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 INTRODUCTION

TODO: Complete introduction

2 OUR CONTRIBUTIONS

Our main contributions are as follows:

1. We propose a generalized version of the L2HMC algorithm that allows for different networks with different step sizes for each distinct leapfrog step. We represent this generalization by carrying a discrete index k through the augmented leapfrog equations, indicating that these functions are allowed to vary.
2. We introduce a topological charge metric $\delta_{\mathcal{Q}_{\mathbb{R}}}(\xi', \xi)$ defined as the squared difference between the initial and proposed configurations $\delta_{\mathcal{Q}_{\mathbb{R}}}(\xi', \xi) = (\mathcal{Q}'_{\mathbb{R}} - \mathcal{Q}_{\mathbb{R}})^2$
3. We include an *annealing schedule* $\{\beta\}_{t=0}^{N_{\mathrm{train}}}$
4. We use the *integrated autocorrelation time* of the topological charge $\tau_{\mathrm{int}}^{\mathcal{Q}}$ as a metric for determining the efficiency of our trained sampler.
5. We compare our results to traditional HMC across a variety of trajectory lengths and inverse coupling constants, and show that our trained model consistently outperforms traditional HMC.

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, $p(x) = \frac{1}{\mathcal{Z}} e^{-S(x)} \propto e^{-S(x)}$, 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: $v^{1/2} \equiv v\big(t + \frac{\varepsilon}{2}\big) = v - \frac{\varepsilon}{2}\partial_x S(x)$
2. Full-step position update: $x' \equiv x(t + \varepsilon) = x + \varepsilon v^{1/2}$
3. Half-step momentum update: $v' \equiv v(t + \varepsilon) = v^{1/2} - \frac{\varepsilon}{2}\partial_x S(x')$

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,

$$x_{i+1} = \begin{cases} x' & \text{with probability } A(\xi'|\xi) \\ x & \text{with probability } (1 - A(\xi'|\xi)), \end{cases} \quad \text{where} \quad A(\xi'|\xi) = \min\left\{1, \frac{p(\xi')}{p(\xi)}\left|\frac{\partial \xi'}{\partial \xi^T}\right|\right\}. \tag{3}$$

The generic 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

Notable changes:

1. Compared to the original implementation, we carry throughought the updates a discrete index $k = 0, 1, \ldots, N_{\mathrm{LF}}$, parameterizing the current leapfrog step.
   In doing so, we are free to consider the case where we use different update functions (Equation 5, Equation 6), with completely independent step sizes ε_j^k, for each of the $j = x, v$ updates.
2. We propose a modified loss function, defined in terms of the topological charge metric,
$$\mathcal{L}_\theta(\xi', \xi, A(\xi'|\xi)) = \frac{-A(\xi'|\xi) \cdot \delta_{\mathcal{Q}_\mathbb{R}}(\xi', \xi)}{a^2} \tag{4}$$
   where $\delta_{\mathcal{Q}_\mathbb{R}}(\xi', \xi) \equiv \left(\mathcal{Q}'_\mathbb{R} - \mathcal{Q}_\mathbb{R}\right)^2$ where $\mathcal{Q}_\mathbb{R}$ is the *topological charge*, defined in Section 5

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:

$$v'_k \equiv \Gamma_k^+(v_k; \zeta_{v_k}) = v_k \odot \exp\left(\frac{\varepsilon_v^k}{2} s_v^k(\zeta_{v_k})\right) - \frac{\varepsilon_v^k}{2}\left[\partial_x S(x_k) \odot \exp\left(\varepsilon_v^k q_v^k(\zeta_{v_k})\right) + t_v^k(\zeta_{v_k})\right], \tag{5}$$

$$x'_k \equiv \Lambda^+(x_k; \zeta_{x_k}) = x_k \odot \exp(\varepsilon_x^k s_x^k(\zeta_{x_k})) + \varepsilon_x^k\left[v'_k \odot \exp(\varepsilon_x^k q_x^k(\zeta_{x_k})) + t_x^k(\zeta_{x_k})\right] \tag{6}$$

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^+, and Λ^+.

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

1. Half-step momentum update: $v'_k = \Gamma^+_k(v_k; \zeta_{v_k})$
2. Full-step half-position update[1]: $x'_k = \bar{m}^t \odot x_k + m^t \odot \Lambda^+_k(x_k; \zeta_{x_k})$
3. Full-step half-position update: $x''_k = \bar{m}^t \odot \Lambda^+_k(x'_k; \zeta_{x'_k}) + m^t \odot x'_k$
4. Half-step momentum update: $v''_k = \Gamma^+(v'_k; \zeta_{v'_k})$

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.

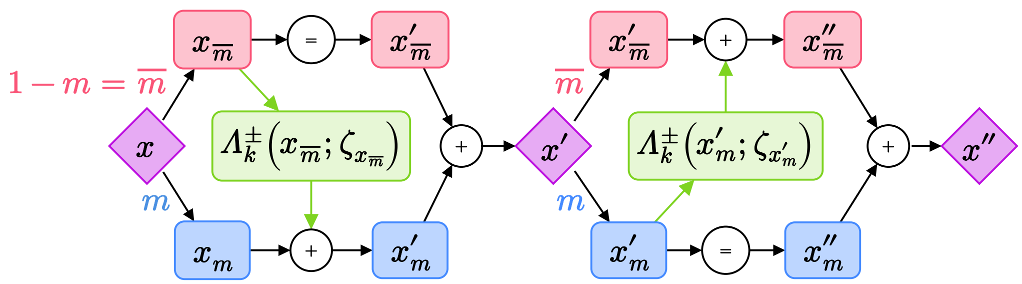

Figure 1: Illustration of the split x update.

As in HMC, we form a complete trajectory by performing N_{LF} leapfrog steps in sequence, followed by a Metropolis-Hastings accept/reject step as described in Equation 3. However, unlike in the expression for HMC, we must take into account the overall Jacobian factor from the update $\xi \to \xi'$, which can be easily computed as

$$\left|\frac{\partial v'_k}{\partial v_k}\right| = \exp\left(\frac{\varepsilon_k}{2} s^k_v(\zeta_{v_k})\right), \qquad \left|\frac{\partial x'_k}{\partial x_k}\right| = \exp\left(\varepsilon_k s^k_x(\zeta_{x_k})\right). \tag{7}$$

In order to perform the updates in the generalized leapfrog integrator, we need to evaluate each of the functions s, t, q. Without loss of generality[2], we temporarily ignore the discrete leapfrog index k, and restrict our attention to the s_x, t_x, q_x functions used in the x update equation, Equation 6.

Each of the s_x, t_x, q_x functions takes as input $\zeta_x = (x, v, \tau)$, with $x \in \mathbb{R}^n$, $v \in \mathbb{R}^n$, and $\tau \in \mathbb{R}^2$. The network splits these inputs and constructs the following intermediate variable (where σ denotes an arbitrary activation function)

$$z_1 = \sigma(w_x^T x + w_v^T v + w_\tau^T \tau + b). \tag{8}$$

This intermediate variable z_1 is then passed through another series of fully-connected layers,

$$z_n = \sigma(w_n^T z_{n-1} + b_n),\ z_{n-1} = \sigma(w_{n-2}^T z_{n-2} + b_{n-2}),\ \ldots,\ z_2 = \sigma(w_2^T z_1 + b_2). \tag{9}$$

The network outputs s_x, t_x, q_x are then defined in terms of this final hidden variable z_n as

$$s_x(\zeta_x) = \alpha_s \tanh(w_s z_n + b_s), \quad t_x(\zeta_x) = w_t^T z_n + b_t, \quad q_x(\zeta_x) = \alpha_q \tanh(w_q z_n + b_q) \tag{10}$$

where α_s, and α_q are trainable scaling factors. The only requirement on the details of the network is that the dimensionality of the outputs s_x, t_x, q_x match the dimensionality of our physical variables $x, v \in \mathbb{R}^n$.

Next, we introduce a loss function

$$\mathcal{L}_\theta(\xi, \xi', A(\xi'|\xi)) = -\frac{A(\xi'|\xi) \cdot \delta(\xi, \xi')}{a^2} \tag{11}$$

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

[1] By this we mean we are performing a complete update step that only updates half of the components of x determined by the mask m^t and its complement $\bar{m}^t$.

[2] Because we maintain a separate network with identical architecture for evaluating the s, t, q functions in the momentum update equation, Equation 5, the procedure is identical

4 Annealing Schedule

To help our sampler overcome the large energy barriers between isolated modes, we introduce an *annealing schedule*, during the training phase

$$\{\beta_t\}_{t=0}^{N} = \{\beta_0, \beta_1, \ldots, \beta_{N-1}, \beta_N\}, \tag{12}$$

where $\beta_0 < \beta_1 < \cdots < \beta_N \equiv 1$, $\beta_{t+1} - \beta_t \ll 1$, N denotes the total number of training steps to be performed. Note that we are free to vary β during the initial training phase as long as we recover the true distribution with $\beta \equiv 1$ at the end of training and evaluate our trained model without this factor. Explicitly, for $\beta_t < 1$ this rescaling factor helps to reduce the height of the energy barriers, making it easier for our sampler to explore previously inaccessible regions of the phase space. In terms of this additional annealing schedule, our target distribution picks up an additional index t to represent our progress through the training phase, which can be written explicitly as

$$p_t(x) \propto e^{-\beta_t S(x)} \tag{13}$$

for $t = 0, 1, \ldots, N$.

5 Lattice Gauge Theory

Note: We define the lattice gauge theory in terms of the link variables $-\pi \leq \varphi_\mu(x) < \pi$, which are identified as being the *position* variable x in Section 3.2.

We consider a two-dimensional $U(1)$ lattice gauge theory defined on an $N_x \times N_t$ lattice with periodic boundary conditions. Our target distribution $p(\varphi)$ is defined in terms of the Wilson action as

$$p(\varphi) \propto e^{\beta_t S(\varphi)}, \quad \text{where} \quad S(\varphi) = \sum_P 1 - \cos(\varphi_P(x)) \tag{14}$$

where φ_P denotes the sum of the link variables around the elementary plaquette,

$$\varphi_P = \varphi_\mu(x) + \varphi_\nu(x+\hat{\mu}) - \varphi_\mu(x+\hat{\nu}) - \varphi_\nu(x) \tag{15}$$

as shown in Figure 2. Explicitly, we can write the

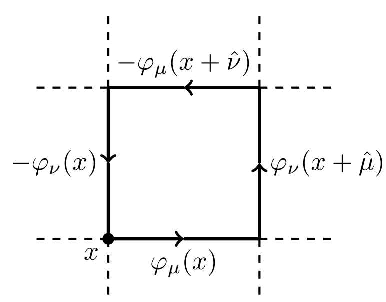

Figure 2: Elementary plaquette, P on the lattice.

$$U_\mu(x) = e^{i\varphi_\mu(x)}, \quad \varphi_\mu(x) \in [-\pi, \pi] \tag{16}$$

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