^*. Current stepsize λ_i, mean μ_i, covariance matrix Σ_i. Current parameter of algorithm $\sigma_i = \lambda_i \Sigma_i$.

$$\log(\lambda_{i+1}) = \log(\lambda_i) + \gamma_{i+1}[\alpha(X_i, Y_{i+1}) - \alpha^*]$$
$$\mu_{i+1} = \mu_i + \gamma_{i+1}(X_{i+1} - \mu_i)$$
$$\Sigma_{i+1} = \Sigma_i + \gamma_{i+1}[(X_{i+1} - \mu_i)(X_{i+1} - \mu_i)^T - \Sigma_i]$$

$\{\gamma_i\}$ is the learning rate, e.g. $\gamma_i = i^{-0.8}$.

Problems with Adaptive MCMC

1. Still ergodic?

Problems with Adaptive MCMC

1. Still ergodic?
2. Some algorithms react badly to poor tuning.

An Example

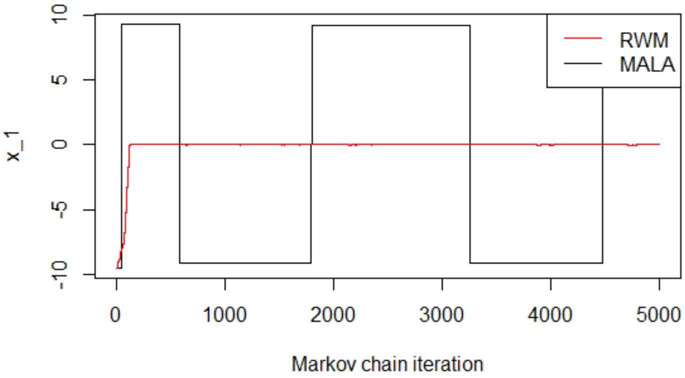
Target: 100-dimensional Gaussian with mean 0 and covariance matrix

$$\Sigma = \begin{bmatrix} 0.01^2 & 0 \\ 0 & I_{99}. \end{bmatrix}$$

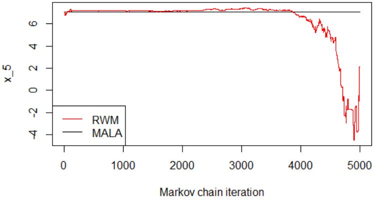
Algorithms: RWM and MALA with Adaptive Tuning

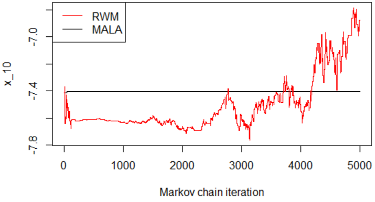
Initial Position: 100 dimensional Gaussian with mean 0 and variance 10.

An Example


10
5
0
-5
-10
x_1
0
1000
2000
3000
4000
5000
Markov chain iteration
RWM
MALA

An Example


x_5
6
4
2
0
-2
-4
RWM
MALA
0
1000
2000
3000
4000
5000
Markov chain iteration


x_10
-7.0
-7.4
-7.8
RWM
MALA
0
1000
2000
3000
4000
5000
Markov chain iteration

Efficiency vs. Robustness to Tuning

	Efficiency	Robustness
RWM	Low	High
MALA	High	Low

Further Research

1. How the dimension impacts efficiency of algorithms
2. Better convergence diagnostics? (e.g. KSD)
3. How to adapt better? (e.g. theory of adaptive MCMC, ML adaptations)
4. Formal robustness to tuning (e.g. spectral gap)
5. Combining efficiency and robustness? (e.g. the Barker proposal)

Summary

1. RWM, MALA, HMC
2. Parameter tuning and optimal scaling
3. MCMC output diagnostics
4. Adaptive MCMC

Reference

[1] James M. Flegal and Galin L. Jones.
Batch means and spectral variance estimators in markov chain monte carlo.
The Annals of Statistics, 38(2), April 2010.

[2] Vivekananda Roy.
Convergence diagnostics for markov chain monte carlo.
Annual Review of Statistics and Its Application, 7(1):387–412, 2020.

[3] Chris Sherlock, Szymon Urbas, and Matthew Ludkin.
The apogee to apogee path sampler, 2022.

[4] Dootika Vats, James M. Flegal, and Galin L. Jones.
Multivariate output analysis for markov chain monte carlo, 2017.