Biostats 285: Advanced Bayesian Computing **Spring 2021**

Lecture 4: April 27

Lecturer: Andrew Holbrook *Scribes: Evan Becker*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

4.1 Introduction

These notes summarize the paper, "Generalizing Hamiltonian Monte Carlo with Neural Networks" (Levy et al., 2018). This paper introduces a method for optimizing Markov chain Monte Carlo (MCMC) transition kernels using deep neural networks, as measured by expected squared jump distance (a proxy for mixing speed). The algorithm can be considered a generalization of Hamiltonian Monte Carlo (HMC), and greatly improves on the vanilla algorithm's performance, especially when sampling multi-modal distributions.

4.1.1 Review of HMC

Given our target distribution $p(x)$, where $x \in \mathbb{R}^n$, we assume that there is a potential energy function $U(x)$ such that $p(x) \propto \exp(-U(x))$. We also assume a momentum vector $v \in \mathbb{R}^n$ with kinetic energy function $K(v) = \frac{v^T M^{-1} v}{2}$ such that $p(v) \propto \exp(-K(v))$. In this case the authors assume the mass matrix is identity. The authors use the leapfrog integrator, whose update steps are given as:

$$v^{\frac{1}{2}} = v - \frac{\epsilon}{2}\partial_x U(x); \quad x' = x + \epsilon v^{\frac{1}{2}}; \quad v' = v - \frac{\epsilon}{2}\partial_x U(x') \tag{4.1}$$

The authors use $\xi \triangleq (x, v)$ to denote the extended state vector, so that the joint probability of position and momentum can be written as

$$p(\xi) = p(x)p(v). \tag{4.2}$$

The authors use the operator $\mathbf{L}\xi \triangleq (x', v')$ to denote this entire leapfrog operation, and introduce another operator, $\mathbf{F}\xi \triangleq (x, -v)$, to denote the momentum flip operation. Chaining these two operators together gives us $\mathbf{FL}\xi = (x', -v')$. In vanilla HMC the volume-preserving properties of our operators ensure the determinant of the jacobian, $\left|\frac{\partial[\mathbf{FL}\xi]}{\partial \xi^T}\right|$, is 1. However, the generalized version of HMC introduced in this paper will have non-trivial jacobian determinants, and so we write the full acceptance formula below (this is just Metropolis-Hastings-Green):

$$A(\mathbf{FL}\xi|\xi) = \min\left(1, \frac{p(\mathbf{FL}\xi)}{p(\xi)}\left|\frac{\partial[\mathbf{FL}\xi]}{\partial \xi^T}\right|\right) \tag{4.3}$$

4.2 L2HMC: Training MCMC Samplers

While HMC is a powerful algorithm, it has a hard time sampling multi-modal distributions, and can only change energy levels via random walk (can think of this as randomly knocking our sampler into different orbits). The authors first list the desired improvements of the new algorithm as:

- fast mixing
- fast burn-in
- mixing across energy levels
- mixing between modes

They also note that this extension of HMC needs to maintain the following properties in order to retain the benefits of the original HMC algorithm:

- The leapfrog operator, $\mathbf{L}\xi$, must be invertible
- The determinant of the jacobian must be at least tractible

4.2.1 State Space

We first extend our state space by introducing a random binary direction variable $d \in \{-1, 1\}$ (drawn from uniform distribution). Our new state vector becomes $\xi \triangleq (x, v, d)$, and joint probability becomes $p(\xi) = p(x)p(v)p(d)$.
In order to make the leapfrog update more expressive, the authors define a random a mask $m^t \in \{0, 1\}^n$ to partition our position vector in preparation for two separate update steps. m^t is drawn uniformly from all combinations of masks where exactly half the entries are 1 and half are 0. We use both the mask m^t and its complement $\bar{m}^t \triangleq \mathbb{1} - m^t$.
Lastly, the authors use $\zeta \subset \xi$ to denote a subset of our full state vector.

4.2.2 Update Steps

The new algorithm introduces three new function types which increase the capability of our leapfrog update. $T(\zeta)$ is always a translation, $S(\zeta)$ scales the current state, $Q(\zeta)$ scales the derivative. When these are all zero the updates reduce to standard HMC.
For a single time step and direction d=1, the four components of our update are:

$$v' = v \odot \exp(\frac{\epsilon}{2} S_v(\zeta_1)) - \frac{\epsilon}{2}(\partial_x U(x) \odot \exp(\epsilon Q_v(\zeta_1)) + T_v(\zeta_1)); \quad \zeta_1 \triangleq (x, \partial_x U(x), t) \tag{4.4}$$

$$x' = x_{\bar{m}^t} + m^t \odot [x \odot \exp(\epsilon S_x(\zeta_2)) + \epsilon(v' \odot \exp(\epsilon Q_x(\zeta_2)) + T_x(\zeta_2))]; \quad \zeta_2 \triangleq (x_{\bar{m}^t}, v, t) \tag{4.5}$$

$$x'' = x'_{m^t} + \bar{m}^t \odot [x' \odot \exp(\epsilon S_x(\zeta_3)) + \epsilon(v' \odot (\epsilon Q_x(\zeta_3)) + T_x(\zeta_3))]; \quad \zeta_3 \triangleq (x'_{m^t}, v, t) \tag{4.6}$$

$$v'' = v' \odot \exp(\frac{\epsilon}{2} S_v(\zeta_4)) - \frac{\epsilon}{2}(\partial_x U(x'') \odot \exp(\epsilon Q_v(\zeta_4)) + T_v(\zeta_4)); \quad \zeta_4 \triangleq (x'', \partial_x U(x''), t) \tag{4.7}$$

Tractible determinants of the jacobian can be calculated for each substep as follows:

$$|J_1| = \exp(\frac{\epsilon}{2} \mathbb{1} \cdot S_v(\zeta_1)) \tag{4.8}$$

$$|J_2| = \exp(\epsilon m^t \cdot S_x(\zeta_2)) \tag{4.9}$$

$$|J_3| = \exp(\epsilon \bar{m}^t \cdot S_x(\zeta_3)) \tag{4.10}$$

$$|J_4| = \exp(\frac{\epsilon}{2} \mathbb{1} \cdot S_v(\zeta_4)) \tag{4.11}$$

If the direcion $d = -1$, the update steps are essentially inverted. Running the leapfrog operator for M steps with a flip operator at the end produces a combined jacobian determinant as follows:

$$\log \left| \frac{\partial [\mathbf{FL}_\theta \xi]}{\partial \xi^T} \right| = d \sum_{t \leq M} \left[\frac{\epsilon}{2} \mathbb{1} \cdot S_v(\zeta_1^t) + \epsilon m^t \cdot S_x(\zeta_2^t) + \epsilon \bar{m}^t \cdot S_x(\zeta_3^t) + \frac{\epsilon}{2} \mathbb{1} \cdot S_v(\zeta_4^t) \right] \tag{4.12}$$

4.2.3 MCMC Transitions

Sampling consists of the following two step sequence: first a leapfrog update and direction flip, followed by a resampling ($\mathbf{R}\xi$) of the momentum and direction variables ($v' \sim \mathcal{N}(0, I)$, $d' \sim \mathcal{U}(\{-1, 1\})$).

1. $\xi' = \mathbf{FL}_\theta \xi$ with probability $A(\mathbf{FL}_\theta \xi | \xi)$, otherwise $\xi' = \xi$.
2. $\xi' = \mathbf{R}\xi$

4.2.4 Loss and Training

Q, S, and T, are implemented using multi-layer perceptrons (MLP) with shared weights. The authors aim to train the parameters θ of these functions such that they maximize the expected squared jump distance (which is equivalent to minimizing lag-one autocorrelation)(Pasarica and Gelman, 2003). Expected squared jump distance is defined as follows:

$$\mathbb{E}_{\xi \sim p(\xi)} [\delta(\xi', \xi) A(\xi', \xi)]; \quad \delta(\xi', \xi) \triangleq ||x - x'||_2^2 \tag{4.13}$$

The authors modify this objective to encourage mixing across the entire state space (otherwise expected jumps could be large while not sampling at all from certain regions). This modification results in the loss function:

$$\ell_\lambda(\xi, \xi', A(\xi'|\xi)) = \frac{\lambda^2}{\delta(\xi, \xi') A(\xi, \xi')} - \frac{\delta(\xi, \xi') A(\xi, \xi')}{\lambda^2} \tag{4.14}$$

Where λ is a scale parameter that captures the length scale of the problem. Finally, the authors also consider this loss over not just the target distribution, but the initial distribution ,π_0, to encourage a faster burn-in (technically they use an initial join distribution $q(\xi) = \pi_0(x) \mathcal{N}(v; 0, I) p(d)$).

$$\mathcal{L}(\theta) \triangleq \mathbb{E}_{p(\xi)} [\ell_\lambda(\xi', \mathbf{FL}_\theta \xi, A(\mathbf{FL}_\theta \xi | \xi))] + \lambda_b \mathbb{E}_{q(\xi)} [\ell_\lambda(\xi', \mathbf{FL}_\theta \xi, A(\mathbf{FL}_\theta \xi | \xi))] \tag{4.15}$$

The psuedocode for training the parameters of our leapfrog operator is provided here:

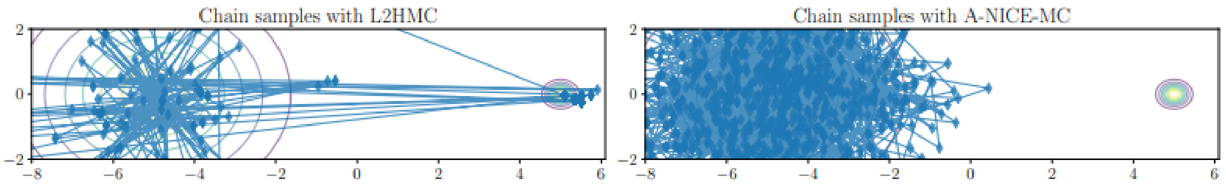
Algorithm 1 Training L2HMC

```
1: for T iterations do
2:     Sample a minibatch $\{\xi_q^{(i)}\}_{i \leq N}$ from initial distribution $q(\xi)$
3:     $\mathcal{L} \leftarrow 0$
4:     for i=1 to N do
5:         $\xi_p^{(i)} \leftarrow \mathbf{R}\xi_p^{(i)}$
6:         $\mathcal{L} \leftarrow \mathcal{L} + \ell_\lambda(\xi_p', \mathbf{FL}_\theta \xi_p, A(\mathbf{FL}_\theta \xi_p | \xi_p)) + \lambda_b \ell_\lambda(\xi_q', \mathbf{FL}_\theta \xi_q, A(\mathbf{FL}_\theta \xi_q | \xi_q))$
7:         $\xi_p^{(i)} \leftarrow \mathbf{FL}_\theta \xi_p^{(i)}$ with probability $A(\mathbf{FL}_\theta \xi_p^{(i)} | \xi_p^{(i)})$
8:     end for
9:     $\theta \leftarrow \theta - \alpha_t \nabla_\theta \mathcal{L}$
10: end for
```

4.3 Brief Summary of Results

The authors test the L2HMC sampler on a variety of challenging energy functions and compare it to common versions of HMC. Detailed metrics such as autocorrelation and expected sample size are presented in the full paper. Once particularly illustrative result is a figure comparing samples of L2HMC and another HMC algorithm (A-NICE-MC) from a mixture of gaussians distribution, and is provided here in figure 4.1.

Figure 4.1: L2HMC can mix between modes for a MoG with different variances, contrary to A-NICE-MC



References

Levy, D., M. D. Hoffman, and J. Sohl-Dickstein. 2018. Generalizing hamiltonian monte carlo with neural networks .

Pasarica, C. and A. Gelman. 2003. Adaptively scaling the metropolis algorithm using expected squared jumped distance. Tech. rep.