Notes on "Neural Tangent Kernel: Convergence and Generalization in Neural Networks" by Jacot et al., 2018

Junghyun Lee[1,2]

[1]Department of Mathematical Sciences
[2]School of Computing
KAIST, Daejeon, South Korea
jh_lee00@kaist.ac.kr

May 27, 2021

This note will cover the first paper[JGH18] that introduced the neural tangent kernel (NTK). In my opinion, the paper is written in a very mathematically (and physically[1]) intricate manner, but with not so much intuition. Many blogs and articles [Dwa19, DH20, Hus20, Pé19] provide excellent expositions to the intuition behind NTKs, but not much materials are available for understanding the paper as it is with all the mathematical subtleties[2].

This note is meant to serve as a companion to reading [JGH18]; this note will supplement the paper with logical gaps, missing mathematical definitions, missing calculation steps...etc. (Some of the elementary proofs will be left as exercises, although the solutions will be provided at the Appendix) Also, borrowing from the mentioned blogs and articles, this note will also cover the intuition behind NTK and some of its implications in the connection between kernel theory and deep learning.

1 Basic Setup (ANN)

Here, we only consider fully-connected networks of depth L, as shown in Figure ?.

2 Deep Learning as Optimization over Function Space

2.1 Essential Functional Analysis

For each given parameter vector $\theta \in \mathbb{R}^P$, the corresponding neural network f_θ can be regarded as an element of the function space $\mathcal{F} = \{f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}\}$, which is basically the collection of all possible functions from $\mathbb{R}^{n_0}$ to $\mathbb{R}^{n_L}$. Let p^{in} be a fixed probability measure over the input space $\mathbb{R}^{n_0}$. Then we may consider the input space as a probability space i.e. $X := (\mathbb{R}^{n_0}, \mathcal{L}_{n_0}, p^{in})$ where $\mathcal{B}_k := \mathcal{L}(\mathbb{R}^k)$ is the usual Lebesgue

[1]as in physics
[2]The closest resource was [Lee].

σ-algebra of $\mathbb{R}^k$. The output space can be considered as a Banach space with the usual Euclidean norm i.e. $Y = (\mathbb{R}^{n_L}, \|\cdot,\|)$.

We consider the following subspace of $\mathcal{F}$:

Definition 1. Bochner space $L^p() \subset \mathcal{F}$

Remark 1. Obviously, $L^p(X;Y)$ is a Banach space...

Really, each element of $L^2(X;Y)$ is an equivalence class...

Especially when $p = 2$, we have a Hilbert space:

Proposition 2. *The function $\langle\cdot,\cdot\rangle_{p^{in}} : L^2(X;Y) \times L^2(X;Y) \to \mathbb{R}$ defined as below, is an inner product.*

$$\langle f, g\rangle_{p^{in}} = \mathbb{E}_{x\sim p^{in}}\left[f(x)^\intercal g(x)\right] \tag{1}$$

Proof. Bilinearity follows directly from the linearity of expectation. Symmetry and positive definiteness are trivial. ■

Let $\|\cdot\|_{p^{in}}$ be the induced norm.

Considering the universal approximation properties of ANNs (see Appendix ? for further discussion) and above discussions, we may restrict ourselves to $L^2(X;Y)$, which we shall refer to as $\mathcal{F}$ from heron.

2.2 Overview

Before diving into more mathematical details, let us get a grip of the big picture, first. We usually think of deep learning as optimizing over $\mathbb{R}^P$ and finding some optimal θ^* with respect to some loss function C i.e.

$$\theta^* = \underset{\theta\in\mathbb{R}^P}{\operatorname{argmin}}\, C(\theta) \tag{2}$$

where $C(\theta)$ is the loss of the neural network constructed using the parameter vector θ. However, the curse of dimensionality and non-convexity of the loss surface deter us from studying the learning dynamics of deep learning. Here, we change the viewpoint of optimizing over the parameter space to **optimizing over the function space $\mathcal{F}$**. In other words, we consider C to be a cost functional[3] defined *on the function space* i.e. $C : \mathcal{F} \to \mathbb{R}$ and reformulate the optimization as follows:

$$\theta^* = \underset{\theta\in\mathbb{R}^P}{\operatorname{argmin}}\, C(f_\theta) = \underset{\theta\in\mathbb{R}^P}{\operatorname{argmin}}\, C \circ F^{(L)}(\theta) \tag{3}$$

Here, $F^{(L)} : \mathbb{R}^P \to \mathcal{F}$ is the ANN realization function that maps the parameter vector θ to the corresponding neural network f_θ. In this perspective, we observe that our difficulties mainly arose from the complexity of $F^{(L)}$, and thus, we can *exploit the intrinsic structure of C and $\mathcal{F}$ to gain a better understanding of the training dynamics of ANNs.*

Such exploitation is done in a kernel trick-fashioned way. Intuitively, we consider **neural network as a feature map**, and we construct the corresponding kernel that can be calculated without knowing what the neural network looks like With that kernel, we impose a topological structure on $\mathcal{F}$ induced by the kernel (spoiler: semi-inner product space) such that the complex training dynamics of ANN can be described by a simple dynamics on $\mathcal{F}$ w.r.t. that topology.

[3]Functional refers to any mapping that maps a function to a real number.

3 Kernel Gradient

We start with a basic definition:

Definition 3. The dual space of $\mathcal{F}$, denoted as $\mathcal{F}^*$, is defined as follows:

$$\mathcal{F}^* = \left\{\mu = \langle d, \cdot\rangle_{p^{in}} : \mathcal{F} \to \mathbb{R} \mid d \in \mathcal{F}\right\}$$

To deal with the cost functional $C : \mathcal{F} \to \mathbb{R}$, we need to first define the (functional) derivative of C. This is given by the following concept:

Definition 4. Let V, W be Banach spaces, and let $C : V \to W$ be a function. Then f is said to be **Fréchet differentiable** at $v \in V$ if there exists a bounded linear operator A : Fréchet derivative

Remark 2. In our case, $V = \mathcal{F}$

Theorem 5 (Riesz representation theorem)**.** *content...*

Because $\mathcal{F}$ is a Hilbert space, we have that by Riesz representation theorem, for all $f_0 \in \mathcal{F}$ there exists some corresponding dual element $d|_{f_0} \in \mathcal{F}$ such that $\partial_f^{in} C|_{f_0} = \langle d|_{f_0}, \cdot\rangle_{p^{in}}$ i.e. $\partial_f^{in} C|_{f_0} \in \mathcal{F}^*$.

Now let us introduce kernels:

Definition 6. A function $K : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}^{n_L \times n_L}$ is a **multi-dimensional kernel** if it is a symmetric tensor in $\mathcal{F} \otimes \mathcal{F}$ i.e. $K(x, x') = K(x', x)^\intercal$ for all $x, x' \in \mathbb{R}^{n_0}$.

Remark 3. The kernel method that we're familiar with (i.e. kernel PCA) deals with the case when $n_L = 1$, which is usually denoted with $k(x, x')$. The connection, however, may be deceiving and sometimes lead to serious confusion. For instance, since $K(x, x')$ is a matrix, one may try to relate it to the Gram matrix. However, Gram matrix is $[k(x_i, x_j)]_{ij}$, while $K(x, x')$ is more like[4] $\phi(x)\phi(x')^\intercal$ where $\phi : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ is the feature map associated with K. Gram matrix for K, thus, have to be defined in a different way, as we will see later on.

In this section, let K be a fixed multi-dimensional kernel.

3.1 Kernel gradient descent

Theorem 7 (Chain rule for Fréchet derivatives)**.** *content...*

Above equation motivates another Hilbertian structure on $\mathcal{F}$ as follows:

Proposition 8. *The function $\langle \cdot, \cdot\rangle_K : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$, defined as below, is an inner product.*

$$\langle f, g\rangle_K = \mathbb{E}_{x,x' \sim p^{in}} \left[f(x)^\intercal K(x, x') g(x')\right] \tag{4}$$

Proof. Bilinearity follows directly from the linearity of expectation. Symmetry and positive semi-definiteness are trivial. ■

Let $\|\cdot\|_K$ be the induced norm. Then Eq. ? can be rewritten as follows:

$$\partial_t C|_{f(t)} = -\|d|_{f(t)}\|_K^2. \tag{5}$$

[4] Think of it as a cross-similarity matrix in the feature space.

Therefore, convergence to a critical point of C is guaranteed if $\|d|_{f(t)}\|_K > 0$. But the functional derivative is defined in terms of

Definition 9. The kernel K is **positive definite** with respect to $\|\cdot\|_{p^{in}}$ if $\|f\|_{p^{in}} > 0 \Rightarrow \|f\|_K > 0$.

3.2 Random functions approximation

Under linear model assumption, given a multi-dimensional kernel K, we shall compute another kernel $\tilde{K}^{(P)}$ such that the ANN realizations follow the kernel gradient descent w.r.t. $\tilde{K}^{(P)}$ under gradient descent dynamics; in the limit $P \to \infty$, it is shown that $\tilde{K}^{(P)} \to K$ i.e. the kernel dynamics w.r.t. $\tilde{K}^{(P)}$ is an approximation to the limiting kernel dynamics w.r.t. K.

Let $\mathcal{D}$ be a fixed distribution on $\mathcal{F}$ whose non-centered covariance is given by K i.e.

$$\mathbb{E}_{f\sim\mathcal{D}}\left[f \otimes f\right] = K, \tag{6}$$

or equivalently,

$$\mathbb{E}_{f\sim\mathcal{D}}\left[f(x) \otimes f(x')\right] = \mathbb{E}_{f\sim\mathcal{D}}\left[f(x)f(x')^\intercal\right] = K(x,x') \quad x, x' \in X. \tag{7}$$

Now sample $\{f^{(p)}\}_{p=1}^P$ from $\mathcal{D}$ uniformly at random. Then these P functions define a natural random *linear* ANN realization function $F^{lin} : \mathbb{R}^P \to \mathcal{F}$ such that

$$\theta \mapsto f_\theta^{lin} = \frac{1}{\sqrt{P}} \sum_{p=1}^P \theta_p f^{(p)} \tag{8}$$

The $\frac{1}{\sqrt{P}}$ provides the necessary scaling of the final outcome so that we can later apply the law of large number as $P \to \infty$. Note that f_θ^{lin} is almost like a inner product between $\theta = (\theta_p)$ and $f = (f^{(p)})$, with an abuse of notation.

Recall that our goal is to optimize the cost $C \circ F^{lin} : \mathbb{R}^P \to \mathbb{R}$. Under gradient descent, each θ_p, which is *dependent on* t, satisfies $\partial_t \theta_p(t) = -\partial_{\theta_p}(C \circ F^{lin})(\theta(t))$. Applying 7 to LHS, we have that

$$-\partial_{\theta_p}(C \circ F^{lin})(\theta) = -\partial_f^{in} C|_{f_{\theta(t)}^{lin}} \left(\partial_{\theta_p} F^{lin}(\theta)\right) = -\left\langle d|_{f_\theta^{lin}}, \partial_{\theta_p} F^{lin}(\theta)\right\rangle_{p^{in}} = -\frac{1}{\sqrt{P}}\left\langle d|_{f_\theta^{lin}}, f^{(p)}\right\rangle_{p^{in}} \tag{9}$$

(Dependency on t is temporarily removed; θ and its components are still functions of t)

Now we can finally figure our how $f_{\theta(t)}^{lin}$ evolves w.r.t. t:

$$\partial_t f_{\theta(t)}^{lin} = \frac{1}{\sqrt{P}} \sum_{p=1}^P \partial_t \theta_p(t) f^{(p)} = -\frac{1}{\sqrt{P}} \sum_{p=1}^P \partial_{\theta_p}(C \circ F^{lin})(\theta(t)) f^{(p)} = -\frac{1}{P} \sum_{p=1}^P \left\langle d|_{f_{\theta(t)}^{lin}}, f^{(p)}\right\rangle_{p^{in}} f^{(p)} \tag{10}$$

Define the **(linear) tangent kernel at θ, denoted as $\tilde{K}_{\theta(t)}^{(P)}$,** as follows:

$$\tilde{K}_\theta^{(P)} = \sum_{p=1}^P \partial_{\theta_p} F^{lin}(\theta) \otimes \partial_{\theta_p} F^{lin}(\theta) = \frac{1}{P} \sum_{p=1}^P f^{(p)} \otimes f^{(p)} \tag{11}$$

Because the RHS is independent of θ, we drop the dependency on θ and write $\tilde{K}^{(P)}$. Note that by the law of large number, as $P \to \infty$, we have that $\tilde{K}^{(P)} \to K$.

Lastly, via a simple computation, we show that the negative of RHS of Eq. 10 is exactly the kernel gradient of C w.r.t. $\tilde{K}^{(P)}$ at $f^{lin}_{\theta(t)}$. Let $x' \in X$ be arbitrary, and let $d = d|_{f^{lin}_{\theta(t)}}$ for simplicity. Then,

$$\begin{aligned}
\left(\frac{1}{P}\sum_{p=1}^{P}\left\langle d, f^{(p)}\right\rangle_{p^{in}} f^{(p)}\right)(x') &= \frac{1}{P}\sum_{p=1}^{P}\mathbb{E}_{x\sim p^{in}}[d(x)^\intercal f^{(p)}(x)]f^{(p)}(x') \\
&= \mathbb{E}_{x\sim p^{in}}\left[\frac{1}{P}\sum_{p=1}^{P} f^{(p)}(x')f^{(p)}(x)^\intercal d(x)\right] \\
&= \mathbb{E}_{x\sim p^{in}}\left[\tilde{K}^{(P)}(x',x)d(x)\right] \\
&= \left\langle \tilde{K}^{(P)}(x',\cdot), d\right\rangle_{p^{in}} \\
&= \left(\Phi_{\tilde{K}^{(P)}}\left(\langle d,\cdot\rangle\right)\right)(x') \\
&= \left(\Phi_{\tilde{K}^{(P)}}\left(\partial^{in}_f C|_{f^{lin}_{\theta(t)}}\right)\right)(x') \\
&= \left(\nabla_{\tilde{K}^{(P)}} C|_{f^{lin}_{\theta(t)}}\right)(x')
\end{aligned}$$

Remark 4. This is a reminiscence of [RR07]; there, high-dimensional input data is mapped to a randomized low-dimensional feature space such that the inner product of the transformed data is approximately equal to the given kernel k, then fast linear methods were applied for downstream tasks. In this case, P-dimensional parameter vector is mapped to $\mathcal{F}$ via a random *linear* mapping using P random functions, whose tensor(outer) product with itself is approximately the given multi-dimensional kernel K. To summarize:

[RR07]	[JGH18]
Mapping to low-dimensional feature space $\mathbb{R}^D$	Mapping to function space $\mathcal{F}$
Inner product on $\mathbb{R}^D$ is similar to the given scalar kernel k	Tensor(outer) product on $\mathcal{F}$ is similar to the given multi-dimensional kernel K
Applying *linear* machine after feature mapping	Constructing random *linear* ANN realization function

4 Neural tangent kernel

Using the (almost) exact same reasoning as Section ?.?, we can show that the ANN f_θ trained using gradient descent on $C \circ F^{(L)}$ evolves along the kernel gradient descent

$$\partial_t f_{\theta(t)} = -\nabla_{\Theta^{(L)}_{\theta(t)}} C|_{f_{\theta(t)}} \tag{12}$$

w.r.t. the **neural tangent kernel (NTK)**, whose definition at θ is given as following:

$$\Theta^{(L)}_\theta = \sum_{p=1}^{P} \partial_{\theta_p} F^{(L)}(\theta) \otimes \partial_{\theta_p} F^{(L)}(\theta). \tag{13}$$

It should be obvious that $F^{(L)}$ is not linear, and thus, $\partial_{\theta_p} F^{(L)}(\theta)$ is dependent on θ, and so is NTK.

The NTK is therefore **random** at initialization and **varies** during training.

This is not good as such properties deter us from analyzing the training dynamics of ANN further. However, in the next two subsections, we describe how in the **infinite width limit**, such properties are replaced with much more desirable ones.

4.1 Initialization

In the **(sequential) infinite width limit**, at **random** $\mathcal{N}(0,1)$ **initialization**, the **NTK converges to a deterministic limiting kernel**, whose recursive computation involves averaging over some Gaussian processes i.e. random NTK $\rightarrow$ deterministic NTK.

4.2 Training

In the **(sequential) infinite width limit**, during **training**, the **NTK stays constant**, and thus gradient descent almost surely converges to some global minimum if the cost is convex (in $\mathcal{F}$) and the NTK is positive definite i.e. varying NTK $\rightarrow$ constant NTK.

5 Case Study: Least-squares regression

References

[DH20] Simon Du and Wei Hu. Ultra-Wide Deep Nets and the Neural Tangent Kernel (NTK), Jul 2020.

[Dwa19] Rajat Vadiraj Dwaraknath. Understanding the Neural Tangent Kernel, Nov 2019.

[Hus20] Ferenc Huszár. Some Intuition on the Neural Tangent Kernel, Nov 2020.

[JGH18] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[Lee] Jin Woo Lee. (Review) Neural Tangent Kernel.

[Pé19] Guillermo Valle Pérez. Notes on Neural Tangent Kernel: Convergence and Generalization in Neural Networks, by Jacot et al. Aug 2019. for the Fall2019 NTK reading group in Oxford.

[RR07] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

A Universal Approximation Properties of ANNs