Neural Ordinary Differential Equations

Yifei Ding

January 24, 2025

Introduction

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t) \Longrightarrow \frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} = f(\mathbf{h}(t), t, \theta)$$

Benefits: memory efficiency, adaptive computation, scalable and invertible normalizing flows, continuous time-series models

Backpropagation

$$L(\mathbf{z}(t_1)) = L\left(\mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta)\mathrm{d}t\right) = L(\mathrm{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \theta))$$

Target:

$$\frac{\partial L}{\partial \theta}$$

Backpropagation

Let

$$\mathbf{a}(t) = \frac{\partial L}{\partial \mathbf{z}(t)},$$

then

$$\frac{\mathrm{d}\mathbf{a}(t)}{\mathrm{d}t} = -\mathbf{a}(t)^\top \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}},$$

thus

$$\frac{\partial L}{\partial \theta} = -\int_{t_1}^{t_0} \mathbf{a}(t)^\top \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \theta} \mathrm{d}t.$$

Backpropagation

Algorithm 2 Complete 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 ${\partial L}/{\partial \mathbf{z}(t_1)}$

$\frac{\partial L}{\partial t_1} = \frac{\partial L}{\partial \mathbf{z}(t_1)}^{\mathsf{T}} f(\mathbf{z}(t_1), t_1, \theta)$ ▷ Compute gradient w.r.t. t_1

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

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

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

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

return $\frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}, \frac{\partial L}{\partial t_0}, \frac{\partial L}{\partial t_1}$ ▷ Return all gradients

Replacing Resnet

Table 1: Performance on MNIST. †From LeCun et al. (1998).

	Test Error	# Params	Memory	Time
1-Layer MLP†	1.60%	0.24 M	-	-
ResNet	0.41%	0.60 M	$\mathcal{O}(L)$	$\mathcal{O}(L)$
RK-Net	0.47%	0.22 M	$\mathcal{O}(\tilde{L})$	$\mathcal{O}(\tilde{L})$
ODE-Net	0.42%	0.22 M	$\boldsymbol{\mathcal{O}(1)}$	$\mathcal{O}(\tilde{L})$

Continuous Normalizing Flows

$$\mathbf{z}_1 = f(\mathbf{z}_0) \Longrightarrow \log p(\mathbf{z}_1) = \log p(\mathbf{z}_0) - \log \left| \det \frac{\partial f}{\partial \mathbf{z}_0} \right|$$

$$\Downarrow$$

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\mathrm{tr}\left(\frac{\mathrm{d}f}{\mathrm{d}\mathbf{z}(t)} \right)$$

A Generative Lantent Function Time-series Model

Acknowledgement

Thank you!