Continuous Normalizing Flow

Bangyan Liao
liaobangyan@westlake.edu.cn

Oct 2024

Outline

Outline

Forward Process

Given a neural network f parameterised by θ, the neural ode can be defined as:

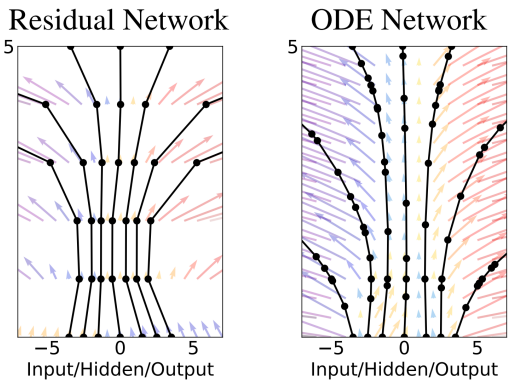
$$\frac{d\mathbf{z}(t_i)}{dt} = f_\theta(\mathbf{z}(t_i), t_i), \mathbf{z}(t_0) = \mathbf{z}_0 \quad (1)$$

Euler Discretization

$$\mathbf{z}(t_{i+1}) = \mathbf{z}(t_i) + f_\theta(\mathbf{z}(t_i), t_i)\Delta t \quad (2)$$

Implicit Discretization

$$\mathbf{z}(t_{i+1}) = \mathbf{z}(t_i) + f_\theta(\mathbf{z}(t_{i+1}), t_{i+1})\Delta t \quad (3)$$



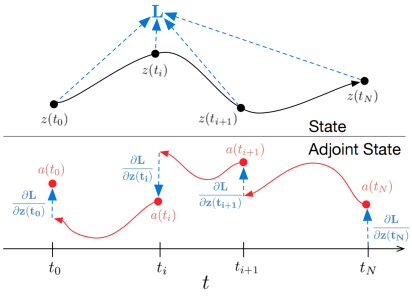
Backward Process

Given a predefine objective function $L(\mathbf{z}(t_N))$, the auto back-propagate has high memory cost. Chen et al.[a] proposes an adjoint method which solves a reverse ODE to get the $\frac{\partial L}{\partial \boldsymbol{\theta}}$.

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \int_{t_0}^{t_N} \left(-\frac{\partial L}{\partial \mathbf{z}(t)}^{\top} \frac{\partial f_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \boldsymbol{\theta}} \right) dt \qquad (4)$$

Adjoint Method

$$\frac{d\mathbf{a}(t_i)}{dt} = -\mathbf{a}(t_i)^{\top} \frac{\partial f_{\boldsymbol{\theta}}(\mathbf{z}(t_i), t_i)}{\mathbf{z}(t_i)}, \mathbf{a}(t_N) = \frac{\partial L}{\partial \mathbf{z}(t_N)} \qquad (5)$$



[a] chen2018neural.

Forward Propagation

$$\mathbf{z}(t_N) = \text{ODESolver}\left(f_{\boldsymbol{\theta}}(\mathbf{z}(t), t), \mathbf{z}(t_0), t_0, t_N\right) \tag{6}$$

Backward Propagation

$$\begin{aligned}
\mathbf{z}(t_0) &= \text{ODESolver}\left(f_{\boldsymbol{\theta}}(\mathbf{z}(t), t), \mathbf{z}(t_N), t_N, t_0\right) \\
\mathbf{a}(t_0) = \frac{\partial L}{\partial \mathbf{z}(t_0)} &= \text{ODESolver}\left(-\mathbf{a}(t)^\top \frac{\partial f_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}}, \frac{\partial L}{\partial \mathbf{z}(t_N)}, t_N, t_0\right) \\
\frac{\partial L}{\partial \boldsymbol{\theta}} &= \text{ODESolver}\left(-\mathbf{a}(t)^\top \frac{\partial f_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \boldsymbol{\theta}}, \mathbf{0}, t_N, t_0\right)
\end{aligned} \tag{7}$$

Algorithm 1 Reverse-mode derivative of an ODE initial value problem

Input: dynamics parameters θ, start time t_0, stop time t_1, final state $\mathbf{z}(t_1)$, loss gradient $\frac{\partial L}{\partial \mathbf{z}(t_1)}$

$s_0 = [\mathbf{z}(t_1), \frac{\partial L}{\partial \mathbf{z}(t_1)}, \mathbf{0}_{|\theta|}]$ ▷ Define initial augmented state

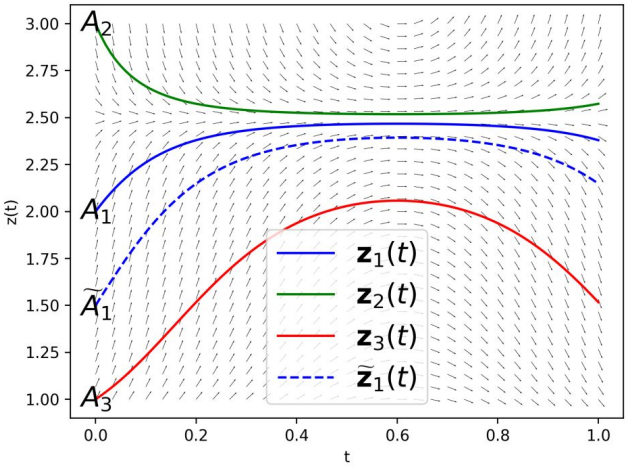
def aug_dynamics($[\mathbf{z}(t), \mathbf{a}(t), \cdot], t, \theta$): ▷ Define dynamics on augmented state

return $[f(\mathbf{z}(t), t, \theta), -\mathbf{a}(t)^\top \frac{\partial f}{\partial \mathbf{z}}, -\mathbf{a}(t)^\top \frac{\partial f}{\partial \theta}]$ ▷ Compute vector-Jacobian products

$[\mathbf{z}(t_0), \frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}] = \text{ODESolve}(s_0, \text{aug\_dynamics}, t_1, t_0, \theta)$ ▷ Solve reverse-time ODE

return $\frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}$ ▷ Return gradients

Example Neural ODE



Remark.

Neural ODEs describe a homeomorphism (flow). They preserve dimensionality. They form non-intersecting trajectories. Neural ODEs are reversible models.

Theorem (Instantaneous Change of Variables)

Let $\mathbf{x}_t$ be a finite continuous random variable with probability $\rho(\mathbf{x}_t)$ dependent on time. Let $\frac{d\mathbf{x}_t}{dt} = f_\theta(\mathbf{x}_t, t)$ be a differential equation describing a continuous-in-time transformation of $\mathbf{x}_t$. Asssuming that f is uniformly Lipschitz continuous in $\mathbf{x}$ and continuous in t, then the change in log probability also follows a differential equation,

$$\frac{d}{dt} log\ \rho(\mathbf{x}_t, t) = -\nabla \cdot f_\theta(\mathbf{x}_t, t) \tag{8}$$

Lemma (Continuity Equation)

Definition. Let $\rho(\mathbf{x}_t, t)$ denotes the evolving probabilistic measure following the differential equation $\frac{d\mathbf{x}_t}{dt} = f_\theta(\mathbf{x}_t, t)$ and $f_\theta(\mathbf{x}_t, t)$ represents the velocity field. The continuity equation describes the relationship between flow field and probabilistic measure as:

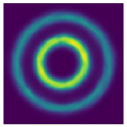
$$\frac{\partial \rho(\mathbf{x}_t, t)}{\partial t} + \nabla \cdot (f_\theta(\mathbf{x}_t, t)\rho(\mathbf{x}_t, t)) = 0 \tag{9}$$

Proof.

$$\begin{aligned}
\frac{d}{dt}\log\, \rho(\mathbf{x}_t, t) &= \frac{\nabla \rho(\mathbf{x}_t, t) \cdot \dot{\mathbf{x}_t} + \partial_t \rho(\mathbf{x}_t, t)}{\rho(\mathbf{x}_t, t)} \\
&= -\frac{\nabla \rho(\mathbf{x}_t, t) \cdot f_\theta(\mathbf{x}_t, t) - \nabla \cdot (f_\theta(\mathbf{x}_t, t)\rho(\mathbf{x}_t, t))}{\rho(\mathbf{x}_t, t)} \\
&= -\nabla \cdot f_\theta(\mathbf{x}_t, t)
\end{aligned} \tag{10}$$

Normalizing Flow

data distribution

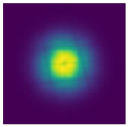


$\mathbf{x} \sim \mu$

f

f^{-1}

standard distribution

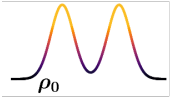


$\mathbf{z} \sim v$

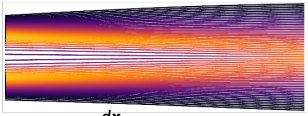
$$log\ \mu(\mathbf{x}) = log\ v(f(\mathbf{x})) + log\ det\ |\nabla f(\mathbf{x})|$$

Continuous Normalizing Flow

data distribution

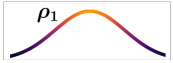


$\mathbf{x}_0 \sim \rho_0$



$$\frac{d\mathbf{x}_t}{dt} = f_\theta(\mathbf{x}_t, t)$$

standard distribution



$\mathbf{x}_1 \sim \rho_1$

$$log\ \rho_0(\mathbf{x}_0, 0) = log\ \rho_1(\mathbf{x}_1, 1) + \int_0^1 -\nabla \cdot f_\theta(\mathbf{x}_t, t)\ dt$$

Outline

FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models[1]

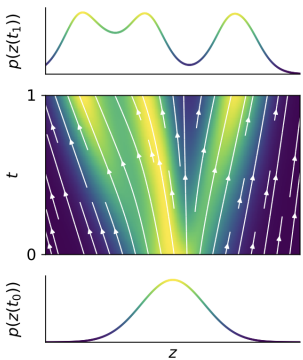
A promising class of generative models maps points from a simple distribution to a complex distribution through an invertible neural network. Likelihood-based training of these models requires restricting their architectures to allow cheap computation of Jacobian determinants. Alternatively, the Jacobian trace can be used if the transformation is specified by an ordinary differential equation. In this paper, we use Hutchinson's trace estimator to give a scalable unbiased estimate of the log-density. The result is a continuous-time invertible generative model with unbiased density estimation and one-pass sampling, while allowing unrestricted neural network architectures. We demonstrate our approach on high-dimensional density estimation, image generation, and variational inference, achieving the state-of-the-art among exact likelihood methods with efficient sampling

[1] **grathwohl2018ffjord**.

Hutchinson's trace estimator

Hutchinson's trace estimator is the unbiased Monte Carlo trace estimator defined as:

$$\begin{aligned} &\mathrm{Tr}(\mathbf{A}) = \mathbb{E}_\epsilon \left[\epsilon^\top \mathbf{A} \epsilon \right] \\ &s.t. \mathbb{E}\left[\epsilon\right] = 0, \mathrm{Cov}(\epsilon) = \mathbf{I}. \end{aligned} \tag{11}$$



How to train your neural ODE: the world of Jacobian and kinetic regularization[2]
Training neural ODEs on large datasets has not been tractable due to the necessity of allowing the adaptive numerical ODE solver to refine its step size to very small values. In practice this leads to dynamics equivalent to many hundreds or even thousands of layers. In this paper, we overcome this apparent difficulty by introducing a theoretically-grounded combination of both optimal transport and stability regularizations which encourage neural ODEs to prefer simpler dynamics out of all the dynamics that solve a problem well. Simpler dynamics lead to faster convergence and to fewer discretizations of the solver, considerably decreasing wall-clock time without loss in performance. Our approach allows us to train neural ODE-based generative models to the same performance as the unregularized dynamics, with significant reductions in training time. This brings neural ODEs closer to practical relevance in large-scale applications.

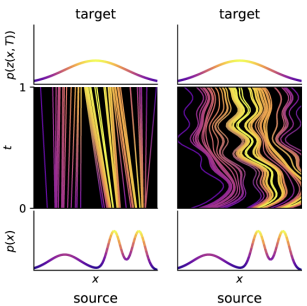
[2] **finlay2020train**.

Kinetic Energy Regularization

$$\mathcal{K}(\boldsymbol{\theta}) = \int_0^1 \|f_\theta(\mathbf{x}_t, t)\|_2^2 dt \tag{12}$$

Jacobian Norm Regularization

$$\mathcal{B}(\boldsymbol{\theta}) = \mathbb{E}_\epsilon \int_0^1 \left\| \epsilon^\top \nabla f_\theta(\mathbf{x}_t, t) \right\|_2^2 dt \tag{13}$$



(a) Optimal transport map (b) generic flow

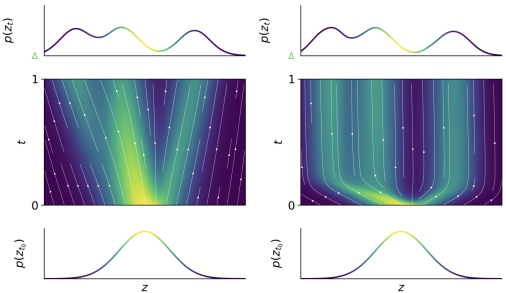
STEER: Simple Temporal Regularization For Neural ODEs[3]

Training Neural Ordinary Differential Equations (ODEs) is often computationally expensive. Indeed, computing the forward pass of such models involves solving an ODE which can become arbitrarily complex during training. Recent works have shown that regularizing the dynamics of the ODE can partially alleviate this. In this paper we propose a new regularization technique: randomly sampling the end time of the ODE during training. The proposed regularization is simple to implement, has negligible overhead and is effective across a wide variety of tasks. Further, the technique is orthogonal to several other methods proposed to regularize the dynamics of ODEs and as such can be used in conjunction with them. We show through experiments on normalizing flows, time series models and image recognition that the proposed regularization can significantly decrease training time and even improve performance over baseline models.

[3] **ghosh2020steer**.

Temporal Regularization

$$\mathbf{x}_1 = \mathbf{x}_0 + \int_0^T f_\theta(\mathbf{x}_t, t)\, dt$$
$$T \sim \text{Uniform}(1-b, 1+b), \quad b < 1 \tag{14}$$



OT-Flow: Fast and Accurate Continuous Normalizing Flows via Optimal Transport[4]

A normalizing flow is an invertible mapping between an arbitrary probability distribution and a standard normal distribution; it can be used for density estimation and statistical inference. Computing the flow follows the change of variables formula and thus requires invertibility of the mapping and an efficient way to compute the determinant of its Jacobian. To satisfy these requirements, normalizing flows typically consist of carefully chosen components. Continuous normalizing flows (CNFs) are mappings obtained by solving a neural ordinary differential equation (ODE). The neural ODE's dynamics can be chosen almost arbitrarily while ensuring invertibility. Moreover, the log-determinant of the flow's Jacobian can be obtained by integrating the trace of the dynamics' Jacobian along the flow. Our proposed OT-Flow approach tackles two critical computational challenges that limit a more widespread use of CNFs. First, OT-Flow leverages optimal transport (OT) theory to regularize the CNF and enforce straight trajectories that are easier to integrate. Second, OT-Flow features exact trace computation with time complexity equal to trace estimators used in existing CNFs. On five high-dimensional density estimation and generative modeling tasks, OT-Flow performs competitively to state-of-the-art CNFs while on average requiring one-fourth of the number of weights with an 8x speedup in training time and 24x speedup in inference.

[4] **onken2021ot**.

Potential Regularization

From the Pontragin Maximum Principle, there exists a potential function Φ such that $f_{\boldsymbol{\theta}}(\mathbf{z}(t_i), t_i) = -\nabla\Phi_{\boldsymbol{\theta}}(\mathbf{z}(t_i), t_i)$. Moreover, optimal control theory states that Φ satisfies

$$-\partial_t\Phi_{\boldsymbol{\theta}}(\mathbf{z}(t_i), t_i) + \frac{1}{2}\|\nabla\Phi_{\boldsymbol{\theta}}(\mathbf{z}(t_i), t_i)\|^2 = 0 \qquad (15)$$

Then, we can define the potential regularization

$$\mathcal{R}(\boldsymbol{\theta}) = \int_0^T \left|\partial_t\Phi_{\boldsymbol{\theta}}(\mathbf{z}(t_i), t_i) + \frac{1}{2}\|\nabla\Phi_{\boldsymbol{\theta}}(\mathbf{z}(t_i), t_i)\|^2\right| dt \quad (16)$$