NTK, its application on PINN and multi-scale NN

Steven Chiu (chiu26@llnl.gov)

The original NTK paper is Jacot, Gabriel, and Hongler (2018), and this tutorial Weng (2022) has great exlaination.

Kernel Methods

Kernel methods: The kernel is a similarity function between two data points: $K : \chi \times \chi \to \mathbb{R}$. It describes how sensitive the predition for one data sample to predict for the other:

- Kernel is symmetric: $K(x, x') = K(x', x)$

Kernel methos is non-parametric, instant-based machine learning algorithms. Suppose we know all labels from training samples $\{x^{(i)}, y^{(i)}\}$.

- New input x: $x = \sum_i K(x^{(i)}, x) y^{(i)}$

Gaussian Processes: A non-parametric method by multivariate Gaussian probability. Suppose a given data points $\{x^{(1), \ldots, x^{(N)}}\}$.

- Covariance matrix: $\sum_{i,j} = K(x^{(i)}, x^{(j)})$

Neural tangent kernel (NTK)

Neural tangent kernel (NTK) (Jacot, Gabriel, and Hongler 2018) is for understanding neural network training via gradient descent.

Loss function: $\mathcal{L} : \mathbb{R}^P \to \mathbb{R}_+$:

$$\mathbb{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(f(x^{(i)};\theta)\nabla_f l(f, y^{(i)}))$$

the gradient of the loss is:

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\nabla_\theta f(x^{(i)};\theta)}_{\text{size } P\times n_L} \nabla_f \underbrace{l(f, y^{(i)})}_{\text{size } n_L\times 1} \tag{1}$$

Neural tangent: for tracking how network parameter θ evolve in time, the Equation **??** becomes an OD system of θ:

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

By the chain rule again,

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

Neural Tangent Kernel (NTK) is described as

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

where for each entry in the output matrix at location $(m, n), 1 \le m, n \le n_L$

$$K_{m,n}(\mathbf{x}, \mathbf{x}';\theta) = \sum_{p=1}^{P} \frac{\partial f_m(\mathbf{x};\theta)}{\partial \theta_p}\frac{\partial f_n(\mathbf{x}';\theta)}{\partial \theta_p}$$

Infinite width networks

The output of L-layer network $f_i(x;\theta)$ for $i = 1, \dots, n_L$ are i.i.d centered Gaussian Process of covariance $\sum^{(L)}$, defined recursively as

$$
\begin{aligned}
\Sigma^{(1)}(\mathbf{x}, \mathbf{x}') &= \frac{1}{n_0}\mathbf{x}^\top \mathbf{x}' + \beta^2 \\
\lambda^{(l+1)}(\mathbf{x}, \mathbf{x}') &= \begin{bmatrix} \Sigma^{(l)}(\mathbf{x}, \mathbf{x}) & \Sigma^{(l)}(\mathbf{x}, \mathbf{x}') \\ \Sigma^{(l)}(\mathbf{x}', \mathbf{x}) & \Sigma^{(l)}(\mathbf{x}', \mathbf{x}') \end{bmatrix} \\
\Sigma^{(l+1)}(\mathbf{x}, \mathbf{x}') &= \mathbb{E}_{f\sim\mathcal{N}(0,\lambda^{(l)})}[\sigma(f(\mathbf{x}))\sigma(f(\mathbf{x}'))] + \beta^2
\end{aligned}
$$

NTK for PINNs

Let

- solution: $u(t) = u(x_b, \theta(t)) = \{u(x_b^i.\theta(t))\}_{i=1}^{N_b}$.
- DE: $\mathcal{L}u(x_r, \theta(t)) = \{\mathcal{L}u(x_r, \theta(t))\}_{i=1}^{N_r}$

where $\theta(t)$ is parameters of a NN that changes over gradient descent.

Lemma 0.1 (NTK for PINNs)**.** *Wang, Yu, and Perdikaris (2020)*

$$
\begin{bmatrix} \frac{du(x_b,\theta(t))}{dt} \\ \frac{d\mathcal{L}u(x_r,\theta(t))}{dt} \end{bmatrix} = -\begin{bmatrix} K_{uu}(t) & K_{ur}(t) \\ K_{ru}(t) & K_{rr}(t) \end{bmatrix} \cdot \begin{bmatrix} u(x_b, \theta(t)) - g(x_b) \\ \mathcal{L}u(x_r, \theta(t)) - f(x_r) \end{bmatrix}
$$

where $K_{ru}(t) = K_{ur}^T(t)$ and $K_{uu}(t) \in \mathbb{R}^{N_b \times N_b}$, $N_{ur}(t) \in \mathbb{R}^{N_b \times N_r}$, and $K_{rr}(t) \in \mathbb{R}^{N_r \times N_r}$ with (i, j)-th entry is given by

$$
\begin{aligned}
(K_{uu})_{ij}(t) &= \frac{du(x_b^i, \theta(t))}{d\theta}, \frac{du(x_b^i, \theta(t))}{d\theta} > \\
&= \sum_{\theta\in\Theta} \frac{du(x_b^i, \theta(t))}{d\theta} \cdot \frac{du(x_b^i, \theta(t))}{\theta}
\end{aligned}
$$

Usage

1. Descide proper structure of NN
2. Tuning coefficients of residual loss, boundary loss during training

How to measure NTK?

- Neural Network Gaussian Processes (NNGP): infinite wide Bayesian nerual network.
- Neural Tangent Kernel (NTK): Randomly initialized infinite wide neural networks trained with gradient descent.

1. Derive by hand
2. Use Monte Carlo approach

This notebook galllery has numerous examples about how to use NTK to analyze gradient descent:

https://github.com/google/neural-tangents/tree/main/notebooks

Examples:

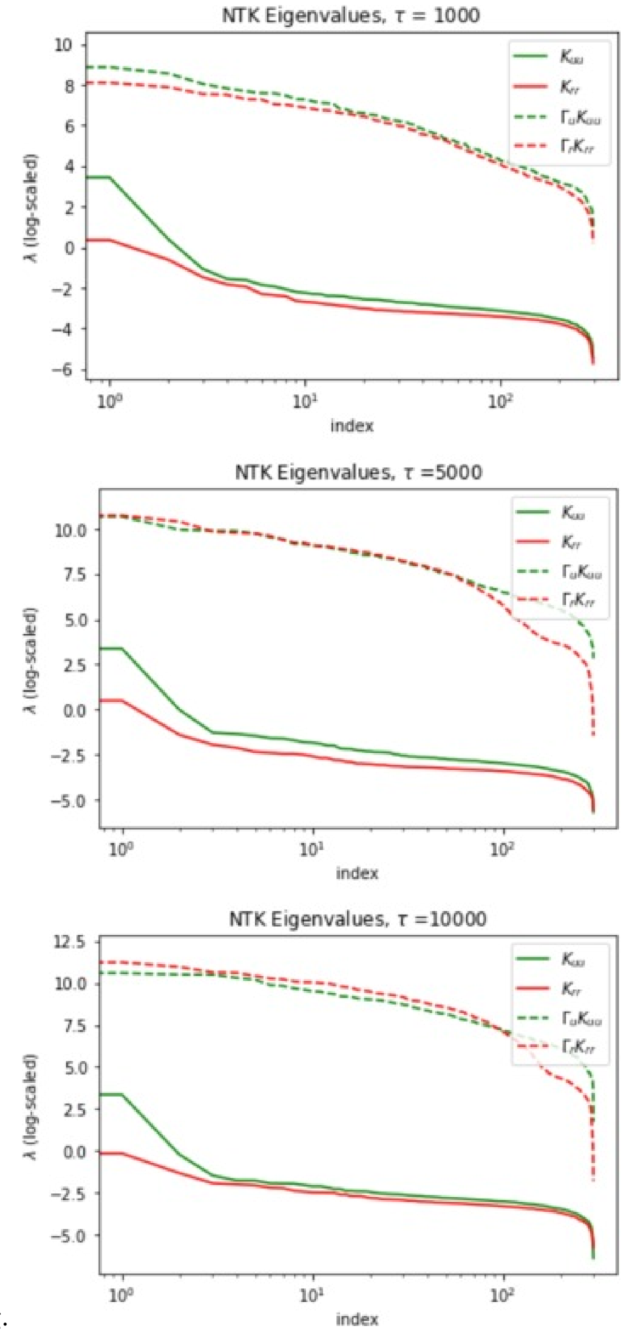
1. Why PINN sometimes fails? (Wang, Yu, and Perdikaris 2020): https://github.com/PredictiveIntelligenceL
2. Fourier feature (Tancik et al. 2020): https://github.com/tancik/fourier-feature-networks (notebook)

Examples of NTK

1. Propose a learning structure or loss functio
2. Derive the neural tangent kernel (NTK) of the structure.
3. Use empirical data to monitor the learning speed during the training.

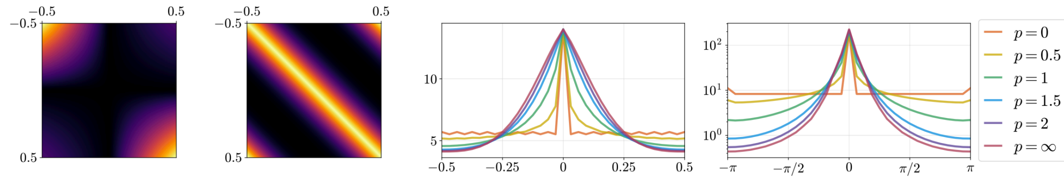
Examples:

1. McClenny and Braga-Neto (2023): Use NTK to explain the self adaptive weights help



training.

2. Tancik et al. (2020): Use NTK Fourier spectrum to measure the learning speed of the net-



(a) No mapping NTK (b) Basic mapping NTK (c) NTK spatial (d) NTK Fourier spectrum

Figure 2: Adding a Fourier feature mapping can improve the poor conditioning of a coordinate-based MLP's neural tangent kernel (NTK). (a) We visualize the NTK function $k_{\mathrm{NTK}}(x_i, x_j)$ (Eqn. 2) for a 4-layer ReLU MLP with one scalar input. This kernel is not shift-invariant and does not have a strong diagonal, making it poorly suited for kernel regression in low-dimensional problems. (b) A basic input mapping $\gamma(v) = [\cos 2\pi v, \sin 2\pi v]^{\mathrm{T}}$ makes the composed NTK $k_{\mathrm{NTK}}(\gamma(v_i), \gamma(v_j))$ shift-invariant (stationary). (c) A Fourier feature input mapping (Eqn. 5) can be used to tune the composed kernel's width, where we set $a_j = 1/j^p$ and $b_j = j$ for $j = 1, \ldots, n/2$. (d) Higher frequency mappings (lower p) result in composed kernels with wider spectra, which enables faster convergence for high-frequency components (see Figure 3).

work on spectrum.

Related works

Multi-scale Fourier Neural operator

The multi-scale DNN (You, Xu, and Cai 2024) is used for supporting the learning of high frequency component of the image. The resulting multi-scale FNO is used to solve oscillation function (Z. Liu, Cai, and Xu 2020).

- What is the relation between Multi-grid FNO (Guo and Li 2024)?