A Summary of Augmented Neural ODEs

Emily Nguyen

1 Introduction

This document provides a summary for a paper on augmented neural ordinary differential equations (NODEs). The assumption is that the audience is familiar with:

1. Ordinary differential equations (ODEs).
2. Neural networks.
3. Linear maps.

1.1 Overview

NODEs are the continuous equivalent of residual neural networks (ResNets) and provide another way to approximate functions (e.g., for classification, regression, etc). However, there are classes of functions that NODEs cannot represent because ODE trajectories are not allowed to intersect. Additionally, NODEs are afflicted with an increasing number of function evaluations (NFE), which occur during training as the flow gets increasingly complex. The authors claim to have found a solution for NODEs that results in fewer computations, more stability, and better generalization.

2 NODEs

The authors begin with a brief derivation of NODEs. For ResNets, the transformation of a hidden state $\mathbf{h}_t \in \mathbb{R}^d$ from one layer to the next is given by

$$\mathbf{h}_{t+1} = \mathbf{h}_t + \mathbf{f}_t(\mathbf{h}_t)$$

where $\mathbf{f}_t : \mathbb{R}^d \mapsto \mathbb{R}^d$ is a differentiable function (like a CNN). This has the form of Euler's method to approximate an ODE. This means that if $\Delta t = 1$, then $\mathbf{h}_t$ can be parametrized by the ODE

$$\frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{h}(t), t), \quad \mathbf{h}(0) = \mathbf{x}$$

where $\mathbf{x}$ is the input. The NODE version of a ResNet's forward pass is solving the ODE with input $\mathbf{x}$ and adjusting $\mathbf{f}(\mathbf{h}(t), t)$ so that output $\mathbf{y}$ is close to $\mathbf{y}_{\text{true}}$.

Let $\phi_t : \mathbb{R}^d \mapsto \mathbb{R}^d$ be the flow of the ODE (i.e., $\phi_t(\mathbf{x}) = \mathbf{h}(t)$), which measures the dependency of the ODE's states on the initial condition $\mathbf{x}$. Then we define the features of the ODE with $\phi(\mathbf{x})\phi_T(\mathbf{x})$ where T is the final time. To get an output in $\mathbb{R}$ instead of $\mathbb{R}^d$ (as you would want for classification or regression), define a NODE $g : \mathbb{R}^d \mapsto \mathbb{R}$ such that $g(\boldsymbol{x}) = \mathcal{L}(\phi(\boldsymbol{x}))$ and $\mathcal{L} : \mathbb{R}^d \mapsto \mathbb{R}$ where $\mathcal{L}$ is a linear map.

2.1 Limitations of NODEs

Functions that require the trajectories of the ODE to intersect render the NODEs ineffective in approximating them.

2.1.1 ResNet vs NODEs

Note that since ResNets are a discretization of ODEs, they can contain discrete jumps through which the trajectories can travel (and avoid intersecting). However, these jumps happen when there are large errors, so we can look at ResNets as ODE solutions with large errors.

2.1.2 NODEs Preserve Topology

The main issue is encapsulated within **Proposition 3**, which states that the "*feature mapping $\phi(\mathbf{x})$ is a homeomorphism, so the features of Neural ODEs preserve the topology of the input space*". This means that NODEs cannot tear connected regions apart to adequately approximate certain functions (i.e., functions where the features $\phi(\mathbf{x})$ for the points are not linearly separable). When NODEs attempt to approximate such functions, the flow gets more complex. This causes the ODE solver to require more steps to evaluate $\mathbf{f}$ at each step, which slows down the computation.

3 Augmented Neural ODEs

The solution provided is to augment the space we're in for solving the ODE from $\mathbb{R}^d$ to $\mathbb{R}^{d+p}$ so that the additional dimensions provide the trajectories more space to avoid intersections. The problem looks like

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{h}(t)\\ \mathbf{a}(t)\end{bmatrix} = \mathbf{f}\left(\begin{bmatrix}\mathbf{h}(t)\\ \mathbf{a}(t)\end{bmatrix}, t\right), \qquad \begin{bmatrix}\mathbf{h}(0)\\ \mathbf{a}(0)\end{bmatrix} = \begin{bmatrix}\mathbf{x}\\ \mathbf{0}\end{bmatrix}$$

where $\mathbf{a}(t) \in \mathbb{R}^p$ is a point in the augmented space, and every data point $\mathbf{x}$ is concatenated with a vector of zeros. This ODE is called an Augmented Neural ODE (ANODE).

3.1 Experiments

The paper details various experiments to compare the generalization accuracy and performance of various methods. The experiments include:

1. ResNet vs NODEs tested on simple regression tasks (trained on $g(\mathbf{x})$ and a linearly separable function) with ResNet outperforming NODEs (see Figure 5)

2. NODEs vs ANODEs tested on $g(\mathbf{x})$ with ANODEs learning simpler flows (see Figure 7), using fewer computations (see Figure 8), and generalizing better (see Figure 9)

3. NODEs vs ANODEs tested on MNIST, CIFAR10, SVHN, and 200 classes of 64×64 ImageNet with ANODEs training faster, achieving lower losses, being less computationally expensive, generalizing better and outperforming NODEs even when using the same number of parameters (see Figures 10-12)

The authors try to make "fair" comparisons between different architectures by running hyperparameter searches for each model and repeating each experiment 20 times, so the experiments appear to be well-designed.

4 Conclusion

Though running their own experiments to test ANODEs vs ResNets would strengthen their arguments, the experiments that the authors did do are rather convincing. ANODEs appear to perform significantly better than NODEs in every way. However, there are some limitations:

1. ANODEs are still slower than ResNets.
2. Changing the dimensions of the input space might not be desirable.
3. Adding too many channels/dimensions can lead to very unfortunate results (see Figure 12).