STOR608: MCMC

Paul Fearnhead

Department of Mathematics & Statistics, Lancaster University

November 2021

Part I: RECAP and INTRODUCTION

- Relevant theory for Markov chains
- What is MCMC
- Metropolis-Hastings Algorithm

Discrete-state Markov Chains

Recap: Dynamics of a discrete-state Markov chain are described by a propability transition matrix, P.

$$P_{ij} = \Pr(X_t = j | X_{t-1} = i).$$

An invariant distribution can be described by a pmf π such that

$$\pi(j) = \sum_i \pi(i) P_{ij}, \quad \text{for all } j.$$

This can be written as $\pi = \pi P$.

If a chain has an invariant distribution and is irreducible and aperiodic then the invariant distribution is unique and is its stationary distribution, i.e.

$$\pi_t \to \pi \text{ as } t \to \infty \text{ for any } \pi_0.$$

Detailed Balance

A sufficient condition for π to be an invariant distribution is that is satisfies detailed balance

$$\pi(i)P_{ij} = \pi(j)P_{ji}$$

for all i, j.

If detailed balance holds then the chain is π- reversible: i.e. if the initial distribution is π then the dynamics of the chain forward in time is identical to the dynamics backward in time.

General-state Markov Chains

A Markov chain on a continuous-state space is defined by a transition kernel $K(\cdot|\cdot)$, such that

$$K(\mathcal{A}|x) = \Pr(X_t \in \mathcal{A}|X_{t-1} = x).$$

Normally this kernel will have a transition density $p(\cdot|\cdot)$

$$K(\mathcal{A}|x) = \int_{\mathcal{A}} p(y|x)\mathrm{d}y.$$

An invariant distribution $\Pi(\cdot)$, with density $\pi(x)$, satisfies

$$\Pi(\mathcal{A}) = \int K(\mathcal{A}|x)\pi(x)\mathrm{d}x.$$

General-state Markov Chains

If the chain is irreducible and aperiodic then the invariant distribution is unique and is its stationary distribution, i.e.

$$\pi_t \to \pi \text{ as } t \to \infty \text{ for any } \pi_0.$$

A sufficient condition for π to be the invariant pdf is that it satisfies detailed balance

$$\pi(x)p(y|x) = \pi(y)p(x|y), \text{ for almost all } x, y.$$

The chain is then reversible.

MCMC

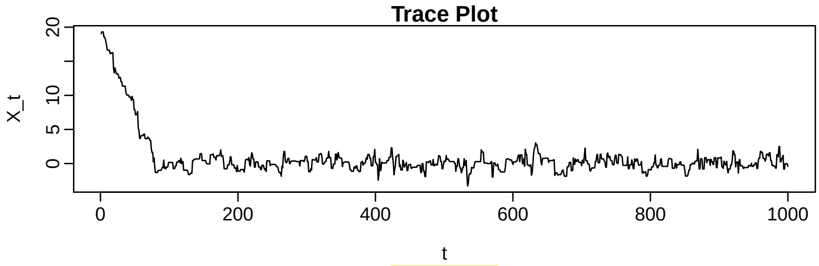
We are going to consider Markov chains that are designed to have a specific stationary distribution, $\pi(x)$.

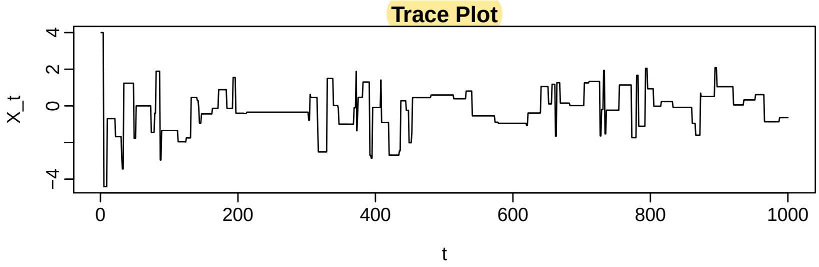
Commonly $\pi(x)$ will be a Bayesian posterior distribution, and we know $\pi(x)$ only up to proportionality.

The idea is that we can simulate our Markov chain, to produce draws $x_1, \ldots, x_N$. Drop the first b draws (as burn-in), and then use $x_{b+1}, \ldots, x_n$ as (approximate, dependent) samples from π.

These can be used to estimate posterior expectations/probabilities. The accuracy of these estimates will depend on how quickly the chain converges to stationarity (i.e. how much burn-in) and the auto-correlation of the chain.

MCMC: Two Examples





Metropolis-Hastings

If we are told π (up to proportionality) how do we construct a Markov chain that has π as its stationary distribution?

The Metropolis-Hastings algorithm: we choose "any" Markov dynamics, a proposal density $q(x|y)$. Simulate a potential new state using this density, and then accept or reject it with probability

$$\alpha(x|y) = \min\left\{1, \frac{\pi(x)q(y|x)}{\pi(y)q(x|y)}\right\}.$$

If we reject we stay where we are.

This produces a reversible chain with the correct invariant distribution.
Unknown normalising constant cancels!

Part II: EXAMPLE MCMC ALGORITHMS

- See some common MCMC algorithms
- Gain an insight into how they mix/different properties
- Understand the need for tuning

We will use demos from `https://chi-feng.github.io/mcmc-demo/`.

Random-walk Metropolis

Arguably the simplest and most common MCMC algorithm is random walk Metropolis. This uses a proposal distribution that is a random walk.

The proposed state, x', is obtained as the realisation of a random variable defined as the current state, x plus noise:

$$X' = x + \epsilon,$$

where ϵ is some zero-mean random variable independent of x.

If ϵ is a symmetric random variable that proposal will satisfy $q(x'|x) = q(x|x')$ which leads to cancellation in the acceptance probability.

Tuning

Most commonly ϵ is a Gaussian random variable with mean 0 and variance $h\mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

A user then needs to choose the value of h.

This choice is important for the mixing of the MCMC algorithm.

Gibbs Sampler

Write $x = (x^{(1)}, \dots, x^{(d)})$. Whilst the joint distribution for x may be complicated, often the conditional distribution for $x^{(i)}$ given the remaining variables, often denoted $x^{(-i)}$, is tractable.

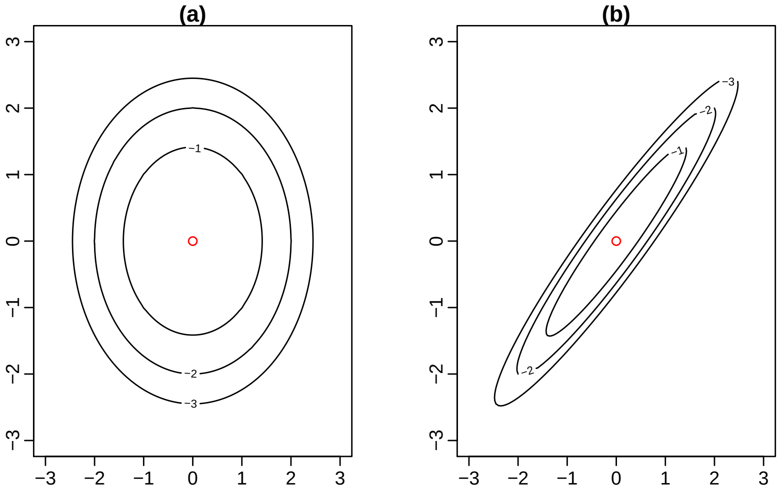
The Gibbs sampler involves proposals of the form $\pi(x^{(i)}|x^{(-i)})$.

The acceptance probability is always 1.

For irreducibility you need to update all components – often in turn (deterministic scan) or at random (random scan).

What determines the efficiency of the Gibbs sampler? Parameterisation is important.

Gibbs Sampler



Gibbs Sampler

Rather than updating univariate components, you can update blocks of components. If this is possible it should improve mixing. Choice of blocks is important.

If the full conditional is not tractable you can use a Metropolis step to update e.g. $x^{(i)}$ given $x^{(-i)}$. This is called Metropolis within Gibbs.

MALA: Langevin Diffusion

The Langevin diffusion is a continuous-time process, defined by the SDE

$$\mathrm{d}X_t = \frac{1}{2}\nabla \log \pi(X_t)\mathrm{d}t + \mathrm{d}B_t$$

This can be interpreted via the approximate dynamics (which are "exact" as $h \to 0$):

$$X_{t+h} = x_t + \frac{h}{2}\nabla \log \pi(x_t) + \sqrt{h}\epsilon_t,$$

where ϵ_t is a vector of IID standard Gaussian rvs.

The Langevin diffusion has $\pi(x)$ as its stationary distribution – but it is impossible to simulate the diffusion exactly.

MALA

We can use the Langevin diffusion to motivate a proposal distribution

$$X' = x + \frac{h}{2}\nabla \log \pi(x) + \sqrt{h}\epsilon_t,$$

and correct for the approximation by the accept-reject step.
(Metropolis-adjusted Langevin Algorithm).

This can have better properties than random walk MH. Again choice of h is important.

Hamiltonian Dynamics

Good mixing requires proposals that move the state a lot but still have high acceptance probability. This motivates the use of dynamic models to propose new values – as these models conserve energy.

Augment our state with a velocity (or momentum) component p. Introduce a potential energy that is $-\log \pi(x)$, and a kinetic energy $p^T M^{-1} p/2$. This gives a Hamiltonian

$$H(x,p) = -\log \pi(x) + \frac{1}{2} p^T M^{-1} p.$$

Hamilton's dynamics

$$\frac{\mathrm{d}x_s}{\mathrm{d}s} = \frac{\partial H}{\partial p} = M^{-1}p, \qquad \frac{\mathrm{d}p_s}{\mathrm{d}s} = -\frac{\partial H}{\partial x} = \nabla \log \pi(x)$$

preserves the value of the Hamiltonian.

HMC

We can view the Hamiltonian as minus the log of a density where x has density $\pi(x)$ and p is independent and has a Gaussian distribution with mean 0 and variance M.

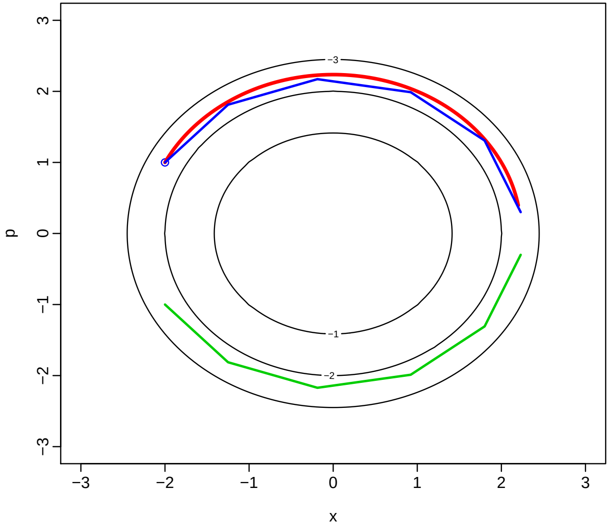
Thus if we could simulate the dynamics exactly we would have a proposal distribution for this joint distribution that would move us around contours of log density.

We can solve the dynamics approximately by the leapfrog approximation

$$p_{s+h/2} = p_s + \frac{h}{2}\nabla \log \pi(x_s), \quad x_{s+h} = x_s + hM^{-1}p_{s+h/2},$$
$$p_{s+h} = p_{s+h/2} + \frac{h}{2}\nabla \log \pi(x_{s+h})$$

and use this (simulated over a suitable time horizon) as a proposal.

HMC



Red is true dynamics; blue is leap-frog path; green is reverse leap-frog path.

HMC

The leapfrog dynamics are deterministic, but have two key properties

- They are anti-reversible.
- They are volume-preserving

This means we can use them within a Metropolis Hasting algorithm.

HMC: One iteration

Given a current state (x, p):

(i) Use the leapfrog method with K steps of size h to simulate a new state (x', p').

(ii) Flip the momentum, to produce the proposed state $(x', -p')$.

(iii) Accept-reject with probability

$$\min\left\{1, \frac{\pi(x')\exp(-p'^T M^{-1} p')}{\pi(x)\exp(-p^T M^{-1} p)}\right\}$$

For irreducibilty we need to move contours – often via a Gibbs-step for p.

(iv) Simulate a new momentum variable $p'' \sim N(0, M)$.

If we do this we can omit step (ii) from the above algorithm.

HMC

Tuning of HMC is particularly important, and involves choosing two parameters, K and h.

If we fix Kh then h affects the numerical error in the Leap-frog integration.

The value Kh affects how far around the contours you move – if chosen poorly you can inadvertently return to close to where you start. (Recent work has tried to come up with adaptive procedure for choosing Kh.)

Part III: EFFICIENCY, THEORY, CODING and PROJECT

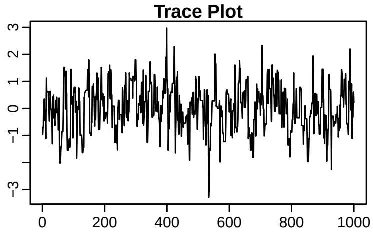
- Understand how to monitor/evaluate mixing of an MCMC algorithm.
- Understand basic theory on tuning/scaling of different algorithms
- See how easy it is to code up basic MCMC algorithms!
- Understand the group projects.

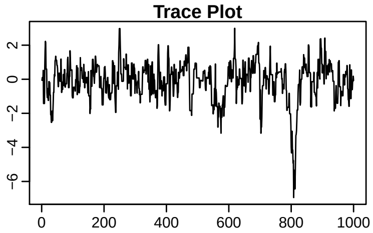
Challenge

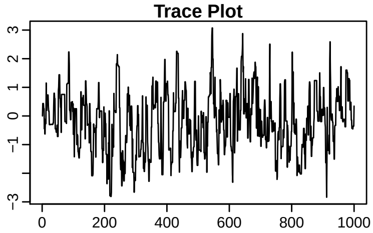
You should code and try out three or four of these MCMC algorithms with a view to answering the questions

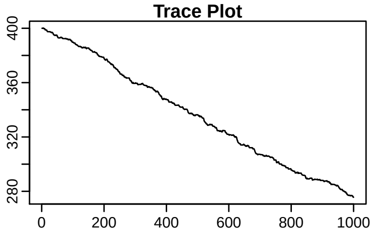
- How should a practitioner tune a given MCMC algorithm?
- If you have a uni-modal target distribution, which MCMC algorithm is best? How does this depend on properties of the target distribution (such as dimension, correlation structure, tail behaviour etc.)

Convergence









Convergence

It can be difficult to know whether your MCMC algorithm has converged to stationarity.

The key issues tend to be (i) missing modes if the target is multi-modal; or (ii) not sampling the tails if the target is heavy-tailed.

Often to try and detect convergence people run independent MCMC algorithms from different starting values.

[In simulation studies, e.g. for your project, this is less of an issue as you often know what the target is!]

Assessing accuracy

To estimate $\mathsf{E}_\pi[g(X)]$ from MCMC we use the empirical average

$$\frac{1}{N-B}\sum_{i=B+1}^{N} g(x_i).$$

We can estimate the accuracy of this estimate of the mean by its Monte Carlo Variance

$$\frac{1}{(N-B)^2}\sum_{i=B+1}^{N}\sum_{j=B+1}^{N}\mathrm{Var}[g(X_i),g(X_j)],$$

The variance of this is approximately

$$\frac{1}{N-B}\mathrm{Var}[g(X)]\left(1+2\sum_{l=1}^{\infty}\mathrm{Cor}[g(X_1),g(X_{1+l})]\right)$$

Assessing accuracy

$$\sum_{i=B+1}^{N} \sum_{j=B+1}^{N} \mathrm{Var}[g(X_i), g(X_j)] = \mathrm{Var}[g(X)] \sum_{i=B+1}^{N} \sum_{j=B+1}^{N} \mathrm{Cor}[g(X_i), g(X_j)]$$

As at stationarity X_i and X_j have same distribution (as X).

Assessing accuracy

$$\begin{aligned}\sum_{i=B+1}^{N}\sum_{j=B+1}^{N}\mathrm{Cor}[g(X_i),g(X_j)] &= \sum_{i=B+1}^{N}\sum_{l=B+1-i}^{N-i}\mathrm{Cor}[g(X_i),g(X_{i+l})]\\ &\approx \sum_{i=B+1}^{N}\sum_{l=-\infty}^{\infty}\mathrm{Cor}[g(X_i),g(X_{i+l})]\end{aligned}$$

Using a change of variable $j = i + l$, and that the correlation is negligible for large l.

The final step is note that each term in the sum is the same; and that the sum over strictly negative l is the same as over strictly positive l.

Assessing accuracy

To estimate $\mathsf{E}_\pi[g(X)]$ from MCMC we use the empirical average

$$\frac{1}{N-B}\sum_{i=B+1}^{N} g(x_i).$$

This is approximately

$$\frac{1}{N-B}\mathrm{Var}[g(X)]\left(1+2\sum_{l=1}^{\infty}\mathrm{Cor}[g(X_1),g(X_{1+l})]\right)$$

Assessing accuracy

So we can assess the accuracy by (i) choosing some function $g(\cdot)$; (ii) plotting the empirical autocorrelation of $g(x_i)$; and estimating the auto-correlation time

$$\tau = 1 + 2\sum_{l=1}^{\infty} \text{Cor}[g(X_1), g(X_{1+l})]$$

or the effective sample size $(N - B)/\tau$.

These measures depend on the choice of $g(\cdot)$.

Estimating τ is difficult – often people assume that the correlation is negligible beyond some large lag L and use

$$\hat{\tau} = 1 + 2\sum_{l=1}^{L} \hat{\text{Cor}}[g(X_1), g(X_{1+l})]$$

Code for RWM and Diagnostics

Theory

The performance of MCMC algorithms has been analysed theoretically. Often these are under very restrictive conditions on the target distribution – though these results give insight into behaviour in other situations.

One important set of results is for target distributions that are for d IID copies of a random variable, and look at performance as $d \to \infty$. These give the following results:

Algorithm	Optimal h	Acceptance Probability
Random Walk	$\propto d^{-1}$	0.234
MALA	$\propto d^{-1/3}$	0.574
HMC (fixed Kh)	$\propto d^{-1/4}$	0.651

Related theory show that you want to match the shape of the variance of the proposal to the target.

Challenge

You should code and try out three or four of these MCMC algorithms with a view to answering the questions

- How should a practitioner tune a given MCMC algorithm?
- If you have a uni-modal target distribution, which MCMC algorithm is best? How does this depend on properties of the target distribution (such as dimension, correlation structure, tail behaviour etc.)

To do this you should perform a thorough and well-designed simulation study for simple target distributions (e.g. Gaussians, or products of t-distributions); you should also relate your findings to theory on optimal tuning of MCMC algorithms.

Recommendations

- You will need to have a reliable way of assessing efficiency of an MCMC algorithm from its output. Comparisons should take account of computational cost.
- You will need to focus your study It is better to get clear recommendations from a focussed study, than try to cover everything.
- Start simple (e.g. IID Gaussian, then independent Gaussian with different variances, then have same variance but different levels of correlation).
- If possible present results in a Figure not a Table.
- You may want to fix many of the components of your study and vary just one at a time.

Recommendations

- In choosing your target distributions you may want to choose classes of proposals that have a single parameter that you can vary – to look at how MCMC behaviour changes.
- You may want to compare your empirical results to the theory.
- When you code things up you must assume there are bugs in the code, and then perform tests to convince yourself there are none.

References (all online)

Introductions to MCMC: Geyer (2011) Introduction to Markov chain Monte Carlo, in Handbook of MCMC. This includes some discussion on assessing accuracy in Section 1.8.2, though you can do better than (1.10)
Craiu and Rosenthal (2014) Bayesian computation via Markov chain Monte Carlo, Annual Review of Statistics.

Overview of HMC: Neal (2011) MCMC using Hamiltonian Dynamics in Handbook of MCMC (arXiv.1206.1901)

Scaling/Theory: Roberts and Rosenthal (2001) Optimal scaling for various Metropolis-Hastings algorithms, Statistical Science
Roberts and Sahu (1997) Updating schemes, correlation structure, blocking and parameterization for the Gibbs Sampler, JRSS B
Beskos et al. (2013) Optimal tuning of the hybrid Monte Carlo algorithm. Bernoulli. Introduction summarises the paper well!

No reference will include just what you need – you need to learn the skill of deciding aspects of the paper are important and what can be skipped.