Implementing Hamiltonian Monte Carlo for Efficient Bayesian Evolutionary Analysis

AWC SUMMER SCHOLARSHIP REPORT

Arman Bilge
armanbilge@gmail.com

22 April 2015

Introduction

Bayesian evolutionary analysis is centered around sampling from the posterior probability distribution of a phylogenetic tree $\mathcal{T}$ and model parameters $\boldsymbol{\theta}$ given the molecular sequence data $\mathcal{D}$ [Bou+14], which by Bayes' theorem is

$$p(\mathcal{T}, \boldsymbol{\theta} \mid \mathcal{D}) \propto p(\mathcal{D} \mid \mathcal{T}, \boldsymbol{\theta})\, p(\mathcal{T} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta}). \tag{1}$$

Although direct sampling from this distribution is impossible, sampling strategies utilising Markov chain Monte Carlo (MCMC) techniques [Met+53] have been quite successful [Ron+12; Dru+12; Bou+14]. The ideal sampling algorithm maximises the number of pseudo-independent samples drawn, called the effective sample size (ESS), by minimising the correlation between consecutive samples. MCMC generally performs poorly by this measure. Because the proposal distribution is ignorant of the target distribution, MCMC exhibits random-walk behaviour and requires several steps to reach an uncorrelated sample [Nea11]. Hamiltonian Monte Carlo (HMC), first described as hybrid Monte Carlo [Dua+87], has the important advantage that the proposal distribution is the target distribution, enabling large changes to be made with each step. Computing an HMC proposal involves solving a system of differential equations which is substantially more time-consuming than an MCMC update. However, Neal [Nea11] showed that, in theory, HMC will outperform MCMC as the problem dimensionality increases. I applied HMC to Bayesian evolutionary analysis and tested for improved performance.

Methods

Let $\pi(\mathbf{q})$ be the target probability density. We can consider a particle in our state space with position given by $\mathbf{q}$ and momentum by $\mathbf{p}$. Then the Hamiltonian for our system is

$$H(\mathbf{q}, \mathbf{p}) = U(\mathbf{q}) + K(\mathbf{p}) \tag{2}$$

with potential energy $U(\mathbf{q}) = -\log \pi(\mathbf{q})$ and kinetic energy $K(\mathbf{p}) = \frac{1}{2}\mathbf{p}^T\mathbf{M}\mathbf{p}$, where $\mathbf{M}$ is a positive-definite matrix that we interpret as the particle's mass. The system's dynamics are described by Hamilton's equations,

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = \frac{\partial H}{\partial \mathbf{p}}, \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = -\frac{\partial H}{\partial \mathbf{q}}, \tag{3}$$

which are integrated to find the state at a particular time. Typically this integration is achieved numerically using the leapfrog method [Nea11]. I use $\mathbf{L}\{\mathbf{q}, \mathbf{p}\}$ to denote an operator that maps the state at time t to the state at time $t + \epsilon L$, as approximated by L leapfrog steps with stepsize ϵ. Additionally, the momentum flip operator gives $\mathbf{F}\{\mathbf{q}, \mathbf{p}\} = \{\mathbf{q}, -\mathbf{p}\}$. With the $\mathbf{L}$ and $\mathbf{F}$ operators we can reach a discrete set of states along an energy contour (subject only to errors in numerical integration). To guarantee ergodicity of the Markov chain, the momentum is randomised after each iteration of the algorithm by

$$\mathbf{R}\{\mathbf{q}, \mathbf{p}\} = \left\{\mathbf{q}, \sqrt{1-\alpha}\mathbf{p} + \sqrt{\alpha}\mathbf{n}\right\}, \mathbf{n} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{M}^{-1}\right), \tag{4}$$

where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. When the parameter $\alpha < 1$ the momentum is only partially refreshed, further suppressing random walk behaviour. Using these

Algorithm 1. A single iteration of the Hamiltonian Monte Carlo algorithm that uses Hamiltonian dynamics to make the proposal and the Metropolis criterion [Met+53] to accept or reject it.

function HAMILTONUPDATE($\{\mathbf{q}, \mathbf{p}\}$)
  $\{\mathbf{q}', \mathbf{p}'\} \leftarrow \mathbf{FL}\,\{\mathbf{q}, \mathbf{p}\}$
  $a \leftarrow \min\left(1, \exp\left(H\left(\mathbf{q}, \mathbf{p}\right) - H\left(\mathbf{q}', \mathbf{p}'\right)\right)\right)$
  $\{\mathbf{q}, \mathbf{p}\} \leftarrow \begin{cases} \{\mathbf{q}', \mathbf{p}'\} & \text{with probability } a \\ \{\mathbf{q}, \mathbf{p}\} & \text{with probability } 1 - a \end{cases}$
  $\{\mathbf{q}, \mathbf{p}\} \leftarrow \mathbf{RF}\,\{\mathbf{q}, \mathbf{p}\}$
  return $\{\mathbf{q}, \mathbf{p}\}$
end function

operators, the HMC algorithm is concisely described by making repeated calls to the HAMILTONUPDATE function given in Algorithm 1.

I implemented the described HMC algorithm in a fork of BEAST v1.8 [Dru+12]. In particular, I represented the HAMILTONUPDATE function as a standard MCMC operator whose Hastings ratio is the exponential of the difference between the current and proposed kinetic energy. To enable evaluation of the gradient of the potential energy, I added a method `differentiate()` to the core `Likelihood` interface and implemented it for the basic probability distributions, coalescent and birth–death tree priors, and the Felsenstein tree likelihood [Fel81]. The tree likelihood implementation is optimised and makes use of the high-performance library BEAGLE [Ayr+12]. The source code for my fork is freely available under version 3 of the GNU General Public Licence.[1]

To evaluate the performance of HMC relative to MCMC, I simulated ten 32-taxa, ten 64-taxa, and ten 128-taxa datasets under the constant size coalescent and HKY substitution model with a strict molecular clock ($n = 2048$, $\theta = 0.5$, $\mu = 0.15$, $\kappa = 8.0$, $\boldsymbol{\pi} = [0.35, 0.30, 0.20, 0.15]$). I fixed the tree topology and model parameters to the truth and performed inference on the node heights using both MCMC and HMC. The MCMC analysis used a auto-optimising scale operator on the root height and a uniform operator on all other node heights. The stepsize ϵ for HMC was auto-optimised during the analysis and nine different combinations were tried for $(L, \alpha) \in \{2, 4, 8\} \times \{0.33, 0.66, 1.0\}$. For optimal performance, the mass matrix $\mathbf{M}$ was set to the inverse of the covariance matrix for each dataset [Nea11]. I measured the performance of each algorithm as the minimum ESS of all of the dimensions divided by the runtime, which is a theoretically-justifiable indicator of overall performance [Tho10].

Results and Discussion

Overall, HMC did not show the improved performance relative to MCMC expected with increasing dimensionality (Figure 1). HMC showed better performance than MCMC only for some 32-taxa datasets. For 32 taxa, HMC was consistently either at least as efficient or at most as efficient as MCMC regardless of the choice of L and α. The choice of $\alpha = 0.66$ did tend to be correlated with better performance, probably because the partial momentum refreshment helps to avoid random-walk behaviour (Figure 2).

A basic analysis in the style of Neal [Nea11] suggests that, theoretically, HMC will scale worse than MCMC in the worst case. Although it seems that practice is consistent with theory, it is important to note that this analysis makes a number of assumptions that are unlikely to hold true in general. While it may be the case that a faithful implementation of HMC will not outperform MCMC, this does not rule out the possibility that some modification of HMC will demonstrate better performance, particularly those taking advantage of efficient numerical approximations (e.g., [CFG14]). Finally, given that this application of HMC depends on the gradient of several non-trivial probability density functions, it is possible that there are still bugs in my implementation affecting my results. The HMC algorithm is very robust in the sense that errors in numerical integration will not prevent it from converging to the target distribution. However, this property makes it very difficult to debug, as problems do not affect correctness but only efficiency.

Although this specific project does not deliver a positive result, my gradient implementations provide the foundation for developing intelligent proposal distributions for Bayesian evolutionary analysis using MCMC.

[1] `https://github.com/armanbilge/B3/tree/hamilton`

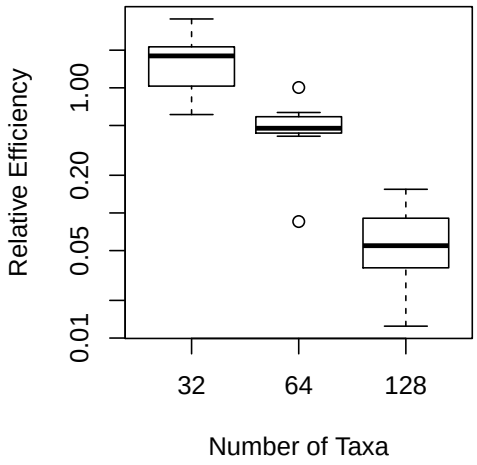

Figure 1: The efficiency of HMC relative to MCMC for the thirty simulated datasets. The best-performing combination of L and α for a dataset was taken as the representative for HMC performance.

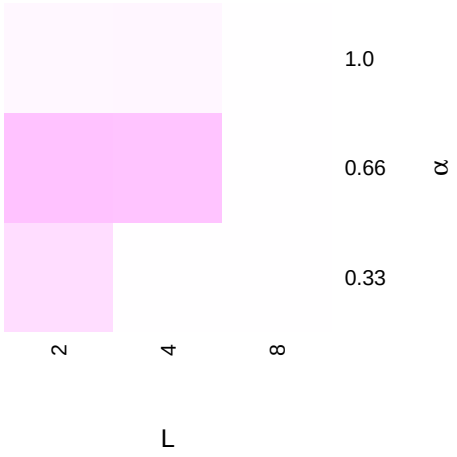

Figure 2: The efficiency of HMC relative to MCMC for various values of L and α for a particular 32-taxa dataset. Scale is from more efficient in magenta to less efficient in blue through equal efficiency in white.

References

[Ayr+12] Daniel L. Ayres et al. 'BEAGLE: An Application Programming Interface and High-Performance Computing Library for Statistical Phylogenetics'. In: *Systematic Biology* 61.1 (2012), pp. 170–173. DOI: 10.1093/sysbio/syr100.

[Bou+14] Remco Bouckaert et al. 'BEAST 2: A Software Platform for Bayesian Evolutionary Analysis'. In: *PLoS Comput Biol* 10.4 (2014), e1003537. DOI: 10.1371/journal.pcbi.1003537.

[CFG14] Tianqi Chen, Emily B. Fox and Carlos Guestrin. 'Stochastic Gradient Hamiltonian Monte Carlo'. In: (2014). URL: http://arxiv.org/abs/1402.4102.

[Dru+12] Alexei J. Drummond et al. 'Bayesian phylogenetics with BEAUti and the BEAST 1.7'. In: *Mol Biol Evol* 29.8 (2012), pp. 1969–1973. DOI: 10.1093/molbev/mss075.

[Dua+87] Simon Duane et al. 'Hybrid Monte Carlo'. In: *Physics Letters B* 195.2 (1987), pp. 216–222. DOI: 10.1016/0370-2693(87)91197-X.

[Fel81] Joseph Felsenstein. 'Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach'. In: *J Mol Evol* 17.6 (1981), pp. 368–376. DOI: 10.1007/BF01734359.

[Met+53] Nicholas Metropolis et al. 'Equation of State Calculations by Fast Computing Machines'. In: *J Chem Phys* 21.6 (1953), pp. 1087–1092. DOI: 10.1063/1.1699114.

[Nea11] Radford M. Neal. 'MCMC Using Hamiltonian Dynamics'. In: *Handbook of Markov Chain Monte Carlo*. Boca Raton, Florida: Chapman and Hall/CRC, 2011, pp. 113–162. URL: http://www.mcmchandbook.net/HandbookChapter5.pdf.

[Ron+12] Fredrik Ronquist et al. 'MrBayes 3.2: Efficient Bayesian Phylogenetic Inference and Model Choice Across a Large Model Space'. In: *Syst Biol* 61.3 (2012), pp. 539–542. DOI: 10.1093/sysbio/sys029.

[Tho10] Madeleine B. Thompson. *Graphical Comparison of MCMC Performance*. Tech. rep. 1010. Department of Statistical Sciences, University of Toronto, Nov. 2010. URL: http://www.utstat.toronto.edu/wordpress/WSFiles/technicalreports/1010.pdf.