Residual Neural Networks

Numerical Methods for Deep Learning

Deep Neural Networks in Practice

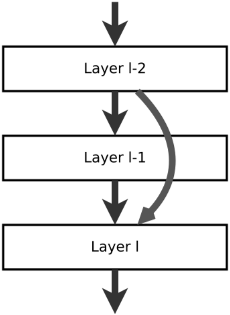
(Some) challenges with deep networks

- Computational costs (architecture have millions or billions of parameters)
- difficult to design
- difficult to train (exploding/vanishing gradients)
- unpredictable performance

Deep Neural Networks in Practice

(Some) challenges with deep networks

- Computational costs (architecture have millions or billions of parameters)
- difficult to design
- difficult to train (exploding/vanishing gradients)
- unpredictable performance



In 2015, He et al. [6, 7] proposed a new architecture that solves many of the problems

Simplified Residual Neural Network

Residual Network

$$
\begin{aligned}
\mathbf{Y}_1 &= \mathbf{Y}_0 + \sigma(\mathbf{K}_0\mathbf{Y}_0 + \mathbf{b}_0) \\
\vdots &= \vdots \\
\mathbf{Y}_N &= \mathbf{Y}_{N-1} + \sigma(\mathbf{K}_{N-1}\mathbf{Y}_{N-1} + \mathbf{b}_{N-1})
\end{aligned}
$$

And use $\mathbf{Y}_N$ to classify. This leads to the optimization problem

$$
\min_{\mathbf{K}_{0,\ldots,N-1},\mathbf{b}_{0,\ldots,N-1},\mathbf{W}} \; E\left(\mathbf{W}\mathbf{Y}_N(\mathbf{K}_1,\ldots,\mathbf{K}_{N-1},\mathbf{b}_1,\ldots,\mathbf{b}_{N-1}),\mathbf{C}^{\mathrm{obs}}\right)
$$

Leads to smoother objective function [8].

Learning Objective: Residual Neural Networks

In this module we consider residual neural networks

Learning tasks:

- regression
- segmentation
- classification

Numerical methods:

- differential equations

Stability of Deep Residual Networks

Why are ResNets more stable?
A small change

$$
\begin{aligned}
\mathbf{Y}_1 &= \mathbf{Y}_0 + h\sigma(\mathbf{K}_0\mathbf{Y}_0 + \mathbf{b}_0) \\
\vdots &= \vdots \\
\mathbf{Y}_N &= \mathbf{Y}_{N-1} + h\sigma(\mathbf{K}_{N-1}\mathbf{Y}_{N-1} + \mathbf{b}_{N-1})
\end{aligned}
$$

Stability of Deep Residual Networks

Why are ResNets more stable?
A small change

$$
\begin{aligned}
\mathbf{Y}_1 &= \mathbf{Y}_0 + h\sigma(\mathbf{K}_0\mathbf{Y}_0 + \mathbf{b}_0) \\
\vdots &= \vdots \\
\mathbf{Y}_N &= \mathbf{Y}_{N-1} + h\sigma(\mathbf{K}_{N-1}\mathbf{Y}_{N-1} + \mathbf{b}_{N-1})
\end{aligned}
$$

This is nothing but a forward Euler discretization of the Ordinary Differential Equation (ODE)

$$
\partial_t \mathbf{Y}(t) = \sigma(\mathbf{K}(t)\mathbf{Y}(t) + \mathbf{b}(t)), \qquad \mathbf{Y}(0) = \mathbf{Y}_0
$$

Get intuition about ResNet behavior by using tools from nonlinear ODEs [5, 4]. A word of warning is [**?**].

ODE Crash Course

Consider the ODE

$$\partial_t \mathbf{y}(t) = f(\mathbf{y}(t)), \quad \mathbf{y}(0) = \mathbf{y}_0$$

with f differentiable and Jacobian

$$\mathbf{J}(\mathbf{y}) = \left(\frac{\partial f}{\partial \mathbf{y}}\right)^\top$$

Then (see also [2, 3, 1])

- If $Re(\mathrm{eig}(\mathbf{J})) > 0$ $\quad \to$ Unstable
- If $Re(\mathrm{eig}(\mathbf{J})) < 0$ $\quad \to$ Stable ($\to$ to stationary point)
- If $Re(\mathrm{eig}(\mathbf{J})) = 0$ $\quad \to$ Stable, energy bounded

Reality: f time-dependent ($\rightsquigarrow$ penalize time derivatives of weights or use heavier tools, e.g., monotone operators, kinematic eigenvalues)

Stability of Residual Network

Assume forward propagation of single example $\mathbf{y}_0$

$$\partial_t \mathbf{y}(t) = \sigma(\mathbf{K}(t)\mathbf{y}(t) + \mathbf{b}(t)), \qquad \mathbf{y}(0) = \mathbf{y}_0$$

Stability of Residual Network

Assume forward propagation of single example $\mathbf{y}_0$

$$\partial_t \mathbf{y}(t) = \sigma(\mathbf{K}(t)\mathbf{y}(t) + \mathbf{b}(t)), \qquad \mathbf{y}(0) = \mathbf{y}_0$$

The Jacobian is

$$\mathbf{J}(t) = \mathrm{diag}\left(\sigma'(\mathbf{K}(t)\mathbf{y}(t) + \mathbf{b}(t))\right)\mathbf{K}(t)$$

Here, $\sigma'(x) \geq 0$ for tanh, ReLU, ...

Stability of Residual Network

Assume forward propagation of single example $\mathbf{y}_0$

$$\partial_t \mathbf{y}(t) = \sigma(\mathbf{K}(t)\mathbf{y}(t) + \mathbf{b}(t)), \qquad \mathbf{y}(0) = \mathbf{y}_0$$

The Jacobian is

$$\mathbf{J}(t) = \mathrm{diag}\left(\sigma'(\mathbf{K}(t)\mathbf{y}(t) + \mathbf{b}(t))\right)\mathbf{K}(t)$$

Here, $\sigma'(x) \geq 0$ for tanh, ReLU, ...

Hence, we need to enforce stability. One option:

1. $\mathbf{J}$ constant in time (or changes slowly)
2. $Re(\mathrm{eig}(\mathbf{K}(t))) = 0$ for every t

Stability of Residual Network

Assume forward propagation of single example $\mathbf{y}_0$

$$\partial_t \mathbf{y}(t) = \sigma(\mathbf{K}(t)\mathbf{y}(t) + \mathbf{b}(t)), \qquad \mathbf{y}(0) = \mathbf{y}_0$$

The Jacobian is

$$\mathbf{J}(t) = \mathrm{diag}\left(\sigma'(\mathbf{K}(t)\mathbf{y}(t) + \mathbf{b}(t))\right)\mathbf{K}(t)$$

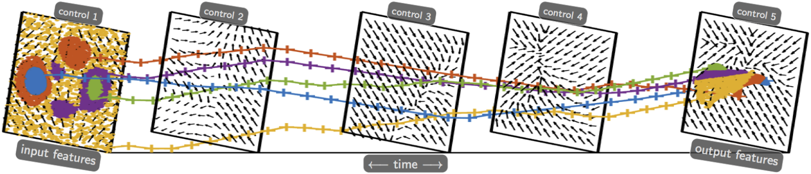
Here, $\sigma'(x) \geq 0$ for tanh, ReLU, ...

Hence, we need to enforce stability. One option:

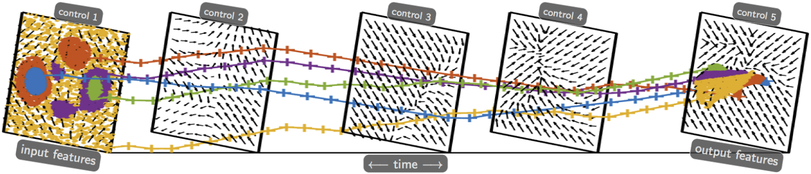
1. $\mathbf{J}$ constant in time (or changes slowly)
2. $Re(\mathrm{eig}(\mathbf{K}(t))) = 0$ for every t

Remember that we learn $\mathbf{K} \rightsquigarrow$ ensure stability by regularization/constraints!

Residual Network as a Path Planning Problem



Residual Network as a Path Planning Problem

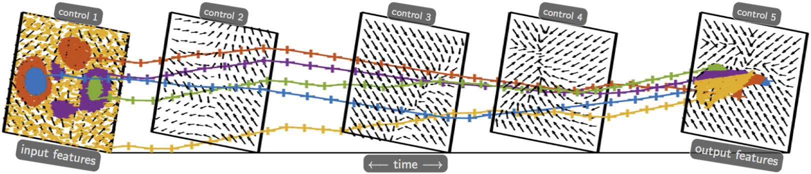


Forward propagation in residual network (continuous)

$$\partial_t \mathbf{Y}(t) = \sigma(\mathbf{K}(t)\mathbf{Y}(t) + \mathbf{b}(t)), \qquad \mathbf{Y}(0) = \mathbf{Y}_0$$

The goal is to plan a path (via $\mathbf{K}$ and b) such that the initial data can be linearly separated

Residual Network as a Path Planning Problem



Forward propagation in residual network (continuous)

$$\partial_t \mathbf{Y}(t) = \sigma(\mathbf{K}(t)\mathbf{Y}(t) + \mathbf{b}(t)), \qquad \mathbf{Y}(0) = \mathbf{Y}_0$$

The goal is to plan a path (via **K** and b) such that the initial data can be linearly separated

Question: What is a layer, what is depth?

Stability: Continuous vs. Discrete

Assume $\mathbf{K}$ is chosen so that the (continuous) forward propagation is stable

$$\partial_t \mathbf{Y}(t) = \sigma(\mathbf{K}(t)\mathbf{Y}(t) + \mathbf{b}(t)), \qquad \mathbf{Y}(0) = \mathbf{Y}_0$$

And assume we use the forward Euler method to discretize

$$\mathbf{Y}_{l+1} = \mathbf{Y}_l + h\sigma(\mathbf{K}_l \mathbf{Y}_l + \mathbf{b}_l)$$

Is the network stable?

Stability: Continuous vs. Discrete

Assume $\mathbf{K}$ is chosen so that the (continuous) forward propagation is stable

$$\partial_t \mathbf{Y}(t) = \sigma(\mathbf{K}(t)\mathbf{Y}(t) + \mathbf{b}(t)), \qquad \mathbf{Y}(0) = \mathbf{Y}_0$$

And assume we use the forward Euler method to discretize

$$\mathbf{Y}_{l+1} = \mathbf{Y}_l + h\sigma(\mathbf{K}_l \mathbf{Y}_l + \mathbf{b}_l)$$

Is the network stable?

Not always ...

Stability: A Simple Example

Look at the simplest possible forward propagation

$$\partial_t \mathbf{Y}(t) = \lambda \mathbf{Y}(t), \qquad \lambda \in \mathbb{C}$$

And assume we use the forward Euler to discretize

$$\mathbf{Y}_{l+1} = \mathbf{Y}_l + h\lambda \mathbf{Y}_l = (1 + h\lambda)\mathbf{Y}_l$$

Then the method is stable only if

$$|1 + h\lambda| \leq 1$$

Not every network is stable! Time step size depends on λ (which depends on $\mathbf{K}$ that is trained).

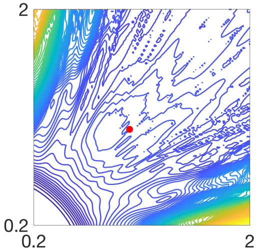
Why you should care about stability - 1

$$\min_{\theta} \frac{1}{2}\|\mathbf{Y}_N(\theta) - \mathbf{C}\|_F^2 \qquad \mathbf{Y}_{j+1}(\theta) = \mathbf{Y}_j(\theta) + \frac{10}{N}\tanh\left(\mathbf{K}\mathbf{Y}_j(\theta)\right)$$

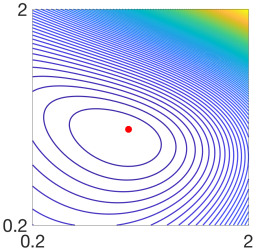
where $\mathbf{C} = \mathbf{Y}_{200}(1,1)$, $\mathbf{Y}_0 \sim \mathcal{N}(0,1)$, and

$$\mathbf{K}(\theta) = \begin{pmatrix} -\theta_1 - \theta_2 & \theta_1 & \theta_2 \\ \theta_2 & -\theta_1 - \theta_2 & \theta_1 \\ \theta_1 & \theta_2 & -\theta_1 - \theta_2 \end{pmatrix}$$

objective, $N = 5$



objective, $N = 100$



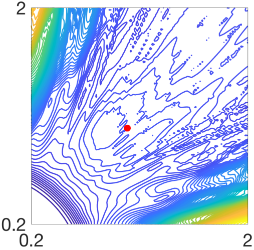
Why you should care about stability - 1

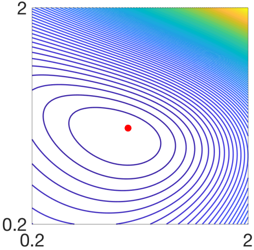
$$\min_{\theta} \frac{1}{2}\|\mathbf{Y}_N(\theta) - \mathbf{C}\|_F^2 \qquad \mathbf{Y}_{j+1}(\theta) = \mathbf{Y}_j(\theta) + \frac{10}{N}\tanh\left(\mathbf{K}\mathbf{Y}_j(\theta)\right)$$

where $\mathbf{C} = \mathbf{Y}_{200}(1,1)$, $\mathbf{Y}_0 \sim \mathcal{N}(0,1)$, and

$$\mathbf{K}(\theta) = \begin{pmatrix} -\theta_1 - \theta_2 & \theta_1 & \theta_2 \\ \theta_2 & -\theta_1 - \theta_2 & \theta_1 \\ \theta_1 & \theta_2 & -\theta_1 - \theta_2 \end{pmatrix}$$

objective, $N = 5$ objective, $N = 100$

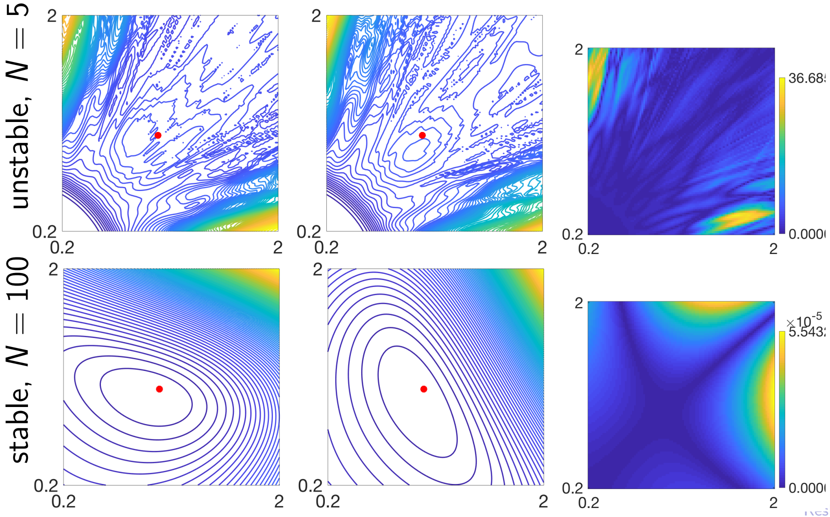




Next: Compare different inputs $\sim$ generalization

Why you should care about stability - 2

objective, $\mathbf{Y}_0^{\mathrm{train}}$ | objective, $\mathbf{Y}_0^{\mathrm{test}}$ | abs. diff



Stability: A Non-Trivial Example

Consider the antisymmetric kernel model

$$\mathbf{K}(t) = \mathbf{K}(t) - \mathbf{K}(t)^\top.$$

Here, $Re(\mathrm{eig}(\mathbf{J}(t))) = 0$ for all $\boldsymbol{\theta}$.

Stability: A Non-Trivial Example

Consider the antisymmetric kernel model

$$\mathbf{K}(t) = \mathbf{K}(t) - \mathbf{K}(t)^\top.$$

Here, $Re(\mathrm{eig}(\mathbf{J}(t))) = 0$ for all $\boldsymbol{\theta}$.

Assume we use the forward Euler to discretize

$$\mathbf{Y}_{l+1} = \mathbf{Y}_l + h\sigma((\mathbf{K}_l - \mathbf{K}_l^\top)\mathbf{Y}_l + \mathbf{b}_l).$$

Tricky question: How to pick h to ensure stability?

Stability: A Non-Trivial Example

Consider the antisymmetric kernel model

$$\mathbf{K}(t) = \mathbf{K}(t) - \mathbf{K}(t)^\top.$$

Here, $Re(\mathrm{eig}(\mathbf{J}(t))) = 0$ for all $\boldsymbol{\theta}$.

Assume we use the forward Euler to discretize

$$\mathbf{Y}_{l+1} = \mathbf{Y}_l + h\sigma((\mathbf{K}_l - \mathbf{K}_l^\top)\mathbf{Y}_l + \mathbf{b}_l).$$

Tricky question: How to pick h to ensure stability?
Answer: Impossible since eigenvalues of Jacobian are imaginary. Need other method than forward Euler.

Multistep Methods

For the antisymmetric kernel model

$$\mathbf{K}(t) = \mathbf{K}(t) - \mathbf{K}(t)^\top$$

forward Euler is unconditionally unstable. One way out is using higher-order Runge Kutta methods and small step size.

$$\mathbf{Z}_{l+1} = \mathbf{Y}_l + hf(\boldsymbol{\theta}_l, \mathbf{Y}_l)$$
$$\mathbf{Y}_{l+1} = \mathbf{Y}_l + \frac{h}{2}\left(f(\boldsymbol{\theta}_l, \mathbf{Y}_l) + f(\boldsymbol{\theta}_{l+1}, \mathbf{Z}_{l+1})\right)$$

Better options: RK3, RK4, semi-implicit time stepping.

Σ: Residual Neural Networks

Idea: Add a skip connection to multilayer perceptrons

Discrete:

$$\mathbf{Y}_{l+1} = \mathbf{Y}_l + \sigma(\mathbf{K}_l \mathbf{Y}_l + \mathbf{b}_l)$$

Continuous:

$$\partial_t \mathbf{Y}(t) = \sigma(\mathbf{K}(t)\mathbf{Y}(t) + \mathbf{b}(t))$$

Discussion:

- train well with hundreds of layers
- won many awards and competitions
- can be analyzed as a differential equation
- ResNets are generally not stable
- Change differential equation and(!) discretization to achieve stability
- next time: ResNet training and optimal control

References

[1] U. Ascher. *Numerical methods for Evolutionary Differential Equations*. SIAM, Philadelphia, 2010.

[2] U. Ascher and L. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, Philadelphia, PA, 1998.

[3] U. M. Ascher and C. Greif. A First Course on Numerical Methods. SIAM, Philadelphia, 2011.

[4] W. E. A Proposal on Machine Learning via Dynamical Systems. *Communications in Mathematics and Statistics*, 5(1):1–11, Mar. 2017.

[5] E. Haber and L. Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34:014004, 2017.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[8] H. Li, Z. Xu, G. Taylor, and T. Goldstein. Visualizing the Loss Landscape of Neural Nets. 2017.