Neural ODEs for Light Curves

Classifying and Approximating Light Curves

Anand Jain, University of Chicago
LSST Data Science Program at Fermilab

FERMILAB-POSTER-19-106-CD

Summary

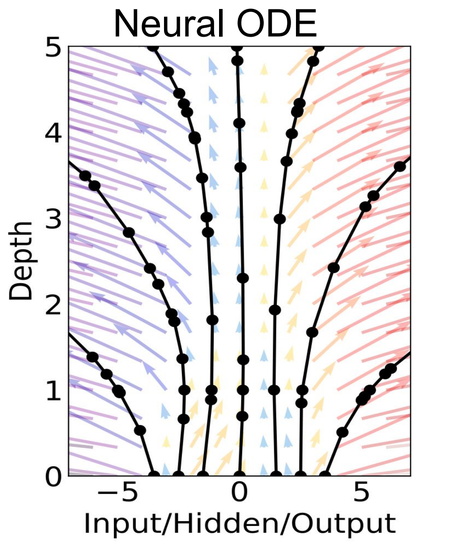
Neural ODEs remove discretizations that traditional layer based neural networks necessarily have. Neural ODEs are a continuous depth model that can trade precision and speed. I show that neural ODEs can estimate light curve time series using LSST Kaggle data.

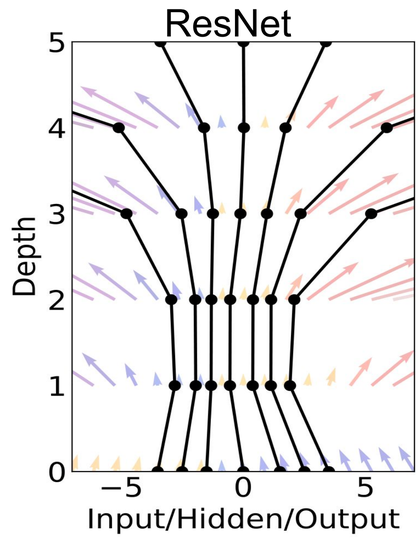
1. Background and Extensions of Neural ODEs

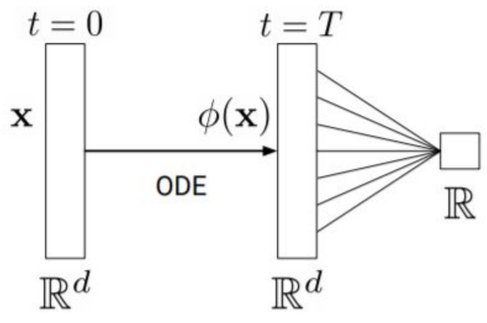
Resnets and RNNs transform a hidden state h_t to h_t+t using a neural network. This is analogous to Euler's method, in which a continuous transformation is discretized. If we take the limit as the step size between layers goes to zero, we arrive at the derivative of the hidden state wrt time.

dh(t)/dt = f(h(t), t, network parameters)

Neural ODEs can be solved at any point in time, making them prime candidates for time series learning.







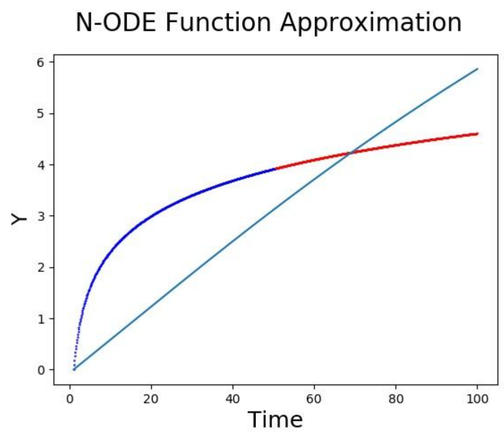
Neural ODE architecture. To get an output for regression or classification, a linear layer is used to map the ODE output to a value in R^N

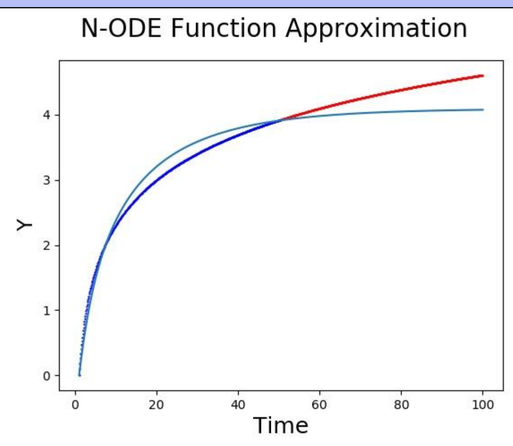
The time series latent ODEs in the original N-ODE paper uses an RNN in the reverse direction to encode the initial value. Rubanova et al. improved results using an ODE-RNN to compute this initial value. Improvements were shown on MuJoCo Physics simulations, Physionet, and Human activity classification. This improvement is directly applicable to LSST object classification.

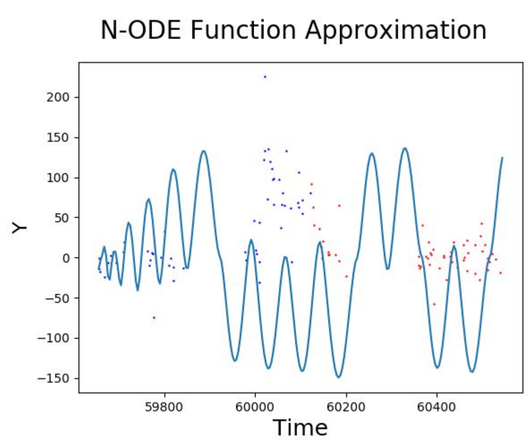
Dupont et al. shows that neural ODEs preserve the topology of the input space, this implies that there are functions that cannot be represented with n-ODEs. By increasing the ODE solvers input dimension, mappings that would be difficult preserving input topology, become much easier.

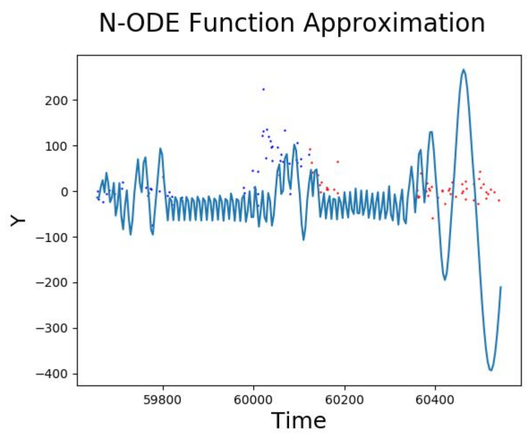
2. Preliminary Results

Large scale evaluation on the entire Kaggle set has not been performed, results mainly using toy data. Using a simple n-ODE, MNIST achieves 99% accuracy on the test set.
Left: 10 iterations. Right: 1 epoch (1000 iterations)
Figures: f(x) = log(x) is learned trained on blue data points, and extrapolated to red points.
A neural ODE learns a function fitting a single light curves' flux/time data using time as an input. I think training on all light curves with the same network would improve the ability for the network to extrapolate/predict.









3. Future Work

I would like to try to apply latent ODEs to an augmented input space and see if the two extensions of the original Neural Ordinary Differential Equations complement each other and achieve even better results.
Additionally, the Julia language has released a package for neural differential equation solving, DiffEqFlux.jl, which I would like to use on the LSST Kaggle data and see if there are differences in accuracy. Since Julia is a compiled language, I hope to demonstrate faster training time than equivalent neural differential equation models in Python.
Making use of more computation, data, augmenting the input space, and latent-odes, I hypothesize that winning Kaggle results can be achieved with less computation and hyperparameter tuning than current challenge winners.

Acknowledgements: Brian Nord (supervisor).

Chen et al: arXiv:1806.07366.
Rubanova et al: arXiv:1907.03907.
Dupont et al: arXiv:1904.01681

This document was prepared by Deep Skies Collaboration (deepskieslab.com) using the resources of the Fermi National Accelerator Laboratory (Fermilab), a U.S. Department of Energy, Office of Science, HEP User Facility. This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.