Introduction to Deep Neural Networks

Numerical Methods for Deep Learning

Why Deep Networks?

- Universal approximation theorem of NN suggests that we can approximate **any** function by two layers.
- But - The width of the layer can be very large $\mathcal{O}(n \cdot n_f)$
- Deeper architectures can lead to more efficient descriptions of the problem.
- No real proof but lots of practical experience.

Deep Neural Networks

How deep is deep?
We will answer this question later ...

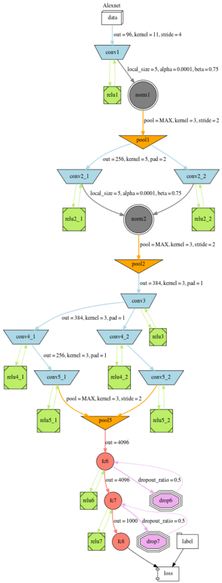
Until recently, the standard architecture was

$$
\begin{aligned}
\mathbf{Y}_1 &= \sigma(\mathbf{K}_0\mathbf{Y}_0 + b_0) \\
\vdots &= \vdots \\
\mathbf{Y}_N &= \sigma(\mathbf{K}_{N-1}\mathbf{Y}_{N-1} + b_{N-1})
\end{aligned}
$$

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

$$\min_{\mathbf{K}_{0,\ldots,N-1},\mathbf{b}_{0,\ldots,N-1},\mathbf{W}} \; E\left(\mathbf{W}\mathbf{Y}_N(\mathbf{K}_1,\ldots,\mathbf{K}_{N-1},b_1,\ldots,b_{N-1}),\mathbf{C}^{\mathrm{obs}}\right)$$

Example: The Alexnet [8] for Image Classification

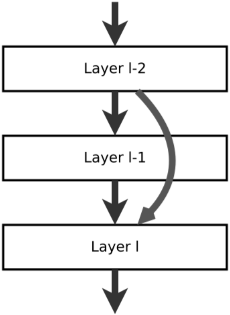


- Complex architectures
- trained on multiple GPUs
- $\approx$ 60 million weights

Deep Neural Networks in Practice

(Some) challenges:

- 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] came with a new architecture that solves many of the problems

Simplified Residual Neural Network

Residual Network

$$
\begin{aligned}
\mathbf{Y}_1 &= \mathbf{Y}_0 + \sigma(\mathbf{K}_0 \mathbf{Y}_0 + b_0) \\
\vdots &= \vdots \\
\mathbf{Y}_N &= \mathbf{Y}_{N-1} + \sigma(\mathbf{K}_{N-1} \mathbf{Y}_{N-1} + b_{N-1})
\end{aligned}
$$

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

$$
\min_{\mathbf{K}_{0,\ldots,N-1}, \mathbf{b}_{0,\ldots,N-1}, \mathbf{W}} \; E\left(\mathbf{W}\mathbf{Y}_N(\mathbf{K}_1, \ldots, \mathbf{K}_{N-1}, b_1, \ldots, b_{N-1}), \mathbf{C}^{\mathrm{obs}}\right)
$$

Leads to smoother objective function [9].

Stability of Deep Residual Networks

Why are ResNets more stable?
A small change

$$\begin{aligned}
\mathbf{Y}_1 &= \mathbf{Y}_0 + h\sigma(\mathbf{K}_0\mathbf{Y}_0 + b_0) \\
\vdots &= \vdots \\
\mathbf{Y}_N &= \mathbf{Y}_{N-1} + h\sigma(\mathbf{K}_{N-1}\mathbf{Y}_{N-1} + b_{N-1})
\end{aligned}$$

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

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

We can understand the behavior by learning the dynamics of nonlinear ODEs [5, 4].

Crash Course on ODEs

Given the ODE

$$\dot{\mathbf{y}} = f(t, \mathbf{y})$$

Assumptions:

1. f differentiable with Jacobian

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

2. $\mathbf{J}$ changes sufficiently slowly in time

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

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

Stability of Residual Network

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

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

The Jacobian is

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

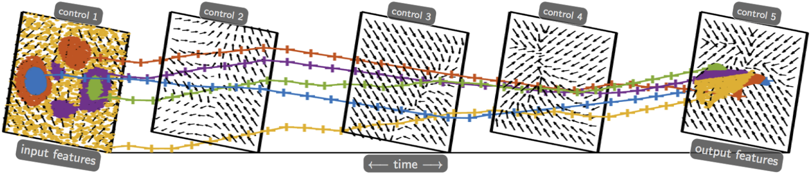
Here, $\sigma'(x) \geq 0$ for tanh, ReLU, ...

Hence, problem is stable when

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

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

Residual Network as a Path Planning Problem



Forward propagation in residual network (continuous)

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

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

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

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

And assume we use the forward Euler method to discretize

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + h\sigma(\mathbf{K}_j\mathbf{Y}_j + b_j)$$

Is the network stable?

Not always ...

Stability: A Simple Example

Look at the simplest possible forward propagation

$$\dot{\mathbf{Y}}(t) = \lambda \mathbf{Y}(t)$$

And assume we use the forward Euler to discretize

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + h\lambda \mathbf{Y}_j = (1 + h\lambda)\mathbf{Y}_j$$

Then the method is stable only if

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

Not every network is stable! Time step size depends on our Jacobian

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 θ.

Assume we use the forward Euler to discretize

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + h\sigma((\mathbf{K}_i - \mathbf{K}_i^\top)\mathbf{Y}_i + b_i)$$

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

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] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 61:10971105, 2012.

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