Extending Surrogate Hamiltonian Monte Carlo

Outline

- Introduction & Motivation
- Previous Work
- Background
    i. Hamiltonian Dynamics + HMC
    ii. Numerical Integration
    iii. Neural ODE
- Methods
- Results
- Demonstration

Introduction & Motivation

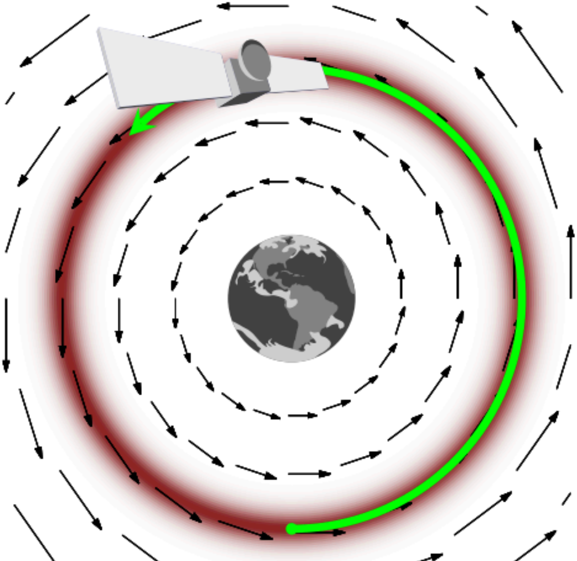
Aim to speed up aspects of (Riemannian) Hamiltonian Monte Carlo sampling by approximating computationally expensive components.

Hamiltonian Systems

The Hamiltonian, $H(q, p)$, is a function of q and p, parameters defining an object's position and velocity.

We can decompose the Hamiltonian into the following, if the Hamiltonian is separable: $H(q, p) = U(q) + K(p)$. $U(q)$ and $K(p)$ represent potential and kinetic energy of the system respectively.

Hamiltonian Systems (cont.)



Hamiltonian Systems (cont.)

The dynamics of the system can be modeled by

$$\frac{\partial H}{\partial q} = -\frac{\partial p}{\partial t}$$

$$\frac{\partial H}{\partial p} = \frac{\partial q}{\partial t}$$

Thus, the system can be evolved over time using numerical integration schemes that are symplectic.

Hamiltonian Systems (cont.)

A Hamiltonian system has key properties:

1. Time Reversibility
2. Volume Preservation
3. Conservation of the Hamiltonian

Hamiltonian Monte Carlo

Position given by parameters q, momentum p drawn from kinetic energy distribution, typically an Isotropic Gaussian.

$U(q) = -log(\pi(q))$

$K(p) = \frac{p^T p}{2}$

Computation of $\frac{\delta U(q)}{\delta q}$, $U(q)$ can be costly sometimes!

Leapfrog Integration

Assuming a separable Hamiltonian form commonly seen in Hamiltonian Monte Carlo, $H(q,p) = U(q) + \frac{p^T p}{2}$, one may use the following update rules for a given starting point in the phase space (q_0, p_0) to simulate dynamics with a fixed step size ϵ :

1. $q_{t+1} = q_t + \epsilon p_t + \frac{\epsilon^2}{2}\frac{\delta p_t}{\delta t}$
2. $p_{t+1} = p_t + \frac{\epsilon}{2}(\frac{\delta p_t}{\delta t} + \frac{\delta p_{t+1}}{\delta t})$

In the case where the Hamiltonian is not separable (RMHMC), the expression is more complex.

Neural ODE

Can be used to model continuous time dynamics. Note that if we aim to approximate a smooth function $y = f(x, \theta)$, then it is also equivalent to approximate $y = \int_{t_0}^{t_1} f^{'}(x, \theta)dt$.

1. Model $f^{'}(x, \theta)$
2. Integrate $f^{'}(x, \theta)$ to obtain $f(x, \theta)$
3. Backpropagate loss from $L(f(x, \theta), y)$ through ODE Dynamics

Contribution

Combine Neural ODEs using symplectic integration schemes with Hamiltonian NNs to approximate Hamiltonian trajectories during sampling.

Methods

Let $f(\theta, q, p)$ be a Neural Network approximating the derivatives of the Hamiltonian of a system, $\frac{\partial H}{\partial q}$ and $\frac{\partial H}{\partial p}$, or equivalently $-\frac{\partial p}{\partial t}$ and $\frac{\partial q}{\partial t}$.

1. Collect $\{(q_i^0, p_i^0)\}_{i=1}^{K}$, from a burn-in period of K samples of trajectory length L
2. Collect $\{(Q_i, P_i)\}_{i=1}^{K}$, where Q_i and P_i are vectors of length L corresponding to an L length trajectory for burn-in sample i.
3. $(\hat{Q}, \hat{P}_i) = ODESolve(f(\theta, q^0, p^0), L, \epsilon)$

Methods (cont.)

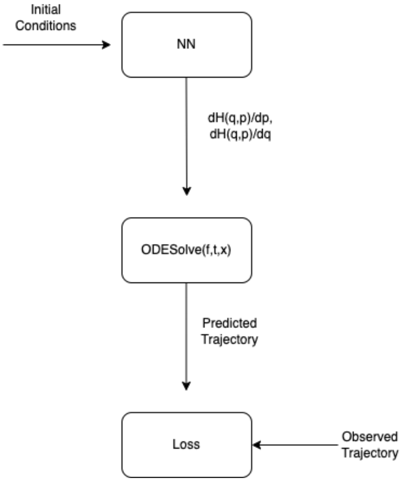


Figure 2: NNODE

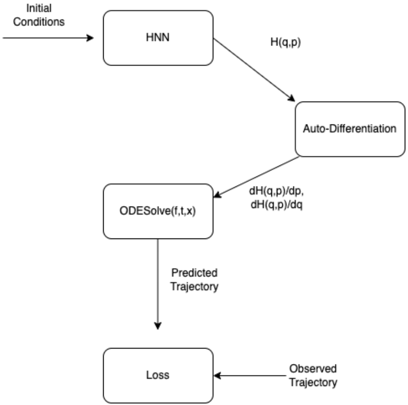


Figure 3: HNNODE