Neural Tangent Kernel: Convergence and Generalization in Neural Networks

Arthur Jacot
École Polytechnique Fédérale de Lausanne
arthur.jacot@netopera.net

Franck Gabriel
Imperial College London and École Polytechnique Fédérale de Lausanne
franckrgabriel@gmail.com

Clément Hongler
École Polytechnique Fédérale de Lausanne
clement.hongler@gmail.com

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

Why this paper

Important topic in studying:

- the learning dynamics of machine learning models with gradient decent.
- the effect of initialization of parameters, model architecture and training data on training dynamics.
- how learning from one data sample effects the predictions for other samples.

Some Math

Loss function

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(f(\mathbf{x}^{(i)}; \theta), y^{(i)})$$

Gradient descent:

$$\theta_{i+1} = \theta_i - \eta \nabla_\theta \mathcal{L}(\theta) \longrightarrow \frac{\theta_{i+1} - \theta_i}{\eta} = -\nabla_\theta \mathcal{L}(\theta)$$

With infinitesimally small learning rate the dynamics of training in parameter space is:

$$\frac{d\theta}{dt} = -\nabla_\theta \mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \nabla_\theta f(\mathbf{x}^{(i)}; \theta) \nabla_f \ell(f, y^{(i)})$$

Therefore the dynamics of training in function space is:

$$\frac{df(\mathbf{x};\theta)}{dt} = \frac{df(\mathbf{x};\theta)}{d\theta} \frac{d\theta}{dt} = -\frac{1}{N} \sum_{i=1}^{N} \underbrace{\nabla_\theta f(\mathbf{x};\theta)^\top \nabla_\theta f(\mathbf{x}^{(i)};\theta)}_{\text{Neural tangent kernel}} \nabla_f \ell(f, y^{(i)})$$

ref: https://lilianweng.github.io/posts/2022-09-08-ntk/

Take home message

At the infinite width limit:

- The Neural Tangent Kernel remains constant during training.
- It is irrelevant to network initialization and depends on model architecture and training data.
- The network function follows a linear differential equation during training, this enables simple closed form equation to describe the training dynamics.

An example: Neural Tangent Kernel (NTK) for PINNs

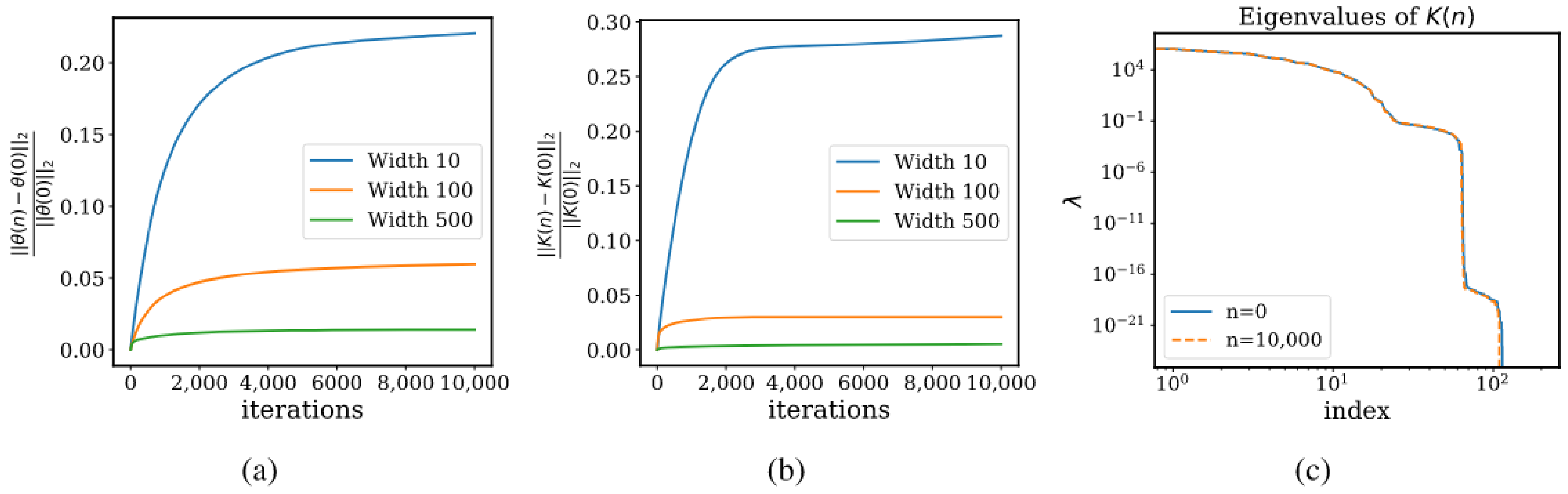

Fig. 2. *Model Problem 7.1 (1D Poisson equation)*: (a) (b) The relative change of parameters $\boldsymbol{\theta}$ and the NTK of PINNs $\boldsymbol{K}$ obtained by training a fully-connected neural network with one hidden layer and different widths (10, 100, 500) via 10,000 iterations of full-batch gradient descent with a learning rate of 10^{-5}. (c) The eigenvalues of the NTK $\boldsymbol{K}$ at initialization and at the last step ($(n = 10^4)$ of training a width $= 500$ fully-connected neural network.

Ref: {Wang, S., Yu, X., & Perdikaris, P. (2022). When and why PINNs fail to train: A neural tangent kernel perspective. Journal of Computational Physics, 449, 110768.}

Neural Tangent Kernel (NTK)

$$\frac{df(\mathbf{x};\theta)}{dt} = \frac{df(\mathbf{x};\theta)}{d\theta}\frac{d\theta}{dt} = -\frac{1}{N}\sum_{i=1}^{N}\underbrace{\nabla_\theta f(\mathbf{x};\theta)^\top \nabla_\theta f(\mathbf{x}^{(i)};\theta)}_{\text{Neural tangent kernel}}\nabla_f \ell(f, y^{(i)})$$

Kernel: a function to compare similarity of two data points:

$$K(\mathbf{x},\mathbf{x}';\theta) = \nabla_\theta f(\mathbf{x};\theta)^\top \nabla_\theta f(\mathbf{x}';\theta)$$

$$K : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}^{n_L \times n_L}$$

n_0: input dimension
n_L: output dimension
of network with L layers

For MSE loss:

$$\nabla_\theta \mathcal{L}(\theta) = f(\mathcal{X};\theta) - \mathcal{Y}, \longrightarrow$$

$$\frac{df(\theta)}{dt} = K_\infty(f(\theta) - \mathcal{Y})$$

$$\frac{dg(\theta)}{dt} = K_\infty g(\theta) \qquad ;\ \text{let } g(\theta) = f(\theta) - \mathcal{Y}$$

$$\int \frac{dg(\theta)}{g(\theta)} = \int K_\infty dt$$

$$\boxed{g(\theta) = Ce^{-\eta K_\infty t}}$$

The closed form equation for training dynamics

References

- https://proceedings.neurips.cc/paper/2018/hash/5a4be1fa34e62bb8a6ec6b91d2462f5a-Abstract.html
- https://en.wikipedia.org/wiki/Neural_tangent_kernel
- https://iclr.cc/virtual_2020/poster_SklD9yrFPS.html
- https://rajatvd.github.io/NTK/
- https://lilianweng.github.io/posts/2022-09-08-ntk/
- https://appliedprobability.blog/2021/03/10/neural-tangent-kernel/
- https://www.inference.vc/neural-tangent-kernels-some-intuition-for-kernel-gradient-descent/

PyTorch implementation:

- https://pytorch.org/tutorials/intermediate/neural_tangent_kernels.html