Differential equations and neural networks

Julius Ruseckas

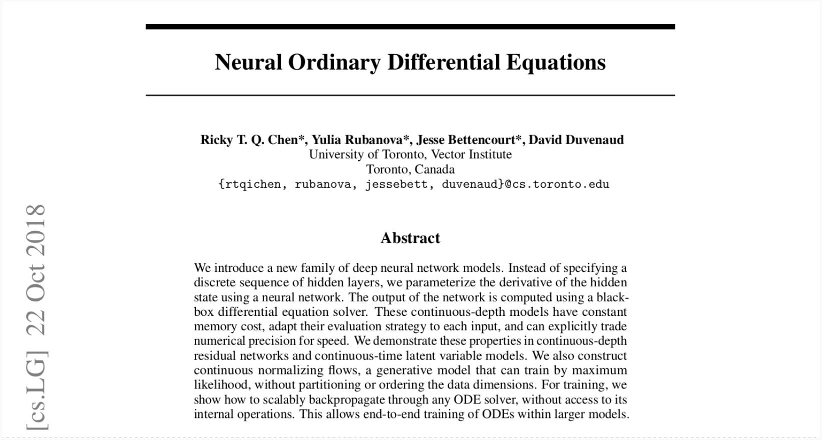
October 28, 2019

Baltic Institute of Advanced Technology

Outline

Motivation

Motivation



Neural Ordinary Differential Equations

Ricky T. Q. Chen*, Yulia Rubanova*, Jesse Bettencourt*, David Duvenaud
University of Toronto, Vector Institute
Toronto, Canada
{rtqichen, rubanova, jessebett, duvenaud}@cs.toronto.edu

Abstract

We introduce a new family of deep neural network models. Instead of specifying a discrete sequence of hidden layers, we parameterize the derivative of the hidden state using a neural network. The output of the network is computed using a black-box differential equation solver. These continuous-depth models have constant memory cost, adapt their evaluation strategy to each input, and can explicitly trade numerical precision for speed. We demonstrate these properties in continuous-depth residual networks and continuous-time latent variable models. We also construct continuous normalizing flows, a generative model that can train by maximum likelihood, without partitioning or ordering the data dimensions. For training, we show how to scalably backpropagate through any ODE solver, without access to its internal operations. This allows end-to-end training of ODEs within larger models.

arXiv:1806.07366 [cs.LG] (2018).

32nd Conference on Neural Information Processing Systems (NeurIPS 2018)

Idea is old

Continuous time recurrent neural networks

$$\tau_i \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{n} w_{j,i}\sigma(x_j - \theta_j) + I_i(t)$$

J. J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons. Proc. Natl. Acad. Sci. USA, **81**, 3088–3092 (1984).

Current research

- W. E, *A proposal on machine learning via dynamical systems*, Commun. Math. Stat. **5**, 1–11 (2017).
- E. Haber, L. Ruthotto, E. Holtham, and S.-H. Jun, *Learning across scales—multiscale methods for convolution neural networks*, arXiv:1703:02009 (2017).
- E. Haber and L. Ruthotto, *Stable architectures for deep neural networks*, Inverse Problems **34**, 014004 (2018).
- B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham, *Reversible architectures for arbitrarily deep residual neural networks*, arXiv:1709.03698 (2017).
- J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud, and J.-H. Jacobsen, *Invertible residual networks*, arXiv:1811.00995 (2018).
- Y. Lu, A. Zhong, Q. Li, and B. Dong, *Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations*, arXiv:1710:10121 (2017).
- T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, *Neural ordinary differential equations*. In Advances in Neural Information Processing Systems 31, p. 6571–6583. Curran Associates, Inc., 2018.
- E. Dupont, A. Doucet, and Y. W. Teh, *Augmented neural ODEs*, arXiv:1904:01681 (2019).
- M. Y. Niu, I. L. Chuang, and L. Horesh, *Recurrent neural networks in the eye of differential equations*, arXiv:1904.12933 (2019).

Ordinary differential equations from the point of view of machine learning

Differential equations

Nonlinear ordinary differential equations used as a machine learning model

$$\frac{d}{dt}\boldsymbol{x}(t) = \boldsymbol{F}(\boldsymbol{x}(t), \boldsymbol{q}(t))$$

- $\boldsymbol{F}$ nonlinear functions
- $\boldsymbol{q}(t)$ arbitrary parameters

Output $\boldsymbol{x}(T)$ is classified using a linear classifier.

Differential equations

Loss function $\mathcal{L}(\boldsymbol{x}(T))$ is a function of a final output $\boldsymbol{x}(T)$.

Question

How to minimize the loss?

Differential equations

Answer

Use the method of Lagrange multipliers

Functional

$$S = \mathcal{L}(\boldsymbol{x}(T)) + \int_0^T \boldsymbol{a}(t)[\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{q}) - \dot{\boldsymbol{x}}]\, dt$$

where $\boldsymbol{a}(t)$ is a vector of Lagrange multipliers

Loss minimization

We obtain:

$$-\frac{d}{dt}\boldsymbol{a}(t) = \boldsymbol{a}(t)\frac{\partial}{\partial \boldsymbol{x}(t)}\boldsymbol{F}(\boldsymbol{x}(t), \boldsymbol{q}(t))$$

with the condition

$$\boldsymbol{a}(T) = \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}(T)}$$

Differential equation for backward propagation, known as adjoint equation

Then

$$\frac{\delta \mathcal{L}}{\delta \boldsymbol{q}(t)} = \boldsymbol{a}(t)\frac{\partial}{\partial \boldsymbol{q}(t)}\boldsymbol{F}(\boldsymbol{x}(t), \boldsymbol{q}(t))$$

Euler-Lagrange form of the equations

Forward and backward equations can be written in the form of the Euler-Lagrange equations

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{a}} - \frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{a}}} =& 0\,,\\ \frac{\partial L}{\partial \boldsymbol{x}} - \frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{x}}} =& 0\end{aligned}$$

with the Lagrangian

$$L(\boldsymbol{x}, \boldsymbol{a}, \dot{\boldsymbol{x}}, t) = \boldsymbol{a}\dot{\boldsymbol{x}} - \boldsymbol{a}\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{q}(t))$$

Hamiltonian form of the equations

Generalized momentum

$$\frac{\partial L}{\partial \dot{\boldsymbol{x}}} = \boldsymbol{a}$$

The Hamiltonian

$$H(\boldsymbol{x}, \boldsymbol{a}, t) = \boldsymbol{a}\dot{\boldsymbol{x}} - L = \boldsymbol{a}\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{q}(t))$$

Forward and backward equations can be written take the form of the Hamilton equations

$$\begin{aligned} \frac{d\boldsymbol{x}}{dt} &= \frac{\partial H}{\partial \boldsymbol{a}}\,, \\ \frac{d\boldsymbol{a}}{dt} &= -\frac{\partial H}{\partial \boldsymbol{x}}\,. \end{aligned}$$

Consequences

- If the parameters $\boldsymbol{q}$ do not depend on t then

$$\boldsymbol{a}\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{q}) = \mathrm{const}_t$$

- Liouville's theorem

$$\frac{d}{dt} P(\boldsymbol{x}, \boldsymbol{a}, t) = 0$$

where $P(\boldsymbol{x}, \boldsymbol{a}, t)$ is the probability density function

How do Neural ODEs learn?

- Mutual information does not change:

$$\frac{d}{dt} I(Y; X(t)) = 0$$

  Neural ODEs cannot learn?

- Coarse graining
- Information entropy

$$\mathcal{H}(t) = -\int P(\boldsymbol{x}, t) \log P(\boldsymbol{x}, t)\, d^N\boldsymbol{x}$$

  has time derivative

$$\frac{d}{dt}\mathcal{H}(t) = \int \mathrm{Tr}\left\{ \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{q}(t)) \right\} P(\boldsymbol{x}, t)\, d^N\boldsymbol{x}$$

From differential equations to neural networks

From differential equations to neural networks

Sum of linear and non-linear parts

$$\frac{d}{dt}\boldsymbol{x}(t) = \boldsymbol{w}(t)\boldsymbol{x} + \boldsymbol{b}(t) + g(\boldsymbol{x})$$

For example, nonlinearity $g(x) = -\gamma x^3$ leads to

$$\frac{d}{dt}x_i(t) = \sum_j w_{i,j}(t)x_j(t) + b_i(t) - \gamma x_i(t)^3$$

and backward propagation

$$-\frac{d}{dt}a_i(t) = \sum_j a_j(t)w_{j,i}(t) - 3\gamma x_i(t)^2 a_i(t)$$

From differential equations to neural networks

Euler's method:

$$\boldsymbol{x}_{l+1} = \boldsymbol{x}_l + \Delta t \boldsymbol{w}_l \boldsymbol{x}_l + \Delta t \boldsymbol{b}_l + \Delta t g(\boldsymbol{x}_l)$$

Let us make a second-order error:

$$\boldsymbol{x}_{l+1} = \boldsymbol{x}_l + \Delta t \boldsymbol{w}_l \boldsymbol{x}_l + \Delta t \boldsymbol{b}_l + \Delta t g(\boldsymbol{x}_l + \Delta t \boldsymbol{w}_l \boldsymbol{x}_l + \Delta t \boldsymbol{b}_l)$$

From differential equations to neural networks

New non-linear function

$$h(x) = x + \Delta t g(x)$$

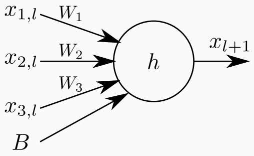
and new matrices

$$\boldsymbol{W}_l = \boldsymbol{I} + \Delta t \boldsymbol{w}_l \,, \\ \boldsymbol{B}_l = \Delta t \boldsymbol{b}_l$$

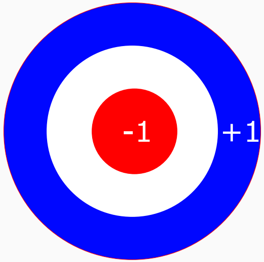
Then

$$\boldsymbol{x}_{l+1} = h(\boldsymbol{W}_l \boldsymbol{x}_l + \boldsymbol{B}_l)$$

We get a feedforward neural network



Let us consider two-dimensional function



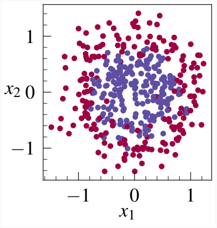
The features of Neural ODEs preserve the topology of the input space, thus Neural ODEs cannot represent this function

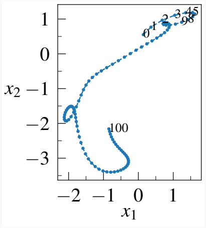
Discrete $>$ Continuous

- Error arising when taking discrete steps allows the trajectories to cross
- Training on finite sample of inputs the flow could squeeze through the gaps between sampled points

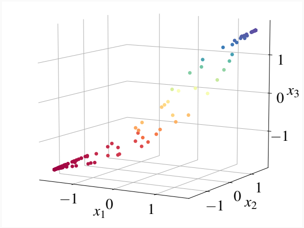
Example

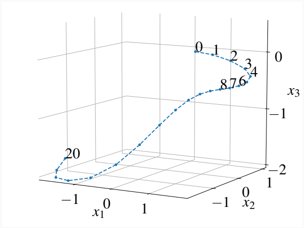
2D space





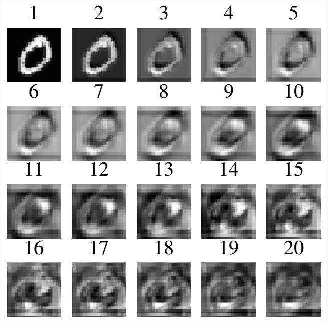
3D space



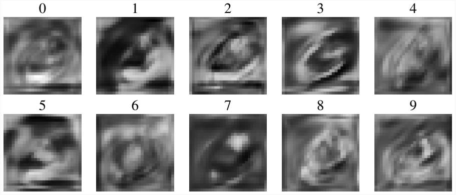


Example II

Activations in the first channel after each layer.



Activations in the final layer



Summary

Summary

- The gradient of the loss function with respect to to the hidden state can be considered as a generalized momentum conjugate to the hidden state, allowing application of the tools of classical mechanics.
- Not only residual networks, but also feedforward neural networks with small nonlinearities and the weights matrices deviating only slightly from identity matrices can be related to the differential equations

Thank you for your attention!

arXiv:1909.03767 [stat.ML] (2019)