Neural Tangent Kernel

Insights from High-Dimensional Probability

Hooman Zolfaghari - Abdollah Zohrabi - Amirreza Velae

Sharif University of Technology

December 20, 2024

1. Introduction
2. Literature Review
3. Proposal
4. References

1 Introduction

2 Literature Review

3 Proposal

4 References

NTK

The **Neural Tangent Kernel (NTK)** captures the behavior of fully-connected deep nets in the infinite-width limit trained by gradient descent. [Jacot et al., 2018]

- An attraction is that a pure kernel-based method is used to capture the power of a fully-trained deep net of infinite width.
- NTK explains the evolution of neural networks during gradient descent training.
- Insight into why wide neural networks can consistently converge to a (?) minimum when minimizing empirical loss.

Problem Definition

A single hidden-layer neural network with i.i.d. random parameters, in the limit of infinite width, is a function drawn from a Gaussian process (GP) [Neal, 1996].

$$\Sigma^{(1)}(\mathbf{x}, \mathbf{x}') = \frac{1}{n_0}\mathbf{x}^\top \mathbf{x}' + \beta^2$$

$$\lambda^{(l+1)}(\mathbf{x}, \mathbf{x}') = \begin{bmatrix} \Sigma^{(l)}(\mathbf{x}, \mathbf{x}) & \Sigma^{(l)}(\mathbf{x}, \mathbf{x}') \\ \Sigma^{(l)}(\mathbf{x}', \mathbf{x}) & \Sigma^{(l)}(\mathbf{x}', \mathbf{x}') \end{bmatrix}$$

$$\Sigma^{(l+1)}(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{f \sim \mathcal{N}(0, \lambda^{(l)})}[\sigma(f(\mathbf{x}))\sigma(f(\mathbf{x}'))] + \beta^2$$

Problem Definition

- The NTK is different from this Gaussian process kernels, and is defined using the gradient of the output of the randomly initialized net with respect to its parameters
- At its core, it explains how updating the model parameters on one data sample affects the predictions for other samples.

Problem Definition

- Model Parameters $\theta \in \mathbb{R}^P$
- The empirical loss function $\mathscr{L} : \mathbb{R}^P \to \mathbb{R}_+$ to minimize during training is defined as follows, using a per-sample cost function $\ell : \mathbb{R}^{n_0} \times \mathbb{R}^{n_L} \to \mathbb{R}_+$

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

- According to the chain rule. the gradient of the loss is:

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

Problem Definition

- Each gradient descent update introduces a small incremental change of an infinitesimal step size

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

- Again, by the chain rule, the network output evolves according to the derivative:

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

Problem Definition

- Here we find the Neural Tangent Kernel (NTK), as defined in the blue part in the last formula: $K : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}^{n_L \times n_L}$

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

- Each entry:

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

- The "feature map" form of one input: $\varphi(\mathbf{x}) = \nabla_\theta f(\mathbf{x}; \theta)$

Problem Definition

- Most critical proposition from the NTK paper
- When $n_1, \ldots, n_L \to \infty$ (network with infinite width), the NTK converges to be:
  1. deterministic at initialization, meaning that the kernel is irrelevant to the initialization values and only determined by the model architecture; and
  2. stays constant during training.

- To give the formula of NTK, we also need to define a derivative covariance:

$$\dot{\Sigma}^{(h)}(x, x') = c_\sigma \mathbb{E}_{(u,v)\sim\mathcal{N}(0,\Lambda^{(h)})}\left[\dot{\sigma}(u)\dot{\sigma}(v)\right].$$

- The final NTK expression for the fully-connected neural network is

$$\Theta^{(L)}(x, x') = \sum_{h=1}^{L+1}\left(\Sigma^{(h-1)}(x, x') \cdot \prod_{h'=h}^{L+1} \dot{\Sigma}^{(h')}(x, x')\right),$$

where $\Sigma^{(h-1)}(x, x')$ and $\dot{\Sigma}^{(h')}(x, x')$ are layer-specific covariance and derivative covariance terms.

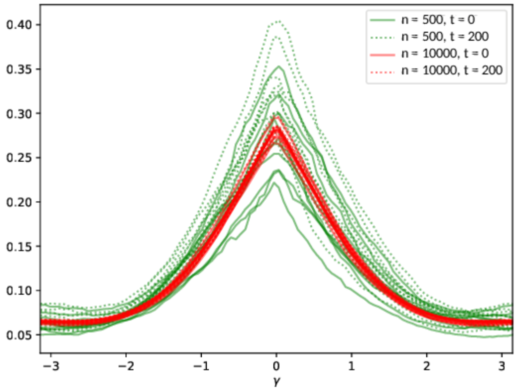
Problem Definition

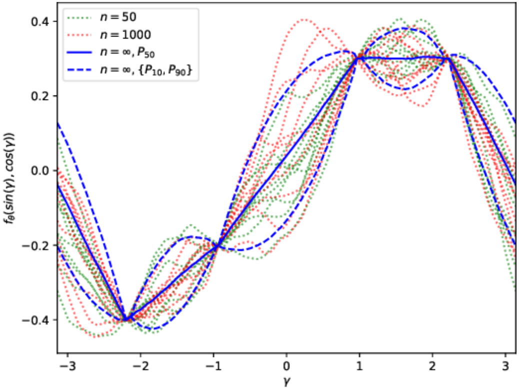
Theorem (Convergence to the NTK at initialization). Fix $\epsilon > 0$ and $\delta \in (0,1)$. Suppose

$$\sigma(z) = \max(0, z) \quad \text{and} \quad \min_{h \in [L]} d_h \geq \Omega\left(\frac{L^6}{\epsilon^4} \log\left(\frac{L}{\delta}\right)\right).$$

Then for any inputs $x, x' \in \mathbb{R}^{d_0}$ such that $\|x\| \leq 1, \|x'\| \leq 1$, with probability at least $1 - \delta$, we have:

$$\left| \left\langle \frac{\partial f(\theta, x)}{\partial \theta}, \frac{\partial f(\theta, x')}{\partial \theta} \right\rangle - \Theta^{(L)}(x, x') \right| \leq (L+1)\epsilon.$$


n = 500, t = 0
n = 500, t = 200
n = 10000, t = 0
n = 10000, t = 200
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
−3
−2
−1
0
1
2
3
γ


$n = 50$
$n = 1000$
$n = \infty, P_{50}$
$n = \infty, \{P_{10}, P_{90}\}$
$f_\theta(sin(\gamma), cos(\gamma))$
0.4
0.2
0.0
−0.2
−0.4
−3
−2
−1
0
1
2
3
γ

1. Introduction
2. Literature Review
3. Proposal
4. References

Exact Computation

- The first efficient exact algorithm for computing the extension of NTK to convolutional neural nets, which we call Convolutional NTK
- Theoretically, also gives the first non-asymptotic proof showing that a fully-trained sufficiently wide net is indeed equivalent to the kernel regression predictor using NTK.

NTK Sketch

Theorem Given $x, x' \in \mathbb{R}^d$ and $\delta \in (0,1), \epsilon \in (0, 1/L)$, the NTK Sketch computes $\Phi^{(L)} \in \mathbb{R}^m$, $m = \mathcal{O}\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right)$ in time

$$\tilde{O}\left(\frac{L^{11}}{\epsilon^{6.7}} + \frac{L^3}{\epsilon^2}d\right)$$

such that

$$\Pr\left[\left|\left\langle \Phi^{(L)}, \Phi^{(L)'} \right\rangle - K_{\mathrm{NTK}}^{(L)}(x,x')\right| \leq \epsilon \cdot K_{\mathrm{NTK}}^{(L)}(x,x')\right] \geq 1-\delta.$$

Finite Width

We provide quantitative bounds measuring the L_2 difference in function space between the trajectory of a finite-width network trained on finitely many samples from the idealized kernel dynamics of infinite width and infinite data.

Spectral Bias

- As an implication of these bounds, eigenfunctions of the NTK integral operator (not just their empirical approximations) are learned at rates corresponding to their eigenvalues
- We demonstrate that the network will inherit the bias of the kernel at the beginning of training even when the width only grows linearly with the number of samples

Generalization Bound

- Assuming:
- Fix failure probability $\delta \in (0,1)$
- data $S = \{(x_i, y_i)\}_{i=1}^n$
- Distribution D is $(\lambda_0, \delta/3, n)$-non-degenerate.
- $\kappa = \mathcal{O}\left(\frac{\lambda_0 \delta}{n}\right)$.
- Width $m \geq \kappa^{-2}\text{poly}(n, \lambda_0^{-1}, \delta^{-1})$.
- Loss function $\ell : \mathbb{R} \times \mathbb{R} \to [0,1]$ is 1-Lipschitz in the first argument such that $\ell(y, y) = 0$.
- Gradient descent runs for $k \geq \Omega\left(\frac{1}{\eta\lambda_0} \log \frac{n}{\delta}\right)$ iterations.

Generalization Bound

- Then with probability at least $1-\delta$ over the random initialization and the training samples,The two-layer neural network $f_{\mathbf{W}(k),\mathbf{a}}$ has population loss $L_D(f_{\mathbf{W}(k),\mathbf{a}}) = \mathbb{E}_{(\mathbf{x},y)\sim D}[\ell(f_{\mathbf{W}(k),\mathbf{a}}(\mathbf{x}), y)]$, bounded as:

$$L_D(f_{\mathbf{W}(k),\mathbf{a}}) \leq \sqrt{\frac{2\mathbf{y}^\top (\mathbf{H}^\infty)^{-1}\mathbf{y}}{n}} + O\left(\sqrt{\frac{\log \frac{n}{\lambda_0 \delta}}{n}}\right).$$

Uniform Convergence

- Overparametrized Multilayer Neural Networks: Uniform Concentration of Neural Tangent Kernel and Convergence of Stochastic Gradient Descent (2024)
- They first show that with random initialization, the NTK function converges to some deterministic function uniformly for all layers as the number of neurons tends to infinity
- Then they apply the uniform convergence result to further prove that the prediction error of multi-layer neural networks under SGD converges in expectation in the streaming data setting

Uniform Convergence

Theorem Under Gaussian initialization, for $m \geq Cd^2 \exp(L^2)$ for some constant C, there exist constants C_1, C_2, and C_3 such that, with probability at least $1 - \exp(-C_1 m^{1/3})$,

$$\left\| H^{(\ell)} - \Phi^{(\ell)} \right\|_\infty \leq C_2 \left(\frac{C_3^L}{m^{1/6}} + \sqrt{\frac{dL \log m}{m}} \right), \quad \forall 1 \leq \ell \leq L.$$

1 Introduction

2 Literature Review

3 Proposal

4 References

Our Proposal

- Guarantees with other/practical Assumptions
  - NTK-like results on Attention Mechanism
  - Extending current results on Gaussian/Bounded to sub-Gaussian or sub-Exponential
- NTK on Unsupervised settings like Generative modeling or Self-Supervised or Transfer learning
- NTK on other architectures like Kolmogorov Arnold Networks, Transformers etc
- NTK Spectral Distribution
- Optimization methods other than (Stochastic) Gradient Descent. e.g. Momentum SGD, using Long Search for GD.
- Certain activation functions. e.g. tanh
- Random number of neurons in Layers.
- Finding Sample complexity, VC-Dimensions, Rademacher Complexity

1. Introduction
2. Literature Review
3. Proposal
4. **References**

[1] L. Weng, “Some math behind neural tangent kernel,” *Lil’Log*, Sep 2022.

[2] A. Jacot, F. Gabriel, and C. Hongler, “Neural tangent kernel: Convergence and generalization in neural networks,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[3] J. Lee, L. Xiao, S. S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington, “On exact computation with an infinitely wide neural net,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[4] Y. Cao and Q. Gu, “Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks,” *Proceedings of the 36th International Conference on Machine Learning (ICML)*, vol. 97, pp. 1047–1055, 2019.

[5] N. Rahaman, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville, “Spectral bias outside the training set for deep networks in the kernel regime,” *International Conference on Learning Representations (ICLR)*, 2019.

[6] G. Meanti, L. Rosasco, A. Rudi, and L. Carratino, “Scaling neural tangent kernels via sketching and random features,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 14536–14546, 2020.

[7] S. Arora, S. S. Du, W. Hu, Z. Li, and R. Wang, “Overparametrized multi-layer neural networks: Uniform concentration of neural tangent kernel and convergence of stochastic gradient descent,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.