NEURAL TRANSFORMATIONS FOR EFFICIENT TOPOLOGICAL MIXING

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a generalized version of the L2HMC algorithm (Lévy et al., 2018), and evaluate its ability to sample from different topologies in a two-dimensional lattice gauge theory. In particular, we demonstrate that our model is able to successfully mix between modes of different topology, significantly reducing the computational cost required to generate independent gauge configurations.

1 NOTATION

For a random variable x, we write $x \sim p(x)$, and say that x is distributed according to $p(x)$.

2 INTRODUCTION

TODO: Complete introduction

3 BACKGROUND

3.1 HAMILTONIAN MONTE CARLO

The Hamiltonian Monte Carlo (HMC) algorithm is a widely used technique that allows us to sample from an analytically known target distribution $p(x)$ by constructing a chain of states $\{x^{(0)}, x^{(1)}, \ldots, x^{(n)}\}$, such that $x^{(n)} \sim p(x)$ in the limit $n \to \infty$. For our purposes, we assume that our target distribution can be expressed as a Boltzmann distribution, $p(x) = \frac{1}{Z} e^{-S[x]} \propto e^{-S[x]}$, where $S[x]$ is the *action* of our theory. In this case, HMC begins by augmenting the state space with a fictitious momentum variable v, normally distributed independently of x, i.e. $v \sim \mathcal{N}(0, \mathbb{1})$. Our joint distribution can then be written as

$$p(x, v) = p(x) \cdot p(v) \propto e^{-S[x]} \cdot e^{-\frac{1}{2} v^T v} = e^{-\mathcal{H}(x,v)} \tag{1}$$

where $\mathcal{H}(x, v)$ is the Hamiltonian of the joint (x, v) system. Notably, this system obeys Hamilton's equations

$$\dot{x} = \frac{\partial \mathcal{H}}{\partial v}, \quad \dot{v} = -\frac{\partial \mathcal{H}}{\partial x} \tag{2}$$

which can be integrated using the *leapfrog integrator* along iso-probability contours defined by $\mathcal{H} = \text{const}$. Explicitly, for a step size ε and initial state $\xi = (x, v)$, the leapfrog integrator generates a proposal configuration $\xi' \equiv (x', v')$ by performing the following series of updates:

1. Half-step momentum update, $v(t) \to v(t + \frac{\varepsilon}{2})$:
$$v^{1/2} \equiv v\Big(t + \frac{\varepsilon}{2}\Big) = v - \frac{\varepsilon}{2} \partial_x S(x) \tag{3}$$
2. Full-step position update $x(t) \to x(t + \varepsilon)$:
$$x' \equiv x(t + \varepsilon) = x + \varepsilon v^{1/2} \tag{4}$$
3. Half-step momentum update, $v\big(t + \frac{\varepsilon}{2}\big) \to v(t + \varepsilon)$:
$$v' \equiv v(t + \varepsilon) = v^{1/2} - \frac{\varepsilon}{2} \partial_x S(x') \tag{5}$$

We can then construct a complete *trajectory* of length $\lambda = \varepsilon \cdot N_{\mathrm{LF}}$ by performing N_{LF} leapfrog steps in sequence. At the end of our trajectory, we either accept or reject the proposal configuration according to the Metropolis-Hastings acceptance criteria,

$$x_{i+1} = \begin{cases} x' & \text{with probability } A(\xi'|\xi) \\ x & \text{with probability } (1 - A(\xi'|\xi)). \end{cases} \tag{6}$$

where

$$A(\xi'|\xi) = \min\left\{1, \frac{p(\xi')}{p(\xi)}\left|\frac{\partial \xi'}{\partial \xi^T}\right|\right\}. \tag{7}$$

The leapfrog integrator is known to be symplectic (conserves energy), so the Jacobian factor reduces to $\left|\frac{\partial \xi'}{\partial \xi^T}\right| = 1$.

3.2 GENERALIZING THE LEAPFROG INTEGRATOR: L2HMC

In (Lévy et al., 2018), the authors propose the L2HMC ("Learning to Hamiltonian Monte Carlo") algorithm, and demonstrate its ability to outperform traditional Hamiltonian Monte Carlo (HMC) on a variety of two-dimensional target distributions. For example, the trained L2HMC sampler is shown to be capable of exploring regions of phase space which are typically inaccessible with traditional HMC. Additionally, they show that the trained sampler is efficient at mixing between modes of a multi-modal target distribution, a feature which is highly desirable for MCMC simulations of lattice gauge theory.

We denote a complete state by $\xi = (x, v, d)$ with target distribution $p(\xi) = p(x, v, d) = p(x) \cdot p(v) \cdot p(d)$. Here we've introduced a (uniformly drawn) binary direction variable $d \in \{-, +\}$ that determines the "direction" of our update, and is distributed independently of both x and v. The key modification of the L2HMC algorithm is the introduction of six auxiliary functions s_i, t_i, q_i for $i = x, v$ into the leapfrog updates, which are parameterized by weights θ in a neural network.

For simplicity, we consider the forward $d = +1$ direction, and introduce the notation:

$$\begin{aligned} v'_k &\equiv \Gamma^+_k(v_k; \zeta_{v_k}) \\ &= v_k \odot \exp\left(\tfrac{\varepsilon}{2} s^k_v(\zeta_{v_k})\right) - \tfrac{\varepsilon}{2}\left[\partial_x S(x_k) \odot \exp\left(\varepsilon q^k_v(\zeta_{v_k})\right) + t^k_v(\zeta_{v_k})\right] \end{aligned} \tag{8}$$

$$\begin{aligned} x'_k &\equiv \Lambda^{\pm}(x_k; \zeta_{x_k}) \\ &= x_k \odot \exp(\varepsilon s^k_x(\zeta_{x_k})) + \varepsilon\left[v'_k \odot \exp(\varepsilon q^k_x(\zeta_{x_k})) + t^k_x(\zeta_{x_k})\right] \end{aligned} \tag{9}$$

where (1.) $\zeta_{v_k} = (x_k, \partial_x S(x_k), \tau(k))$, and $\zeta_{x_k} = (x_k, v_k, \tau(k))$ are subsets of the augmented space independent of the variable being updated (v, x respectively), (2.) $\tau(k) = \left[\cos\frac{2\pi k}{N_{\mathrm{LF}}}, \sin\frac{2\pi k}{N_{\mathrm{LF}}}\right]$, $k = 0, 1, \ldots, N_{\mathrm{LF}}$, is a discrete time variable parameterizing our trajectory, and (3.) we indicate the forward $d = +1$ direction by the $+$ superscript on Γ^+, Λ^+ respectively.

This allows us to write the complete leapfrog update (in the forward $d = +1$ direction) as:

1. Half-step momentum update:
$$v'_k = \Gamma^+_k(v_k; \zeta_{v_k}) \tag{10}$$
2. Half-step position update:
$$x'_k = \bar{m}^t \odot x_k + m^t \odot \Lambda^+_k(x_k; \zeta_{x_k}) \tag{11}$$
3. Half-step position update:
$$x''_k = \bar{m}^t \odot \Lambda^+_k(x'_k; \zeta_{x'_k}) + m^t \odot x'_k \tag{12}$$
4. Half-step momentum update:
$$v''_k = \Gamma^+(v'_k; \zeta_{v'_k}) \tag{13}$$

Note that in order to keep our leapfrog update reversible, we've split the x update into two sub-updates by introducing a binary mask $m^t = m^t \odot \mathbb{1} + \bar{m}^t \odot \mathbb{1}$ that updates half of the components of x sequentially, as shown in Figure 1. As in HMC, we form a complete trajectory by performing N_{LF}

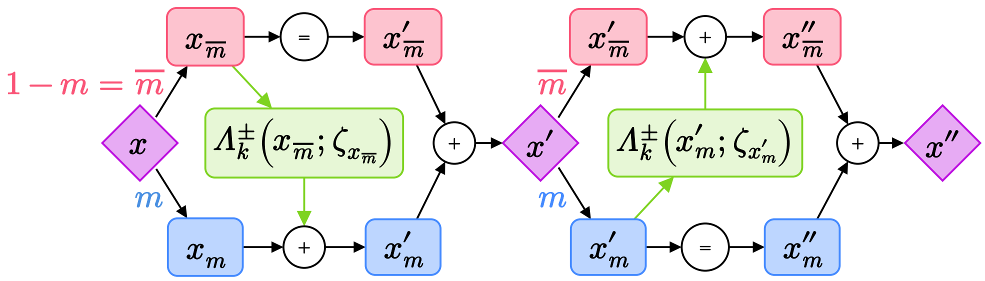

Figure 1: Illustration of the split x update.

leapfrog steps sequentially and use the Metropolis-Hastings criteria (Equation 7) to either accept or reject the proposed configurations. However, unlike in the expression for HMC, we must take into account the Jacobians of each of the transformations which are easily computed to give

$$\left|\frac{\partial v'_k}{\partial v_k}\right| = \exp\left(\frac{\varepsilon}{2} s_v^k(\zeta_{v_k})\right), \quad \left|\frac{\partial x'_k}{\partial x_k}\right| = \exp\left(\varepsilon s_x^k(\zeta_{x_k})\right) \tag{14}$$

Next, we introduce a loss function

$$\mathcal{L}_\theta(\xi, \xi', A(\xi'|\xi)) = -\frac{\delta(\xi, \xi')}{a^2} \tag{15}$$

where $\delta(\xi, \xi')$ is a suitably chosen *metric function*, and a is a scaling factor.

Acknowledgments

This research used resources of the argonne leadership computing facility, which is a doe office of science user facility supported under contract DE_AC02–06CH11357.This work describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the work do not necessarily represent the views of the u.s. doe or the united states government.

References

Daniel Lévy, M. Hoffman, and Jascha Sohl-Dickstein. Generalizing hamiltonian monte carlo with neural networks. *ArXiv*, abs/1711.09268, 2018.

A Appendix