"Hey, that's not an ODE": Faster ODE Adjoints with 12 Lines of Code

Patrick Kidger, Ricky T. Q. Chen, Terry Lyons

UNIVERSITY OF OXFORD

Summary

We train neural ODEs, using the adjoint method, much faster: with a median of 40% fewer function evaluations.

This is done by observing that the backward (adjoint) equations exhibit a particular structure, which we may adjust the ODE solver to exploit. This change is simple to implement: at least using `torchdiffeq`, it requires only 12 lines of code.

Background: Neural ODEs

- Differential equations with neural network vector fields.
- Use ODEs as components of a differentiable computation graph.

For example: supervised learning of the map $x \mapsto y$ via

$$z(0) = \ell_\theta^1(x) \qquad z(t) = z(0) + \int_0^t f_\theta(z(s))\,\mathrm{d}s \qquad y \approx \ell_\theta^2(z(T))$$

with $\ell_\theta^1, \ell_\theta^2$ linear, and f_θ a neural network.

Background: Adjoint equations

Neural ODEs may be backpropagated through via the adjoint equations. For a loss function L, then $\frac{\mathrm{d}L}{\mathrm{d}\theta} = a_\theta(0)$ and $\frac{\mathrm{d}L}{\mathrm{d}z(0)} = a_z(0)$, where

$$a_z(T) = \frac{\mathrm{d}L}{\mathrm{d}z(T)}, \qquad a_z(t) = a_z(T) - \int_T^t a_z(s) \cdot \frac{\partial f}{\partial z}(s, z(s), \theta)\,\mathrm{d}s,$$
$$a_\theta(T) = 0, \qquad a_\theta(t) = a_\theta(T) - \int_T^t a_z(s) \cdot \frac{\partial f}{\partial \theta}(s, z(s), \theta)\,\mathrm{d}s.$$

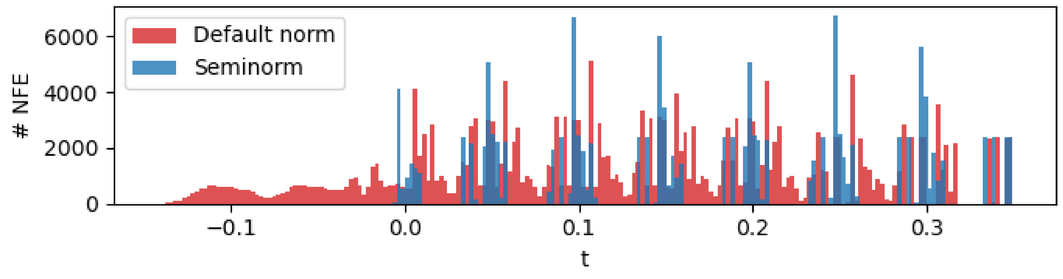
Method

a_θ does not appear on the right hand side of the adjoint equations. Thus it is just an integral—not an ODE. Using an ODE solver, it assumes that errors now propagate to create errors later, and takes steps small enough to compensate—in this case taking too many steps.

By switching the norm (used to measure error) for a seminorm that does not see a_θ, then many fewer steps are needed.

Results

Hamiltonian neural network: histogram of step locations.



Neural controlled differential equation: NFE for varying tolerances

	Default norm		Seminorm	
RTOL, ATOL	Accuracy (%)	Bwd. NFE (10^6)	Accuracy (%)	Bwd. NFE (10^6)
$10^{-3}, 10^{-6}$	92.6±0.4	14.36±1.09	92.5±0.5	**8.67±1.60**
$10^{-4}, 10^{-7}$	92.8±0.4	30.67±2.48	92.5±0.5	**12.75±2.00**
$10^{-5}, 10^{-8}$	92.4±0.7	77.95±4.47	92.9±0.4	**29.39±0.80**

Find out more:

Full paper: `https://arxiv.org/abs/2009.09457`
Code: `https://github.com/patrick-kidger/FasterNeuralDiffEq`
Email: `kidger@maths.ox.ac.uk`