



Neural Tangent Kernel: Convergence and Generalization in Neural Networks¹

Kailong Wang[†]

[†]Rutgers University

August 24, 2023

¹Arthur Jacot, Franck Gabriel, and Clément Hongler. *Neural Tangent Kernel: Convergence and Generalization in Neural Networks*. Feb. 10, 2020. arXiv: 1806.07572 [cs, math, stat]. URL: <http://arxiv.org/abs/1806.07572> (visited on 08/13/2023). preprint.



On the equivalence of Neural Network and Kernel Machine

On the equivalence at initialization—Infinity width Multi-Layer Perceptron (MLP) initialize with Gaussian weights and biases is equivalent to a Gaussian Process.²

On the equivalence at training—Infinity width MLP with ReLU activation under squared loss trained by gradient descent is equivalent to a Gaussian Process.³

On the equivalence at training—Infinite width MLP with ReLU activation under soft margin loss trained by subgradient descent is equivalent to a support vector machine.⁴

²Jaehoon Lee et al. *Deep Neural Networks as Gaussian Processes*. Mar. 2, 2018. arXiv: 1711.00165 [cs, stat]. URL: <http://arxiv.org/abs/1711.00165> (visited on 07/08/2023). preprint.

³Jacot, Gabriel, and Hongler, *Neural Tangent Kernel*.

⁴Yilan Chen et al. *On the Equivalence between Neural Network and Support Vector Machine*. Nov. 11, 2021. arXiv: 2111.06063 [cs, math, stat]. URL: <http://arxiv.org/abs/2111.06063> (visited on 05/27/2022). preprint.

Table of Contents

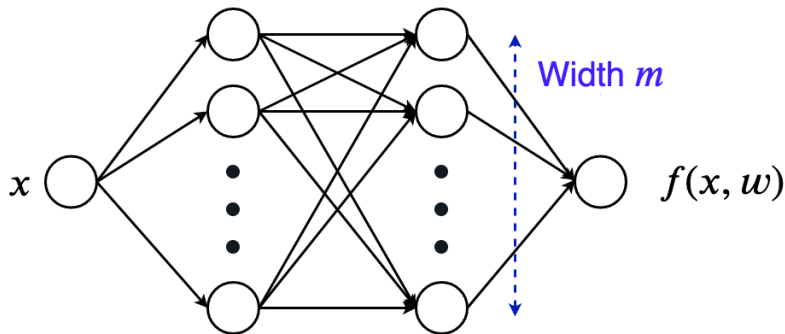
① Notations

② Neural Tangent Kernel (NTK)

③ Bridge the Gap



The Architecture of a Multi-Layer Perceptron (MLP)



Notations—Parameters Space

L : the depth of the network (*a.k.a.* the number of layers). *E.g.* 0 is the input layer, L is the output layer.

$N^{(L)}$: the width of the network (*a.k.a.* the number of neurons in each layer). *E.g.* $N^{(0)}$ is the dimension of the input, $N^{(L)}$ is the dimension of the output.

P : the total number of parameters, which is $\sum_{l=1}^L N^{(l-1)} N^{(l)}$.

The parameters between each layer are denoted as $W^{(l)} \in \mathbb{R}^{N^{(l-1)} \times N^{(l)}}$ and $b^{(l)} \in \mathbb{R}^{N^{(l)}}$, which are initialized as i.i.d. $\mathcal{N}(0, 1)$.



Notations—Function Space

σ : the activation function. *E.g.* $\sigma(x) = \max\{0, x\}$ is the ReLU.

\mathcal{F} : the function space of the MLP. *E.g.* $\mathcal{F} = \{f : \mathbb{R}^{N^{(0)}} \rightarrow \mathbb{R}^{N^{(L)}}\}$.

$\tilde{f}_\theta^{(l)}(x) : \mathbb{R}^{(l-1)} \rightarrow \mathbb{R}^{(l)}$: the pre-activations

$f_\theta^{(l)}(x) : \mathbb{R}^{(l)} \rightarrow \mathbb{R}^{(l)}$: the post-activations

Clearly, we have $\theta \in \mathbb{R}^P$ and

$$f_\theta^{(0)}(\mathbf{x}) = \mathbf{x}$$

$$\tilde{f}_\theta^{(l)}(\mathbf{x}) = W^{(l)\top} f_\theta^{(l-1)}(\mathbf{x}) + b^{(l)}$$

$$f_\theta^{(l)}(\mathbf{x}) = \sigma(\tilde{f}_\theta^{(l)}(\mathbf{x}))$$



Gradient Descent—Parameter Space

For a task with input $x \in \mathbb{R}^{n \times N^{(0)}}$ and output $y \in \mathbb{R}^{n \times N^{(L)}}$, the total loss \mathcal{L} is defined with per-sample loss ℓ

$$\mathcal{L}(\theta) = \operatorname{argmin}_{\theta} \frac{1}{n} \sum_{i=1}^n \ell(\mathcal{F}(\mathbf{x}_i), y_i).$$

According to the chain rule, the gradient of the loss is

$$\nabla_{\theta} \mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^n \nabla_{\theta} \mathcal{F}(\mathbf{x}_i) \nabla_{\mathcal{F}} \ell(\mathcal{F}, y_i).$$

If we let the gradient descent update the parameters with infinitesimal step size, then the gradient descent can be approximated as a derivative on the time dimension t

$$\frac{d\theta}{dt} = -\nabla_{\theta} \mathcal{L}(\theta) = -\frac{1}{n} \sum_{i=1}^n \nabla_{\theta} \mathcal{F}(\mathbf{x}_i) \nabla_{\mathcal{F}} \ell(\mathcal{F}, y_i).$$



Gradient Descent—Function Space

Then for any pair $(\mathbf{x}^*, \mathbf{x}_i)$, the output of the network slowly evolves as

$$\frac{d\mathcal{F}(\mathbf{x}^*)}{dt} = \frac{d\mathcal{F}(\mathbf{x}^*)}{d\theta} \frac{d\theta}{dt} = -\frac{1}{n} \sum_{i=1}^n \nabla_{\theta} \mathcal{F}(\mathbf{x}^*)^T \nabla_{\theta} \mathcal{F}(\mathbf{x}_i) \nabla_{\mathcal{F}} \ell(\mathcal{F}, y_i).$$

And the inner product of the gradient defines the Neural Tangent Kernel (NTK), which is also called kernel gradient

$$K(\mathbf{x}^*, \mathbf{x}_i) = \nabla_{\theta} \mathcal{F}(\mathbf{x}^*)^T \nabla_{\theta} \mathcal{F}(\mathbf{x}_i)$$

With random features⁵ of p (where $p \ll P$) samples of function space \mathcal{F}_k^p , these functions define a linear parametrization: $\mathbb{R}^P \rightarrow \mathcal{F}$ and the NTK can be approximated as

$$K(\mathbf{x}^*, \mathbf{x}_i) \approx \mathbb{E} [f_k^p(\mathbf{x}^*)^T f_k^p(\mathbf{x}_i)].$$

⁵Ali Rahimi and Benjamin Recht. “Random Features for Large-Scale Kernel Machines”. In: ().

The good and the bad

Recall the knowledge of Convex Optimization, the gradient descent converges to the global minimum of the convex loss function. For NTK,

The Good: For the MLP trained by gradient descent, the network output slowly evolves along the (negative) kernel gradient with respect to the neural tangent kernel (NTK) during training.

The Bad: The MLP (not to mention other NN architectures) is usually non-convex and the linear parametrization does not exist. Which leads to unstable NTK at initialization and during training.

Question: Can we prove the NTK is deterministic and staying constant, for example in an asymptotic manner?

NTK becomes deterministic at initialization⁶

Theorem: Neural Network as Gaussian Process

For a network of depth L at initialization, with a Lipschitz nonlinearity σ , and in the limit as the layers width $n_1, n_2, \dots, n_L \rightarrow \infty$, the NTK $\Theta^{(L)}$ converges in probability to a deterministic limiting kernel:

$$\Theta^{(L)} \rightarrow \Theta_{\infty}^{(L)} \otimes I_{N_L}.$$

The scalar kernel $\Theta_{\infty}^{(L)} : \mathbb{R}^{N_0} \times \mathbb{R}^{N_0} \rightarrow \mathbb{R}$ is defined recursively as

$$\begin{aligned}\Theta_{\infty}^{(1)}(\mathbf{x}^*, \mathbf{x}_i) &= \Sigma_{\infty}^{(1)}(\mathbf{x}^*, \mathbf{x}_i) \\ \Theta_{\infty}^{(L+1)}(\mathbf{x}^*, \mathbf{x}_i) &= \Theta_{\infty}^{(L+1)}(\mathbf{x}^*, \mathbf{x}_i) \dot{\Sigma}_{\infty}^{(L+1)}(\mathbf{x}^*, \mathbf{x}_i) + \Sigma_{\infty}^{(L+1)}(\mathbf{x}^*, \mathbf{x}_i) \\ \dot{\Sigma}_{\infty}^{(L+1)}(\mathbf{x}^*, \mathbf{x}_i) &= \mathbb{E}_{f \sim \mathcal{N}(0, \Sigma^{(L)})} [\dot{\sigma}(f(\mathbf{x}^*)) \dot{\sigma}(f(\mathbf{x}_i))]\end{aligned}$$

⁶Lee et al., *Deep Neural Networks as Gaussian Processes*.

The NTK is constant during training

Previously, we have shown that (omit the inputs for simplicity)

$$\frac{d\mathcal{F}_\theta}{dt} = -\alpha \mathbf{K}_\infty \nabla_f \mathcal{L}.$$

To track the evolution of θ in time, we can consider it as a function of time t . With Taylor expansion, we have⁷

$$\begin{aligned}\mathcal{F}_\theta(t) &= \mathcal{F}_\theta(0) + \nabla_\theta \mathcal{F}_\theta(0) [\theta(t) - \theta(0)] \\ \theta(t) - \theta(0) &= -\alpha \nabla_\theta \mathcal{F}_\theta^\top \nabla_f \mathcal{L} \\ \mathcal{F}_\theta(t) - \mathcal{F}_\theta(0) &= -\alpha \nabla_\theta \mathcal{F}_\theta(0)^\top \nabla_\theta \mathcal{F}(\mathbf{X}; \theta(0)) \nabla_f \mathcal{L} \\ \frac{d\mathcal{F}_\theta(t)}{dt} &= -\alpha K(\theta(0)) \nabla_f \mathcal{L} \\ &= -\alpha \mathbf{K}_\infty \nabla_f \mathcal{L}\end{aligned}$$

⁷Jaehoon Lee et al. “Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2020.12 (Dec. 1, 2020), p. 124002. ISSN: 1742-5468. DOI: 10.1088/1742-5468/abc62b. arXiv: 1902.06720 [cs, stat]. URL: <http://arxiv.org/abs/1902.06720> (visited on 08/13/2023).

The light that is Shed by the NTK

With the equivalence of the NTK and the Gaussian Process, Bayesian Inference can be applied to explain the generalization of the neural network.

For the same reason, the computation technique of Bayesian Inference (e.g. MCMC) can be applied to the neural network.

Since NTK is the gradient defined on the function space, calculus of variations can be applied to the neural network.

Other analytic tools such as Spectrum Analysis or Random Matrix can be applied to find tighter bounds.

The Undiscovered Area

Empirical results shows that the Neural Network optimized by NTK does not achieve State-of-the-Art (SOTA) performance.

The condition of the equivalence of NTK and Gaussian Process is too strong, which might not be practical.

The NTK of other architectures (e.g. CNN, RNN) is not well studied (Has been majorly solved in 2023).