ODE NETS

Teo Price-Broncucia
CU Boulder
April 2019

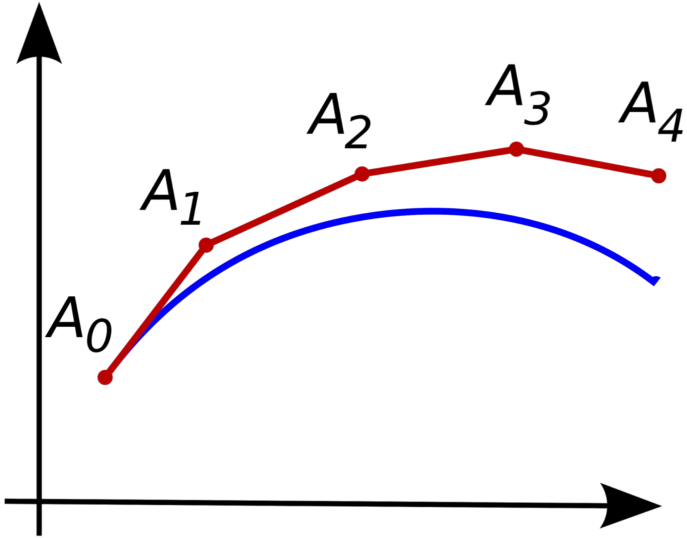
Neural Ordinary Differential Equations

- By Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, David Duvenaud.
  - From Vector Institute at University of Toronto.
- Won best paper at NIPS 2018 in December.
- Builds on work as old as 1988 when LeCun *et al.* proposed adjoint method for continuous time neural networks.

Can we think of RNN as an ODE?

Euler Method

$$y_{n+1} = y_n + hf(t_n, y_n)$$

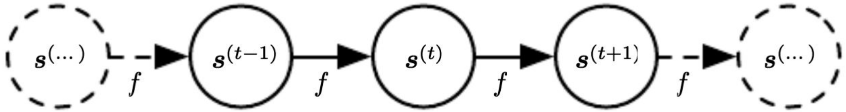


- A way to solve first order ODE when given initial value
- *h* is step size from t_n to t_{n+1}
- Recurrent because definition of y at time n+1 refers back to same definition at time n.

RNN

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

In textbook:



- Recurrent because definition of h at time t+1 refers back to same definition at time t.
- “Dynamical System”
- Looks very similar to Euler method

RNN to ODE

- If we add more and more layers and take smaller and smaller steps:

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

  - With input layer **h**(0) the output layer **h**(T) is defined as the solution to this ODE problem.
  - Can use ODE solver where the selected accuracy is the equivalent of depth.

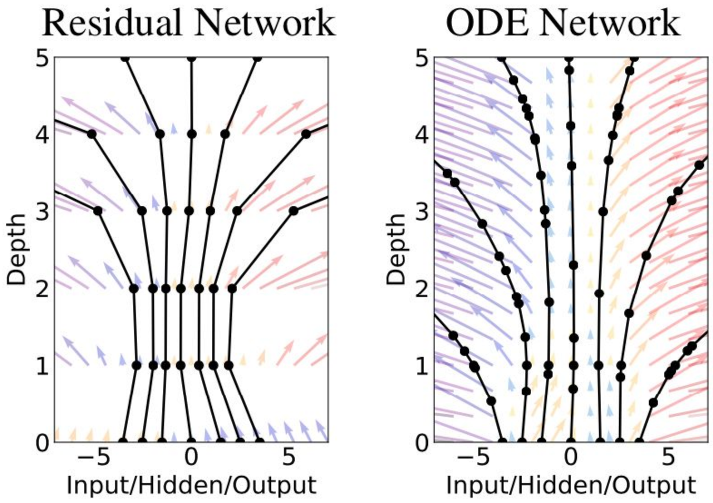


Figure 1: *Left:* A Residual network defines a discrete sequence of finite transformations. *Right:* A ODE network defines a vector field, which continuously transforms the state. *Both:* Circles represent evaluation locations.

Why use ODE solvers?

Advantages

- Memory Efficiency:
  - Don’t have to store all intermediate quantities of forward pass.
  - Constant memory as function of depth!
- Adaptive Computation
  - Don’t have to use Euler’s method, can use a variety of modern ODE solvers.
  - Can choose lower accuracy after training for low power/real-time applications.
- Parameter Efficiency
- Normalizing Flows
- Continuous time series models
  - Don’t need to artificially discretize observations.

How Do We Do Backpropagation?

- We want gradients with respect to theta and the starting value $z(t_0)$.
- We will define adjoint states of the form:

$$\mathbf{a}(t) = \frac{dL}{d\mathbf{z}(t)} \xrightarrow{\text{Can show}} \frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)\frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}(t)}$$

- We can define analogous adjoints for theta and t

$$\mathbf{a}_\theta(t) := \frac{dL}{d\theta(t)}, \quad \mathbf{a}_t(t) := \frac{dL}{dt(t)}$$

Backpropagation cont.

- Combine into an augmented system

$$\frac{d}{dt}\begin{bmatrix}\mathbf{z}\\ \theta\\ t\end{bmatrix}(t) = f_{aug}([\mathbf{z},\theta,t]) := \begin{bmatrix} f([\mathbf{z},\theta,t])\\ \mathbf{0}\\ 1\end{bmatrix}$$

$$\boxed{\frac{\partial\theta(t)}{\partial t} = \mathbf{0} \qquad \frac{dt(t)}{dt} = 1}$$

We know this

$$\mathbf{a}_{aug} := \begin{bmatrix}\mathbf{a}\\ \mathbf{a}_\theta\\ \mathbf{a}_t\end{bmatrix},\ \mathbf{a}_\theta(t) := \frac{dL}{d\theta(t)},\ \mathbf{a}_t(t) := \frac{dL}{dt(t)}$$

Backpropagation cont.

$$\frac{\partial f_{aug}}{\partial[\mathbf{z},\theta,t]} = \begin{bmatrix} \frac{\partial f}{\partial \mathbf{z}} & \frac{\partial f}{\partial \theta} & \frac{\partial f}{\partial t} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}(t)$$

$$\boxed{\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)\frac{\partial f(\mathbf{z}(t),t,\theta)}{\partial \mathbf{z}(t)}}$$

$$\frac{d\mathbf{a}_{aug}(t)}{dt} = -\begin{bmatrix}\mathbf{a}(t) & \mathbf{a}_\theta(t) & \mathbf{a}_t(t)\end{bmatrix}\frac{\partial f_{aug}}{\partial[\mathbf{z},\theta,t]}(t) = -\begin{bmatrix}\mathbf{a}\frac{\partial f}{\partial \mathbf{z}} & \mathbf{a}\frac{\partial f}{\partial \theta} & \mathbf{a}\frac{\partial f}{\partial t}\end{bmatrix}(t)$$

- This augmented adjoint state is solved again with an ODE solver.
- Then the relevant gradients are found by integrating backward.

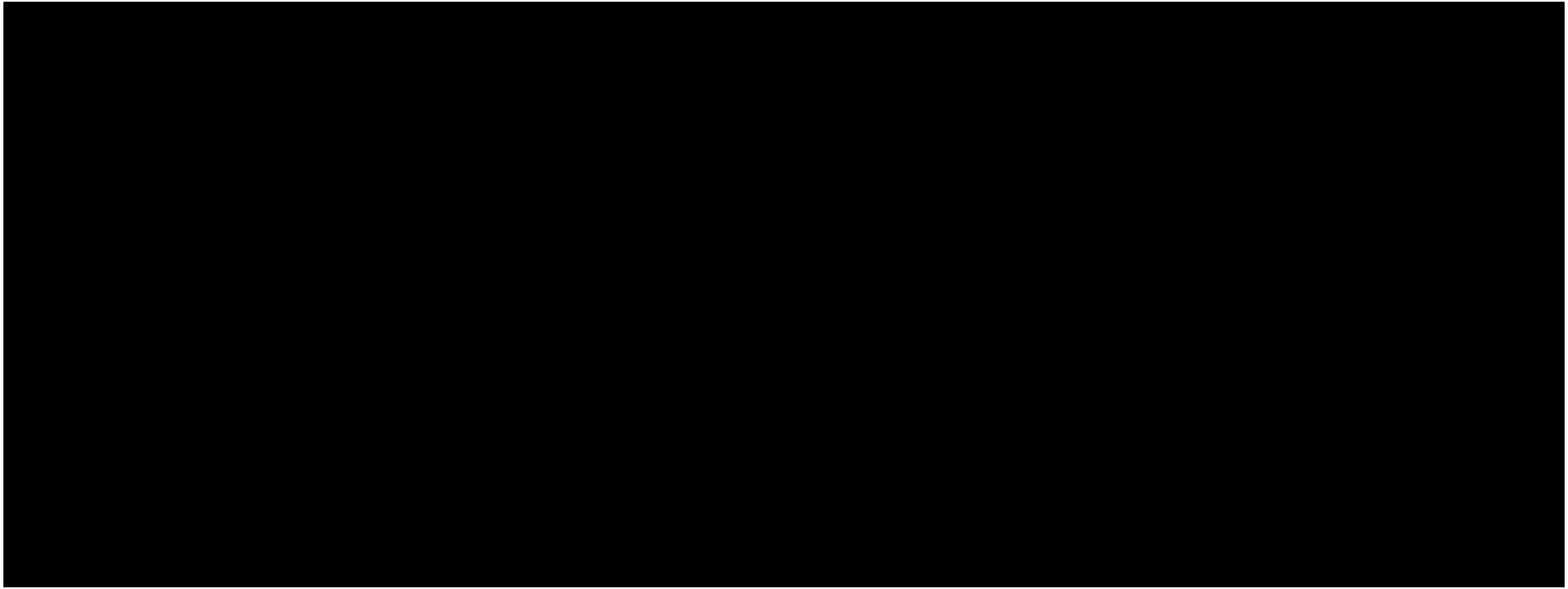
Backpropagation cont.

$$\frac{dL}{d\theta} = \mathbf{a}_\theta(t_0) = -\int_{t_N}^{t_0} \mathbf{a}(t) \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \theta} \, dt$$

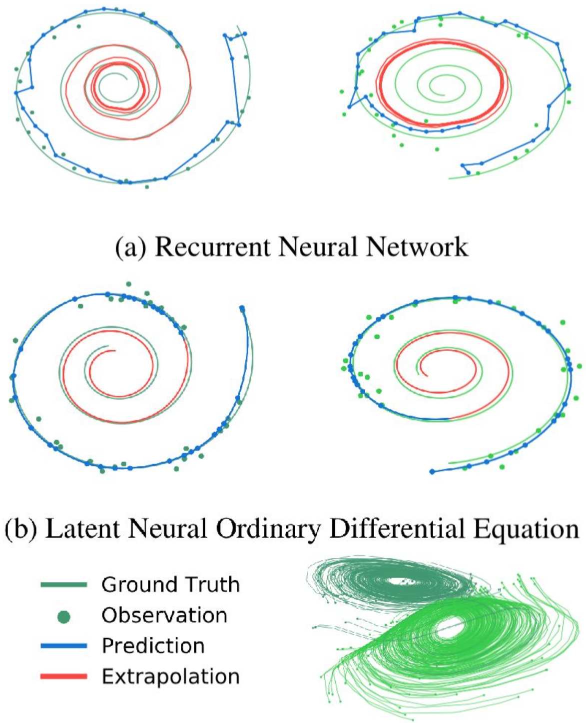
$$\frac{dL}{dt_N} = \mathbf{a}(t_N) f(\mathbf{z}(t_N), t_N, \theta)$$

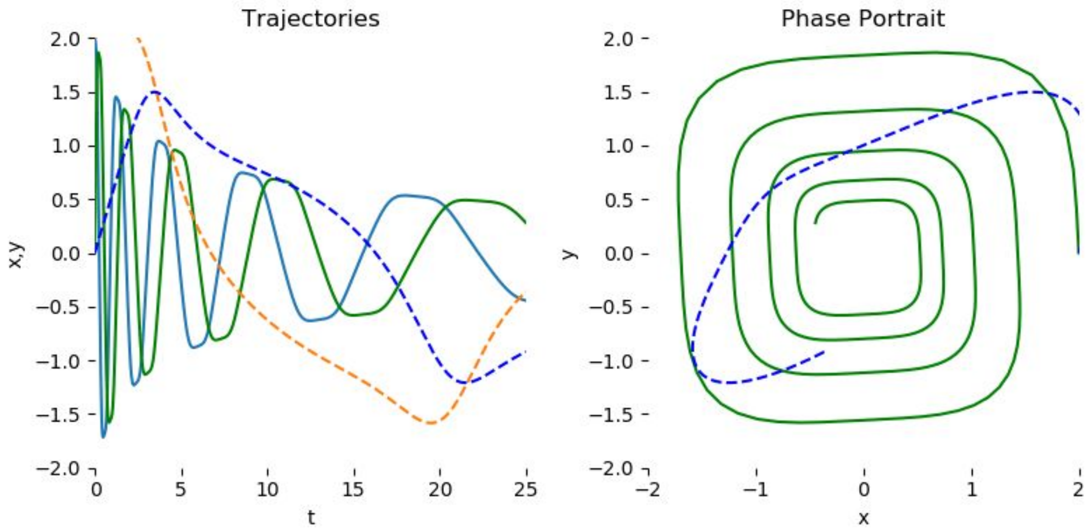
$$\frac{dL}{dt_0} = \mathbf{a}_t(t_0) = \mathbf{a}_t(t_N) - \int_{t_N}^{t_0} \mathbf{a}(t) \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial t} \, dt$$

Examples!
Pictures!



Continuous Normalizing Flows





Time-series Latent ODE

Drawbacks + Challenges

- Minibatching not so straightforward.
- Unique solution?
  - With Lipshitz nonlinearities, yes.
- Must choose error tolerance on forward/backward passes
- Numerical error from backwards pass?
  - Not seen in application

Thanks!